1 og 0 Kennslubók í Python og Raspberry Pi Forritun

Sigurður Óskar Óskarsson

13. júlí 2025

Efnisyfirlit

1 Þakkir

Þessi kennslubók er enn í vinnslu (innsetning höfundar) Þessa kennslubók hefði ekki hægt að gera nema með aðstoð þeirra vina og kennara sem kenndu mér forritun og rafmagnsfræði í framhaldsskólum og háskólum, með þeirra aðstoð.Efnið kemur mikið frá þeim og kenndu við þá skóla og þá góða þekkingu sem þeir kenndu sem ég tek saman hér í kennslubók ásamt verkefnabók sem fylgir með.Ég vona að þú nemandi eigir eftir að hafa gaman og gaman, því sum verkefni eru það stór að aðstoð góðra vinnumanna og kennara er næstum ómöguleg, en takk fyrir vini og fyrrum kennara, Kafla um skynjara eru færðar sérstakar þakkir, þar sem hann var alfarið unnin af rafvirkjanemum við Menntaskólann á Ísafirði vorið 2025. Nemendurnir tóku að sér að vinna og setja saman efnið í þessum kafla og eiga þeir heiður skilið fyrir sitt framlag og frumkvæði.

Þakkir til:

Eiríkur Guðmundsson, Viktor Ýmir Elíasson, Tómas Guðmundsson, Hermann Ingjaldsson, Finnur Guðmundsson, Jóhann Karlsson, Tækniskóli (Skóli Atvinnulífs), Fjölbrautaskóla Vesturlands á Akranesi (FVA)., Pétur Örn Sigurðsson

2 Kynning

Python er frábært byrjendamál – það er auðvelt að lesa og skrifa, en jafnframt öflugt og mikið notað af sérfræðingum um allan heim. Raspberry Pi, á hinn bóginn, opnar dyr að skapandi verkefnum þar sem hugmyndir þínar verða að veruleika. Hvort sem þú hefur áhuga á að skrifa tölvuleiki, vinna með gögn, búa til öpp, tengja tæki saman eða stjórna rásum og skynjurum með Raspberry Pi, þá er þetta réttur staður til að byrja. Ekki hafa áhyggjur ef þú hefur aldrei forritað eða notað Raspberry Pi áður. Þessi bók er skrifuð með það í huga að allir geti fylgt með, óháð fyrri þekkingu. Lykilatriðið er að vera forvitin, spyrja spurninga og ekki gefast upp þó að þú rekist á hindranir. Það er eðlilegur hluti námsferlisins. Til að aðstoða þig enn frekar við námið hefur þessi bók verið hönnuð sem bæði lærdóms- og vinnubók. Þú getur skrifað glósur og svör beint í bókina, til að hjálpa þér að hugsa skipulega og skipuleggja verkefnin þín. Það er engin rétt eða röng leið – þetta er þitt ferðalag, og það er í þínum höndum að gera það skemmtilegt og fræðandi. Láttu þig dreyma stórt, prófaðu eitthvað nýtt, og njóttu þess að læra. Forritun er ekki bara um tölur og kóða – það er skapandi, lausnamiðuð hugsun sem getur hjálpað þér að sjá heiminn í nýju ljósi.Raspberry Pi er lítið tölvuborð sem hægt er að nota fyrir ýmis verkefni, frá einföldum forritunartilraunum til flókinna sjálfvirkniverkefna. Nú er komið að þér. Taktu skrefið og byrjaðu á þessari ferð – hún gæti verið fyrsta skrefið að óvæntri og spennandi framtíð.

Gangi þér vel!

- Sigurður Óskar Óskarsson

3 Hvað er Forritunarmál

Forritunarmál er aðferð sem við mennirnir notum til að geta talað við tölvur. Tölvur virka mjög ólíkt því hvernig við mennirnir virkum, við höfum gjörólíka styrki og veikleika. Tölvan hugsar í binary (röð af 0 og 1) á meðan við mennirnir hugsum í mannlegu máli. Þess vegna er mikið mál fyrir menn að geta talað með hætti sem tölvan skilur.

Einhvern veginn þurfum við að breyta okkar mannlega máli yfir í þennan 0 og 1 sem tölvan skilur. Það er ekki lítið mál. Til að áorka þessu notum við svokallað forritunarmál en forritunarmál er skilgreining á því hvað tákn sem við mennirnir skrifum þýða í heimi tölvunnar. Það er kortlagning frá einhvers konar mannamáli yfir í tölvumál.

3.1 Keyrsla forrits

Við skrifum skjal. Við byrjum efst og fikrum okkur svo niður, línu fyrir línu. Allt sem við skrifum verður að vera málfræðilega rétt samkvæmt forritunarmálinu sem við erum að skrifa í þá stundina. Alveg eins og grunneiningarnar hjá okkur eru setningar þá eru grunneiningarnar í forritunarmálinu, skipanir. Yfirleitt er ein skipun á hverri línu en það er þó alls ekki algilt.

Þegar við erum búin að segja það sem við viljum segja, vistum við skjalið og keyrum. Þá er þetta nýja skjal okkar þýtt yfir í skjal sem tölvan skilur (tvíundarskjal/binary), sem er svo keyrt í gegnum örgjörvann. Niðurstaðan, ef einhver, prentast svo á skjáinn hjá okkur.

3.2 Python er

Bakenda forritunarmál en það þýðir að maður keyrir það beint í tölvunni en ekki í vafra eins og Javascript. Þegar maður keyrir forrit á bakendanum þá hefur maður hefðbundinn aðgang að tölvunni eins og skjölum, jaðartækjum o.þ.h.

Skriptumál. Það þýðir að þú þarft ekki að sjá um að þýða forritin sem þú skrifar, yfir í svokallað tvíunda sem örgjörvinn getur svo keyrt. Forritunarmálið sér um það sjálft. Skriptumál eru yfirleitt hægvirkari því þau þurfa alltaf að hafa áhyggjur af þýðingunni á meðan þau eru að keyra skjalið.

4 Fyrstu skrefin

Nú ætlum við að setja upp tölvuna til að geta byrjað að forrita. Við byrjum á því að setja upp ritil og svo lærum við að ferðast um skjalakerfi tölvunnar okkar og endum svo á því að skrifa og keyra einfalt Python-forrit.

4.1 Ritlar

Það fyrsta sem við þurfum að gera er að öðlast getuna til að móta texta með þægilegum hætti. Það gerum við með svokölluðum ritli/editor. Það er til mikill fjöldi af ritlum og hvaða ritill hentar hverjum og einum er matsatriði. Eftirfarandi eru tenglar á nokkra valda ritla.

- Geany
- Kate
- Bluefish
- Vim (erfiður fyrst, bestur til langs tíma)
- Thorny Einnig er yfirgripsmikinn lista yfir ritla að finna á Wikipedia, hér.

Um aðskilnað ábyrgðar. Í hugbúnaðarþróun er það mikil speki að brjóta ábyrgðina upp eins mikið og mögulegt er. Með því er átt við að hver eind sjái bara um það sem henni ber og ekkert meir. Þetta á líka við um ritla. Ritlar eiga bara að vera það, ritlar. Þetta er forrit til að leyfa okkur að móta texta með þægilegum hætti. Eitthvað hefur orðið um það að ritlar reyni að stækka hlutverk sitt og sjá líka um t.a.m. keyrslu kóðans. Við vörum við þeirri þróun þar sem þarna er virknin að móta texta orðin samofin virkninni að keyra forrit.

Segjum að þú leggir einhverja orku í að læra um hvernig maður keyrir Python-forrit í viðkomandi ritli. Segjum svo að seinna viljirðu skipta um ritil þar sem nú er kominn einhver betri. Ætlarðu þá að fara að læra aftur hvernig maður keyrir Python-forrit? Ætlarðu alltaf að læra allt upp á nýtt í hvert skipti sem þú vilt skipta um ritil? Í hvert skipti sem þú vilt skipta út nokkrum sköpuðum hlut? Því meira magn af virkni sem er föst saman, því verra. Við viljum að þetta séu pínkulitlar einingar sem við getum hent inn og út og tengt saman eins og okkur sýnist.

Aðskiljum ábyrgðina og látum ritilinn bara vera það, ritill. Þess utan þá er aðskilnaður ábyrgðar algjör grunnspeki í allri hugbúnaðarþróun. Venjum okkur

strax á að hugsa svona, þá náum við meiri árangri til lengri tíma.

4.2 Windows eða Unix

Nú getum við skrifað skjöl í tölvunni okkar. Næst kemur að uppsetningunni til að geta keyrt skjölin með Python. Þá skiptir máli hvort við séum að nota Windows eða Unix. Til að vita hvort þú sért á Windows eða Unix þá er Unix allt sem er ekki Windows. Þetta er s.s. bara Windows og svo allir aðrir. Við munum fyrst fara yfir uppsetninguna fyrir Unix og svo Windows.

4.3 Unix

Ef þú ert með Unix þá notum við hið svokallaða terminal (ctrl-alt-t) en terminal-ið er forrit sem við notum til að stjórna tölvunni með texta. Til að nota terminalið þarftu að læra að ferðast um í skjalakerfinu, sem er útskýrt í Linux-kaflanum Ferðumst-um-tölvuna,hér. Í Unix er Python yfirleitt uppsett sjálfkrafa í byrjun. Til að keyra Python- forrit (sem þú gætir t.a.m. hafa skrifað í einum ritlanna hér að ofan), þarftu að opna terminal-ið og færa þig á möppuna sem þú vistaðir skjalið undir. Svo keyrirðu forritið með því að skrifa neðangreint. Þetta verður svo útskýrt betur í næsta kafla.

```
1 python skjal.py
```

4.4 Nauðsynlegar Linux skipanir að kunna

- **pwd**: Present Work Directory – Í hvaða möppu er ég?
- **clear**: Hreinsa skjá.
- **ls**: Listi yfir það sem er í möppunni þar sem þú ert staddur (list).
- **./**: Hér.
- **../**: Ein mappa niður (fjær rótinni).
- **man** *skipun*: Upplýsingar um skipun (manual).
- **info** *skipun*: Upplýsingar um skipun.
- **grep** *skipun*: Listar upp öll áðurkomin tilfelli fyrir þessa skipun.
- **cp**: Afrita.
- **mv**: Færa og/eða endurskýra.
- **rm**: Eyða.
- **rm -r**: Eyðir möppu og innihaldi.
- **mkdir**: Búa til möppu.
- **rmdir**: Eyða möppu (tómri).
- **cat**: Sýnir innhald textaskrár.
- **tail**: Endir skráa.
- **less**: Ein skjáfylli í einu.
- **Gæsalappir**:
  - `"breytur"`: Geymir breytur.
  - `'texti'`: Geymir texta.
- **Subshell**: Undirforrit.
- **chmod** [*dec/oct* eða bókstafir]: Breyta attribute á skrám.
- **chmod** +**x** *nafn skráar*: Gefur keyrslurétttindi á skrána.
- **chmod 777** *nafn skráar*: Gefur full réttindi á skrána (`7 = 111`).

5 Thonny

(Efnir er á ensku en hefur verið þýtt yrif á íslensku efitr Sigurður óskar.)

Ert þú Python-byrjandi að leita að tæki sem getur stutt nám þitt? Þessi grein er fyrir þig! Sérhver forritari þarf stað til að skrifa kóðann sinn. Þessi grein mun fjalla um frábært tól sem heitir Thonny sem gerir þér kleift að byrja að vinna með Python í byrjendavænu umhverfi. Í þessari grein muntu læra:

- Hvernig á að setja Thonny upp á tölvunni þinni
- Hvernig á að vafra um notendaviðmót Thonny til að nota innbyggða eiginleika þess
- Hvernig á að nota Thonny til að skrifa og keyra kóðann þinn
- Hvernig á að nota Thonny til að kemba kóðann þinn

Thonny er ókeypis Python Integrated Development Environment (IDE) sem var sérstaklega hannað með byrjendur Pythonista í huga. Nánar tiltekið er það með innbyggt villuleitarforrit sem getur hjálpað þér þegar þú lendir í viðbjóðslegum villum, og það býður upp á möguleika á að gera skref í gegnum tjáningarmat, meðal annarra virkilega frábærra eiginleika.

5.1 The beginner-Frendly python Editor

Hægt er að nálgast niðurhal á vefnum í gegnum vafra með því að fara á vefsíðu Thonny. Þegar þú ert kominn á síðuna muntu sjá ljósgráan kassa efst í hægra horninu, svona:

Mynd 5.1: mynd1

Þegar þú hefur fundið gráa reitinn skaltu smella á viðeigandi tengil fyrir stýrikerfið þitt. Þessi kennsla gerir ráð fyrir að þú hafir hlaðið niður útgáfu 3.0.1.

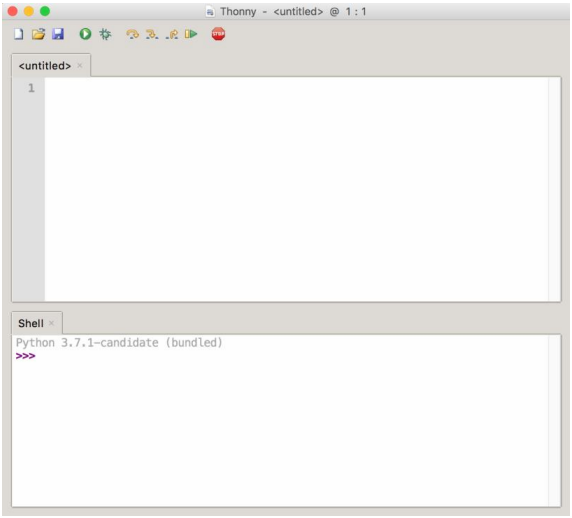
5.2 Command Line Download

Þú getur líka sett upp Thonny í gegnum skipanalínu (CMD) kerfisins þíns. Í Windows geturðu gert þetta með því að ræsa forrit sem heitir Command Prompt en á MacOS og Linux ræsirðu forrit sem heitir Terminal. Þegar þú hefur gert það skaltu slá inn eftirfarandi skipun:

```
$ pip install thonny
```

5.3 The User Interface

Við skulum ganga úr skugga um að þú skiljir hvað Thonny hefur upp á að bjóða. Hugsaðu um Thonny sem vinnuherbergið þar sem þú munt búa til ótrúleg Python-verkefni. Vinnuherbergið þitt inniheldur verkfærakistu sem inniheldur mörg verkfæri sem gera þér kleift að verða rokkstjarna/pythonista. Í þessum hluta muntu læra um hverja eiginleika notendaviðmótsins sem mun hjálpa þér að nota hvert verkfæri í Thonny verkfærakistunni þinni.



Mynd 5.2: Forrit Thonny

5.4 The Code Editor and Shell

Nú þegar þú hefur uppsett Python skaltu opna forritið. Þú ættir að sjá glugga með nokkrum táknum fyrir ofan og tvö hvít svæði:

Taktu eftir tveimur aðalhlutum gluggans. Efsti hlutinn er kóðaritarinn þinn, þar sem þú munt skrifa allan kóðann þinn. Neðri helmingurinn er skelin þín, þar sem þú munt sjá úttak úr kóðanum þínum.

5.5 The Icons

Mynd 5.3: Ták í Tornny

Yfir efstu muntu sjá nokkur tákn. Við skulum kanna hvað hver þeirra gerir. Þú munt sjá mynd af táknunum hér að neðan, með staf fyrir ofan hvert og eitt. Við munum nota þessa stafi til að tala um hvert af táknunum:

Við vinnum okkur frá vinstri til hægri, hér að neðan er lýsing á hverju tákni á myndinni.

A: Pappírstáknið Gert til að búa til nýja skrá. Venjulega í Python viltu aðgreina forritin þín í aðskildar skrár. Þú munt nota þennan hnapp síðar í kennslunni til að búa til fyrsta forritið þitt í Thonny!

B: Opna möpputáknið Gert til að opna skrá sem er þegar til á tölvunni þinni. Þetta er gagnlegt ef þú kemur aftur í forrit sem þú vannst við áður.

C: Disklingatáknið Notað til að vista kóðann þinn. Ýttu á þetta snemma og oft. Þú munt nota þetta síðar til að vista fyrsta Thonny Python-forritið þitt.

D: Spilatáknið Notað til að keyra kóðann þinn. Þegar þú keyrir kóðann þinn segirðu Python: „Gerðu það sem ég sagði þér að gera!“ eða „Lestu í gegnum kóðann minn og framkvæmdu það sem ég skrifaði.“

E: Villutáknið Gert til að kemba kóðann þinn. Það er óhjákvæmilegt að þú lendir í villum þegar þú skrifar kóða. Villuhnappur Thonny er notaður til að koma auga á og rannsaka villur, sem geta stafað af röngum rökum eða setningafræði. *(Skemmtileg staðreynd: Pöddur í forritun eru kallaðar „bugs“ vegna sögulegrar tengingar við tæknivandamál!)*

F-H: Örvatáknin Gera þér kleift að keyra forritin þín skref fyrir skref. Þetta getur verið gagnlegt við kembingu. Þessi tákn eru notuð eftir að þú ýtir á villutáknið. Gul auðkennd stika gefur til kynna hvaða línu eða hluta Python er að meta.

F: Örin segir Python að taka stórt skref, þ.e. hoppa í næstu línu eða kóðablokk.

G: Örin segir Python að taka lítið skref, þ.e. kafa djúpt í hvern hluta tjáningar.

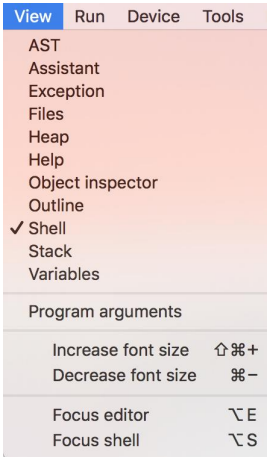
H: Örin segir Python að fara út úr kembiforritinu.

I: **Ferilskráartáknið** Gert til að fara aftur í spilunarham úr kembistillingu. Þetta er gagnlegt þegar þú vilt ekki lengur fara skref fyrir skref í gegnum kóðann og vilt frekar að forritið ljúki keyrslu.

J: **Stöðvunartáknið** Notað til að hætta að keyra kóðann þinn. Þetta er sérstaklega gagnlegt ef kóðinn þinn keyrir forrit sem opnar nýjan glugga og þú vilt stöðva það. Þú munt nota stöðvunartáknið síðar í kennslunni.

5.6 Other UI Features

Til að sjá meira af öðrum eiginleikum sem Thonny hefur upp á að bjóða skaltu fara á valmyndastikuna og velja View Töfragardína. Þú ættir að sjá að Shell er með gátmerki við hliðina á því, þess vegna sérðu Shell-hlutann í Thonny-forritsglugganum:



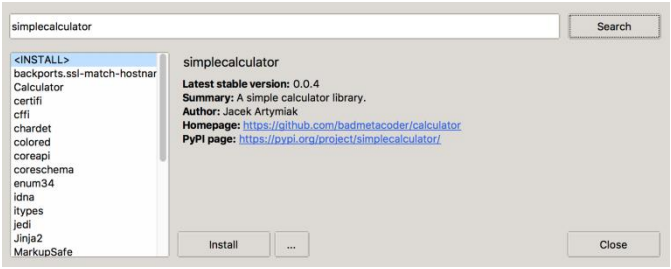
Mynd 5.4: Valgluggi

5.7 The Package Manager

Þegar þú heldur áfram að læra Python getur það verið mjög gagnlegt að hlaða niður Python-pakka til að nota inni í kóðanum þínum. Þetta gerir þér kleift að

nota kóða sem einhver annar hefur skrifað inn í forritið þitt. Íhugaðu dæmi þar sem þú vilt gera nokkra útreikninga í kóðanum þínum. Í stað þess að skrifa eigin reiknivél gætirðu viljað nota þriðja aðila pakka sem kallast simplecalculator. Til að gera þetta muntu nota pakkastjóra Thonny. Pakkastjórinn mun leyfa þér að setja upp pakka sem þú þarft að nota með forritinu þínu. Nánar tiltekið gerir það þér kleift að bæta fleiri verkfærum við verkfærakistuna þína. Thonny hefur þann innbyggða ávinning að takast á við hvers kyns árekstra við aðra Python-túlka. Til að fá aðgang að pakkastjóranum, farðu á valmyndastikuna og veldu Tools > Manage Packages... Þetta ætti að opna nýjan glugga með leitarreit. Sláðu inn simplecalculator í þann reit og smelltu á Leita-hnappinn.

Úttakið ætti að líta svipað út:



Mynd 5.5: Til að sækja "pakka"á netinu

Setja upp til að setja upp þennan pakka. Þú munt sjá lítinn glugga sem sýnir kerfið á meðan það setur upp pakkann. Þegar því er lokið ertu tilbúinn til að nota einfalda reiknivél í kóðanum þínum! Í næsta hluta muntu nota einfalda reiknivélarpakkann ásamt öðrum færni sem þú hefur lært í þessari kennslu til að búa til einfalt reiknivélarforrit.

6 Byrjum að Forritua

Nú erum við búin að fara yfir kynninguna á Python og setja upp tölvukerfið þannig að við getum byrja að forrita. Byrjum að forrita.

6.1 Halló heimur

Búum nú til forritið Halló Heimur. Það gerum við með því að afrita neðangreindan texta yfir í ritilinn þinn, og vista svo.

```
1    # Þetta er einfalt forrit
2    print("Halló heimur!")
```

Svo cd-um við okkur þangað sem skjalið var vistað og skrifum þar py, python eða python3, og svo nafnið á skjalinu. Að svo stöddu prentast Halló Heimur, á skjáinn.

6.2 Margar línur af kóða

Í dæmi 1 skrifuðum við Halló Heimur. En hvað ef við viljum prenta út fleiri línur? Jú, við getum bara búið til fleiri print-skipanir. Þá bara afritum við þessa print-línu, skellum aukaeintaki af henni svo niður fyrir hana, og voila, nú erum við komin með tveggja línu forrit. Þegar tölvan keyrir skriftirnar okkar byrjar hún alltaf efst og færir sig svo alltaf niður um eina línu eftir því sem hún klárar að keyra línurnar. Svipað því hvernig við mennirnir lesum.

6.3 Breitur

Í forritun er til hugtakið breyta. Breyta er bara hólf sem hægt er að setja gildi í. Breytur bera nafn með rentu, þær breytast. Gott dæmi um breytu væri t.a.m. breytan aldur. Aldur gæti þá táknað aldur einhverrar manneskju í árum. Hún gæti þá t.a.m haft gildið 33, eða 67 o.s.frv. Við getum skilgreint breytu og veitt henni gildi með eftirfarandi kóða.

```
aldur=33;
```

6.4 Breytur og þeirra tag (string og number

Breytur í Python geta verið annaðhvort strengir eða tölur. Í eftirfarandi kóða skilgreinum við tvær breytur, einn streng og eina tölu og prentum svo aðra þeirra út á skjáinn.

```
nafn='olafur';
aldur=33;
print(aldur);
Tag breytu
Til að koma að því af hvaða tagi breyta er, notum við type fallið,
    svona
tag_breytu=type(einhver_breyta);
print(tag_breytu);
```

6.5 Breytunafngift

Þegar við erum að gefa breytum gildi þurfum við að vera meðvituð um nokkur atriði.

6.6 Reglur

Breytunöfn eru gerð úr bókstöfum, tölustöfum og undirstrikum. Breytur mega ekki byrja á tölustaf. Ekki má nota millibil/spacebar í breytunöfnum. Mælt er með að nota undirstrik í staðinn. Python hefur frátekið nokkur orð eins og function, if og for. Til að forðast misskilning ber að varast að skýra breytur þessum nöfnum.

6.7 Tvíræðni

Sum tákn eru mjög lík og ber því að forðast að nota þau þar sem notandinn gæti haldið að um hitt táknið sé að ræða. Þegar við tjáum okkur í vísindaheiminum á sá sem við tölum við aldrei að geta komið til baka og sagt meinarðu a- eða b-útgáfuna. Allt sem við segjum á að vera ótvírætt.

l (bókstafurinn) og 1 (tölustafurinn). O (bókstafurinn) og 0 (tölustafurinn).

6.8 Stutt og skýr

Reyndu að hafa breytunöfnin stutt en skýr. Láttu þau lýsa eins vel og hægt er, hvað breytan stendur fyrir, í eins stuttu máli og hægt er.

6.9 Há - lág stafir

Breytunöfn geta verið með bæði há og lágstöfum. Mælt er með því að nota bara alltaf lágstafi í breytuheitum.

6.10 Athugasemdir

Stundum viljum við aðeins útskýra hvað er að gerast í kóðanum. Þá viljum við skrifa inn línu sem ekki er forritskóði. Þess háttar línur kallast athugasemdir. Það eru til tvær tegundri af athugasemdum í Python, einnar línu og margra lína. Eftirfarandi er dæmi um bæði.

```
#athugasemd 1.
#athugasemd 2.
#athugasemd 3.
"""
Þessar línur
keyrast ekki
sem kóði.
"""
```

6.11 Til hvers að skrifa athugasemdir

Tilgangurinn með því að skrifa athugasemdir í kóða er að útskýra í stuttu máli hvað er að gerast. Þá er mikilvægt að skrifa bara athugasemdir þar sem þeirra er þörf og að þær séu stuttar, hnitmiðaðar og svari líklegustu spurningunum sem lesandinn hefur varðandi kóðann. Það er mikil list að nota athugasemdir rétt og geta þær stórbætt lesanleikann ef þær eru notaðar rétt.

7 Tölur

Í Python, eins og í flestum C-afleiddum forritunarmálum, er til gagnatagið tala. Þegar breyta er af taginu tala er auðvelt að framkvæma tölfræðilega vinnslu á henni.

7.1 Strengur eða tala

Breytur eru alltaf af einhverju tagi. Þær geta m.a. haft tagið strengur eða tala. Hvernig aðgerðir framkvæmast á strengi er ólíkt því hvernig aðgerðir framkvæmast á tölur. Því þurfum við að vera meðvituð um af hvaða tagi viðkomandi breyta er hverju sinni. Að ruglast á tagi breytna og komast fram við t.a.m. strengi eins og þeir séu tölur er mjög algeng villa, bæði hjá byrjendum og líka reyndar hjá lengra komnum.

7.2 Dæmi um töluvinnslu

Til að láta forritið leggja breyturnar saman tölulega þá þurfum við að breyta þeim úr string yfir í int. Að láta breytur fara úr einu tagi yfir í annað kallast typecasting. Eftirfarandi dæmi sýnir hvernig við látum notandann slá inn tvær tölur, forritið grípur strengina sem input fallið skilar, typecast-ar viðkomandi strengjum yfir í tölu, framkvæmir viðkomandi aðgerð á þessar tölur og prentar niðurstöðuna á skjáinn.

```
1 innsleginn_strengur=input();
2 innsleginn_strengur=rstrip(innsleginn_strengur);
3 print(innsleginn_strengur);
```

Þar sem breyturnar eru núna orðnar að tölum þá getum við nú framkvæmt allar tölulegar aðgerðir á þeim, eins og t.a.m. +-* og /, með sama hætti.

7.3 Punktur, ekki komma, táknar kommu

Þar sem breyturnar eru núna orðnar að tölum þá getum við nú framkvæmt allar tölulegar aðgerðir á þeim, eins og t.a.m. +-* og /, með sama hætti.

7.4 Einfaldar reikniaðgerðir

Þegar breyta er af taginu tala getum við framkvæmt einfaldar reikniaðgerðir með henni. T.a.m. getum við gert.

```
price_one=10;
price_two=20;
total=(price_one+price_two);
print(total);
```

7.5 Int og Float

Í Python eru til tvenns konar tölur, int og float. Int er bara heilar tölur, eins og 1,2,3, 100 o.s.frv. Float er hins vegar, eins og nafnið gefur til kynna, fljótandi. Hún getur t.a.m. verið 1.231 eða 0.9811. Í eftirfarandi forriti tökum við inn streng frá notandanum, umbreytum honum í float og notum svo þessa tölu til að framkvæma útreikninga. Svo þegar við prentum niðurstöðuna á skjáinn þá pössum við að það sé með tveimur aukastöfum.

```
radius=input("Sláðu inn radíus hrings:");
radius=float(radius);
ummal = (2 * radius * 3.1415);
print("Ummál hringsins er: %.2f" %ummal);
```

Taktu eftir punktinum. Veldi Til að setja tölu í eitthvað veldi notum við veldistáknið **. 1. Hér verður temp jafnt og 2 í þriðja veldi. Eða 8. temp=2**3; 2. Hér verður temp jafnt og 2 í fjórða veldi, eða 16. temp=pow(2,4);

7.6 Hæsta og lægsta talan. (min og max)

Við getum fengið hæstu eða lægstu tölu sem send er inn í fall með min og max.

```
a = int(input("Sláðu inn tölu 1: "));
b = int(input("Sláðu inn tölu 2: "));
print("Lægri talan er " , min(a,b));
```

8 Strengir

Python hefur, eins og flest forritunarmál, svokölluð gagnatög. Eitt af stærri og mikilvægari gagnatögum sem við vinnum með er hinn svokallaði strengur. Strengur er í rauninni bara listi af stöfum sem er geymdur í breytu. Í þessum kafla munum við fara yfir helstu strengjaaðgerðirnar.

8.1 Dæmi um strengjavinnslu

Þegar notandinn slær inn streng í gegnum fallið input, þá er skilagildið sem input() skilar, af taginu string. Input-fallið er forritað með þeim hætti að það skilar alltaf streng. Þetta þurfum við að vera meðvituð því hvernig við leggjum saman strengi er ekki eins og hvernig við leggjum saman tölur. Skoðum eftirfarandi dæmi.

```
1 tala1 = input("Sláðu inn fyrri töluna: ");
2 tala2 = input("Sláðu inn seinni töluna: ");
3 badar=(tala1+tala2);
4 print("Summan er: ",badar);
```

Í ofangreindu forriti erum við að taka við inntaki frá notandanum og leggja það svo saman. En þar sem fallið input skilar gagnataginu streng þá erum við þarna að reyna að leggja saman tvo strengi, en ekki tölur. Þar sem strengjasamlagning virkar með þeim hætti að strengirnir koma bara hver á fætur öðrum, þá verður niðurstaðan þarna bara fyrri talan og svo kemur seinni talan beint á eftir.

8.2 Gæsalappir tákna streng

Skoðaðu Halló Heimur-forritið sem skrifað er hér fyrir neðan. Taktu eftir því að Halló Heimur er þarna innan í gæsalöppum. Í Python tákna gæsalappir byrjun strengs. Svo leitar forritið að næstu gæsalöppum sem það finnur og það verður þá endir strengsins.

```
1 print("Halló Heimur.");
```

8.3 Einföld og tvöföld komma

Í Python hafa einföld komma og tvöföld komma, sömu virkni. Ef við viljum skrifa streng sem hefur einfaldar kommur inni í sér, getum við skilgreint hann með tvöfaldri kommu, og notað svo einfalda kommu til að tákna kommurnar, og öfugt. Í eftirfarandi dæmi erum við að skilgreina streng sem inniheldur strenginn "Hallo Heimur". Taktu eftir því hvernig einföld og tvöföld komma geta tekið á sig hlutverk hinnar, eftir hentisemi.

```
kynning=("Hallo Heimur");
kynning="'Hallo Heimur'";
```

8.4 Sértákn innan í streng \n, \", \t

En ef við viljum prenta gæsalappirnar sjálfar á skjáinn þá megum við auðvitað ekki skella gæsalöppunum inn beint því þá ruglast Python og heldur að þessar gæsalappir tákni endinn á strengnum. Til að segja Python að tákn eigi ekki að vera túlkuð af Python þá setjum við skástrik fyrir framan það eins og eftirfarandi dæmi, sem prentar textann Bilbó innan í gæsalöppum, sýnir.

```
print("\"Bilbó\"");
```

Það eru fleiri tákn sem hafa merkingu í forritunarmálinu og þurfum við að setja þetta skástrik fyrir framan þau, eins og t.a.m. newline (\n) og tab (\t). Að nota skástrik með þessum hætti er hefð í forritunarheiminum. Mörg önnur tungumál virka svona líka.

8.5 Losna við auka endabil

Það kemur fyrir að maður er með streng, og annaðhvort fremst í strengnum eða aftast er eitt aukamillibil. Þetta aukamillibil getur gert það að verkum að strengurinn passar ekki inn í mynstur o.þ.h. En það getur verið lúmskt erfitt fyrir okkur mennina að greina að ekki sé um sama streng að ræða. Til að losna við þessa mögulegu villu er til fall sem heitir strip. Með strip þá fjarlægjast öll aukamillibil fremst og aftast í strengnum. ' python' myndi t.a.m. verða 'python'. Dæmi um keyrslu strip.

```
nafn_an_bila=nafn.strip();
```

8.6 Íslenskir Stafir

Til að geta prentað íslenska stafi í forritunum okkar þurfum við að hafa eftirfarandi línu efst í forritinu.

```
# -*- coding: UTF-8 -*-
```

8.7 Inntak

Til að leyfa notanda forrits að senda upplýsingar inn í það í miðri keyrslu notum við input-fallið. Það sem við sendum inn í input-fallið prentast á skjáinn. Input stöðvar svo keyrslu forritsins á meðan notandinn er að skrifa textann. Þegar notandinn ýtir svo á enter, færist það sem notandinn skrifaði inn í breytuna vinstra megin við input-fallið og forritið heldur svo áfram keyrslu.

```
innsleginn_texti = input("Sláðu inn einhvern texta");
```

8.8 Skeyta saman textastrengjum

Stundum erum við með tvo strengi og okkur langar að skeyta þeim saman, þ.e.a.s. við viljum að þessir tveir strengir verði lagðir saman til að mynda einn stærri streng. Til þess notum við plúsvirkjann (+), svona:

```
nafn = "Daniel";
simi= "5551234";
nafn_og_simi=nafn+" "+simi;
print(nafn_og_simi);
```

Þarna bjuggum við til tvær breytur og skelltum streng í þær báðar. Nafn í fyrri og símanúmer í seinni. Þess næst bjuggum við til þriðja strenginn, sem er bara fyrsti strengurinn, svo streng sem er ekkert nema eitt línubil og svo seinni strenginn. Að lokum prentuðum við þennan nýja streng út.

8.9 Plús virkinn (+)

Plúsvirkinn er notaður til að leggja hluti saman. Þetta verður hins vegar svolítið snúið því við erum með breytur sem geta verið annaðhvort strengur eða tala. Það hvernig tölur leggjast saman er augljóst. Þá leggur tölvan bara tölurnar saman og það verður niðurstaðan. Það hvernig við leggjum saman strengi er ekki alveg jafn augljóst en það sem gerist er að seinni strengurinn bætist bara aftan við þann fyrri. Hvernig við leggjum streng og tölu saman er sýnt hér fyrir neðan. Niðurstaðan, plús virkjans, skilast svo alltaf vinstra megin við sama merkið.

```
#prentum streng og tölu.
name="Oli";
age=33;
print(name,"%d" %(age));
```

8.10 Um Strengi

Strengir eru í rauninni bara listi af stöfum. Við getum fengið aðgang að einstökum stöfum eða stafamengi í þessum streng, þá bara segjum við strenginn og svo númer hvaða stak við viljum fá, innan í hornklofa. Í eftirfarandi dæmi prentum við fyrsta stafinn í strengnum á skjáinn.

```
texti="hallo";
fyrsti_stafurinn=texti[0];
print(fyrsti_stafurinn);
```

Taktu eftir því að fyrsti stafurinn er númer 0 en ekki 1. Það er því fyrsti stafurinn er númer 0 en ekki 1. Tölvur byrja yfirleitt að telja á 0. Að ruglast á því að byrja á 0 eða 1 er algeng byrjendavilla og kallast off-by-one villa á ensku.

8.11 Neikvætt index

Við getum svo prentað síðasta staf í streng með því að segja bara -1, í staðinn fyrir töluna. Í rauninni er hægt að fara niður í mínus tölur og ertu þar komin/n aftast í strenginn og ert að færa þig framar. T.a.m. er fimmti aftasti stafurinn í streng á indexi -5.

8.12 Stafasneið

Við getum svo fengið sneið af stöfum úr streng, þá segjum við bara fyrra númerið, svo kemur tvípunktur og svo seinna númerið. Í eftirfarandi dæmi erum við að sækja úr strengnum stafi númer 2, 3 og 4 í strengnum.

```
textinn="Góðan daginn öllsömul.";
sneid=texti[2:4];
print(sneid);
Ef við viljum fá síðastu þrjá stafina, gerum við svona:
textinn="Góðan daginn öllsömul.";
sneid=texti[-3:];
print(sneid);
```

Þarna erum við í rauninni að segja, frá staf -3, og út að enda. Í staðinn fyrir tvípunktinn þarna er gott að segja við sjálfan sig „til“. Og ef ekkert kemur eftir það, þá merkir það .. að enda þess sem við erum að tala um.

8.13 á og lágstafir (upper,lower)

Við getum skellt streng í há eða lág stafi með föllunum upper og lower. T.a.m. gætum við fengið afrit af strengnum nafn þar sem allir stafirnir eru orðnir hástafir, svona:

```
nafn="oLi";
nafn_i_hastofum = nafn.upper();
print(nafn);
```

Það er einnig hægt að fá fyrsta staf viðkomandi orðs í hástafi með því að nota fall sem kallast title().

8.14 Hástöfum fyrsta staf

Stundum viljum við að fyrsti stafurinn í hverju orði sé hástafur. Til þess er til fall sem heitir `title`. Réttara væri kannski að skíra það `uppercase_first` eða eitthvað í þá áttina.

Svona notum við `title`:

```
textinn='ja godan daginn';textinn_u=textinn.title();
print(textinn_u);
```

8.15 Lengd strengs

Við getum fengið lengd strengs / stafafjölda, svona.

```
stafafjoldi=len(strengur);
```

8.16 Snúa streng við, [::-1]

Við getum fengið lengd strengs / stafafjölda, svona.

```
strengurinn_a_hvolfi=einhver_strengur[::-1];
```

8.17 Teljum (count)

Ef við viljum telja hversu oft eitthvað tákn kemur fyrir í streng, notum við count().

```
textinn="hallo";
fjoldi_ella=textinn.count("l");
print(fjoldi_ella);
```

8.18 Skiptum út tákni (replace)

Við getum skipt út einu tákni fyrir annað í strengjum. Það gerum við svona.

```
1 nafn_breytt=nafn.replace("a","A",2);
```

Í línunni hér að ofan bjuggum við til streng sem er eins og annar strengur nema bara öll eintök af a verða a stóru A.

8.19 Skiptum út tákni mörgum sinnum

Við getum svo framkvæmt viðkomandi replace oft, en það gerum við með því að bæta við þriðju færibreytunni, hún táknar hversu oft við viljum framkvæma viðkomandi breytingu. T.a.m. ef við viljum framkvæma ákveðna útskiptingu tvisvar þá gerum við:

```
1 nafn_breytt=nafn.replace("a","A",);
```

Ef við viljum framkvæma viðkomandi útskiptingu bara yfir allt skjalið óháð því hversu mörg tilvikin eru, þá skellum við inn kommunni fyrir þriðju færibreytuna en tökum svo ekki fram neina tölu. Þá fattar Python að við viljum framkvæma þetta á öll tilfellin í strengnum.

8.20 Losnum við auka línubil

```
1 innsleginn_strengur=input();
```

9 Lykkjur

Stundum þurfum við að framkvæma einhverja aðgerð x oft. Það hversu oft viðkomandi framkvæmd verður framkvæmd er þá háð einhverjum breytum, eins og t.a.m. fjölda nemenda í bekk, fjölda punkta á mynd eða eitthvað í þá áttina. Kóðinn til að framkvæma aðgerð x oft kallast lykkja og eru til þrjár tegundir af þeim: while, do og for.

9.1 While

While er bara enska fyrir á-meðan. Með while-lúppu þá framkvæmum við það sem er í slaufusviganum, svo lengi sem eitthvað skilyrði gildir. T.a.m. getum við sagt 'Á meðan aldur er hærri en 30, framkvæmdu virkni x.' Og þá væri mjög sniðugt ef það er eitthvað í virkni x sem færir okkur nær því að while-lúppan endi. While-lykkja endurkeyrir ákveðna forritsblokk aftur og aftur á meðan ákveðið skilyrði er fyrir hendi. Þegar þetta skilyrði er ekki lengur satt þá hættir forritið að keyra while- lykkjuna og heldur áfram á næstu línu.

```
1 while(skilyrði):
2     forritsblokk
```

9.2 Eilífðarlykkjan

Stundum viljum við bara að lykkjan sé alltaf true sama hvað er að gerast í kóðanum. Þá þurfum við að skilja að það sem er hægra megin við while-orðið er í rauninni setning sem tölvan er alltaf að meta upp á hvort hún sé true eða false. Til að yfirskrifa það allt saman þá getum við bara sagt beint True og þá verður þetta while alltaf true og því heldur lykkjan bara að keyra aftur og aftur. Eftirfarandi er dæmi um eilífðarlykkju.

```
1 while True:
2     print("hi");
```

9.3 Do

Do er mjög sjaldan notuð. Hún er eins og while að öllu leyti nema að hún byrjar á að framkvæma eitt stykki ítrun áður en hún fer að skoða skilyrðin. Þá skrifar maður do, svo kemur blokkin og svo kemur while í lokin. Do gerir ekkert sem ekki er hægt að gera betur með for lykkjum.

9.4 For

Algengustu og mikilvægustu lykkjurnar, í Python sem og öllum öðrum málum sem erfa forritunarmálið C, er hin svokallaða for lykkja. Í öllum þessum forritunarmálum þá er það alltaf sama þemað með for lykkjuna. Það kemur alltaf orðið for og svo koma þrjár tölur, þær eru eftirfarandi.

9.5 Fyrsta talan

Fyrsta talan táknar þá upphafsstöðuna. Þ.e.a.s. einhver breyta fær eitthvað gildi í upphafi.

9.6 Mið talan

Næsta talan táknar svo svo-framarlega-sem. Þ.e.a.s. blokkin sem kemur á eftir, verður framkvæmd aftur og aftur, svo framarlega sem þetta skilyrði heldur.

9.7 Lokatalan

Að lokum kemur hvað eigi að gerast í lokin á hverri lúppu. Og það er alltaf eitthvað sem færir okkur nær því að loka lúppunni. Dæmigerð Python for lykkja lítur svona út:

```
1 for teljari in range(0,10,1):
2     foritsblokk;
```

Þarna erum við í rauninni að segja, að breytan teljari fær í upphafi gildið 0. Og svo framarlega sem teljari sé minni en 10, þá framkvæmum við eftirfarandi blokk, og hækkum svo teljarann alltaf um einn í lokin.

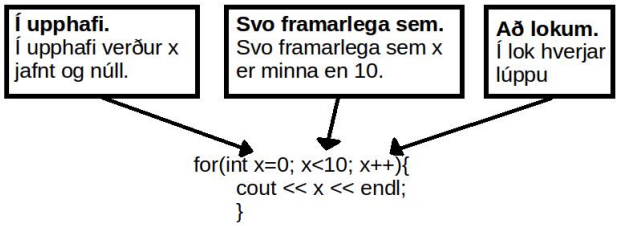
9.8 Sleppa aftasta skilyrðinu

Í Python er stundum öftustu tölunni sleppt, en þegar svo er þá stillist hún sjálfkrafa á 1. Það er því þessi tala er nánast alltaf sett á 1 hvort eð er og höfundum

málsins fannst það ofaukið að vera alltaf að taka fram sama hlutinn aftur og aftur. Þegar öftustu tölunni er sleppt, er þess vegna hægt að bæta henni bara aftur við og allt virkar eins. Ímyndaðu þér bara að hún sé þarna og hafi gildið 1.

9.9 For lykkjur breytast ekki mikið milli forritunarmála

Eftirfarandi útskýringarmynd er tekin úr forritunarmálinu C++, en Python sem og flest forritunarmál virka nánast eins hvað þetta varðar.



Mynd 9.1: h: Hermann

Þú getur auðvitað skellt inn breytum í staðinn fyrir þessar tölur. T.a.m. væri alveg hægt að notast við breytu í staðinn fyrir töluna tíu þarna í miðjunni. Þá væri virknin, svo framarlega sem breytan x er minni en þessi nýja breyta okkar.

9.10 Þversumma

Við reiknum þversummu tölu með því að taka alla einstakar tölur í talnarunu og leggja þær saman. Til dæmis þá er þversumman fyrir töluna 521, 5+2+1, eða 8.

9.11 Hreiðraðar for lykkjur

Stundum viljum við ítra í gegnum lista, og fyrir hvert stak ítrum við í gegnum hvert stak í einhverjum öðrum lista. Til þess þá skrifum við svokallaðar hreiðraðar lykkjur (nested loops). Þá skrifum við for lykkju en innan í blokkinni af henni sjálfri kemur svo önnur for lykkja. Skoðum neðangreint forrit.

```
1 for x in range(0,10):
2     for y in range(0,10):
3         print(y,end="");
4     print();
```

Þetta gæti verið lykkja sem er að ítra í gegnum hvern einasta punkt í mynd. Þá förum við í gegnum öll gildin í x, og fyrir hvert einasta stak í því förum við í gegnum hvert einasta stak í y. Í fyrstu línunni er lykkja sem virkar þannig að breytan x fær fyrst gildið 0, svo framkvæmist blokkin og að því loknu hækkar x um einn og blokkin framkvæmist aftur, þar til x er ekki lengur minna en 10. Innan í blokkinni erum við svo með y, sem í upphafi fær gildið 0, svo framkvæmir hún viðkomandi blokk og hækkar gildið sitt um 1, þar til y er ekki lengur minna en 10. Innst er svo það framkvæmist á hvern einasta punkt, en það er bara að prenta út gildið á y og passa að prenta ekki nýja línu eftir á. Taktu eftir aukaprint skipuninni þarna sem er einu indenti neðar. Sú skipun keyrist þegar búið er að fara í gegnum hvert einasta y og þar sem þetta er tóm print lína þá prentast ekkert nema ný lína með því.

9.12 Brjótast út úr lykkju (break)

Að hætta í lykkju, break. Við getum verið að keyra lykkju en svo ef eitthvað ákveðið ástand kemur upp að þá viljum við hætta að keyra lykkjuna. Til þess að hætta keyrslu lykkjunnar notum við svokallaða break-skipun. Eftirfarandi kóði er dæmi um notkun break.

```
while True:
    innsleginn_strengur = input("String to capitalize [q to quit]:")
    innsleginn_strengur=innsleginn_strengur.capitalize();
    if innsleginn_strengur == "Q":
        break;
    print(innsleginn_strengur);
```

9.13 Fara í næstu ítrun (continue)

Stundum viljum við ekki vinna með stöðuna sem komin er upp en við viljum heldur ekki hætta keyrslu. Við viljum bara hætta þessari ítrun af lykkjunni og fara beint í þá næstu. Það gerum við með continue.

```
while True:
    value = raw_input("Integer, please [q to quit]:")
    #quit
    if value == 'q':
        break;
# an even number
    number = int(value);
    if number % 2 == 0:
        continue;
    print(number, "squared is", number*number);
```

Kóðinn hér að ofan er eins og kóðinn áður nema í þetta skiptið þá vinnum við ekki með sléttar tölur. Ef þetta er slétt tala þá bara förum við beint yfir í næstu

ítrun. Prófaðu að setja þetta upp í tölvunni þinni og keyra. Svona getum við alltaf haft val um hvort við klárum viðkomandi ítrun eða ekki með continue.

Fall er mengi kóða sem bundinn er saman til að ná einni virkni. Þegar búið er að skilgreina ákveðna virkni sem fall að gefnu nafni er hægt að kalla á viðkomandi virkni með því að tilgreina nafn virkninnar og setja svo sviga á eftir. En svigi sem opnast og lokast (), segir yfirleitt að um fall sé að ræða. Við skilgreinum fall með því að skrifa def, og svo nafnið á viðkomandi falli og svo kemur kóðinn sem það inniheldur. Í sviganum efst í fallinu tökum við fram hverjar færibreyturnar eru í fallinu en færibreytur eru breytur sem eru sendar inn í fall. Færibreytur eru þá gögn sem fallið þarf til að framkvæma það. Föll hafa svo svokallað skilagildi, en það er gildi sem skilað er til baka úr fallinu. Eftirfarandi er fall sem leggur saman þær tvær tölur sem sendar eru inn til þess, og skilar niðurstöðunni til baka sem skilagildi.

```
def leggja_saman(tala1,tala2):
    samtals=(tala1 + tala2);
    return samtals;
#Hér köllum við svo á fallið.
nidurstada=leggja_saman(4,5);
#Taktu eftir því að við sendum 4 og 5 inn sem færibreytur,
#og niðurstöðunni er svo skilað í breytuna nidurstada.
#og að lokum prentum við niðurstöðuna á skjáinn.
print(nidurstada);
```

9.14 Nú getum við bara kallað á fallið

Eftir að ofangreind skilgreining á því hvernig tölur leggjast saman hefur verið skilgreind, þurfum hvorki við né nokkur annar að vera meðvituð/uð um það hvernig það gerist. Við getum bara kallað á þetta fall og fallið veit allt um það hvernig farið er að því að leggja tvær tölur saman. Að beita þessari tækni leysir gríðarleg vandamál og gerir það að verkum að við erum alltaf að endurnýta virknina sem við útfærum. Líka á milli tölvukerfa og verkefna.

9.15 Brjótum kerfið upp

Að skrifa föll bætir endurnýtingu kóðans þar sem við leysum vandamál bara einu sinni á einum stað og frá og með þeim degi getum við alltaf bara vísað í viðkomandi fall til að leysa viðkomandi verkefni. Líka eftir 3 ár þegar við verðum að vinna við eitthvað allt annað tölvukerfi. Maður vinnur nefnilega aldrei bara eitt verkefni. Til að leysa eitt verkefni þarf alltaf að leysa mörg minni og binda þau svo saman.

9.16 Innbyggð föll í Python

Við getum skrifað okkar eigin föll en svo getum við líka haft aðgang að föllum sem aðrir hafa skrifað fyrir okkur. Dæmi um föll sem við höfum sjálfkrafa aðgang að í Python.

- input()
- int()
- abs()
- min()
- sqrt()
- pow()
- len()
- count()
- swapcase()
- upper()
- lower()

9.17 Svigarnir

Þú getur getið þér til um hvað þessi föll gera. Á eftir hverju nafni kemur svigi, það þýðir að þetta er fall. Flest forritunarmál hafa þann sið að láta sviga tákna fall. Inn í svigann er svo hægt að setja breytur, þessar breytur eru þá sendar inn í viðkomandi fall og kallast þá færibreytur.

9.18 Nefndar færibreytur

Við erum búin að sjá hvernig hægt er að senda færibreytur inn í föll. Þar skiptir röðunin máli, þ.e.a.s., hvar í röðuninni færibreytan kemur, ákvarðar í hvaða breytur hún fer í fallkeyrslunni. En stundum viljum við bara segja að ákveðin breyta eigi að fá ákveðið gildi, óháð því hvar í röðun færibreytnanna hún er. Það gerum við svona.

```
1 def kynning(nafn,aldur):
2     print(nafn+", "+aldur+" ára.");
3 kynning(aldur=33,nafn="Jon");
```

Þarna búum við til fall með færibreytum og köllum á það þar sem við handvirkt tökum fram hvaða færibreyta á að fá hvaða gildi. Þar með skiptir röðunin ekki máli lengur. Að kalla á föll með þessum hætti leiðir til betri lesanleika og skrifanleika þar sem tekið er fram í kóðanum hvaða breyta er að fá hvaða gildi. Við þurfum því ekki lengur að vita hvaða færibreytunúmer táknar hvaða breytu.

9.19 Sjálfgefin gildi

Þarna búum við til fall með færibreytum og köllum á það þar sem við handvirkt tökum fram hvaða færibreyta á að fá hvaða gildi. Þar með skiptir röðunin ekki máli lengur. Að kalla á föll með þessum hætti leiðir til betri lesanleika og skrifanleika þar sem tekið er fram í kóðanum hvaða breyta er að fá hvaða gildi. Við þurfum því ekki lengur að vita hvaða færibreytunúmer táknar hvaða breytu.

```
def kasta_kvedju(name, kvedja='Godan daginn!'):
    print name,kvedja;
kasta_kvedju('Oli');
```

9.20 Forrit færibreitur

```
#Þessi lína hleður módúlunni sys,
#sem við þurfum til að taka við færibreytum.
import sys;
print('Þetta forrit fékk færibreyturnar:', sys.argv);
```

Nú getum við keyrt forritið svona.

```
python forritid.py tra la la
```

Og úttak forritsins myndi þá líta svona út. Þetta forrit fékk færibreyturnar: ["forrit.py", "tra", "la", "la"] Taktu eftir því að stak númer 0 er nafnið á forritinu.

9.21 Stjörnufræðingurinn

Þú ert að vinna sem stjörnufræðingur þar sem hinar ýmsu upplýsingar um hin og þessi sólkerfi eru í sífellu að koma inn. Þegar svo er er oft sniðugt að geta reiknað afleidd gögn, og ekki sakar þá að kunna smá Python.

10 Listar

Listi er röð af hólfum. Í hvert þessara hólfa er svo hægt að setja eitt stykki hlut af þínu vali. Þessi hlutur má vera af hvaða tagi sem er. Eftirfarandi er dæmi um lista af strengjum vikudaganna.

```
1 vikudagar = ['Mánudagur', 'Þriðjudagur', 'Miðvikudagur',
      'Fimmtudagur','Föstudagur'];
```

Til að vísa í fyrsta stakið í listanum hér þá getum við gert vikudagar[0]. Taktu eftir því þarna segjum við vikudagar og svo 0 í hornklofa. Það þýðir að þetta er núllta stakið í listanum. En listar byrja alltaf að telja á núll. Þegar við vísum í stök í lista með þessum hætti er oft sagt at, þegar maður er að fara að segja hvar viðkomandi stak er í listanum. Sem dæmi, vikudagar[3] væri þá sagt vikudagar-at-3.

10.1 Breyta einu staki

Neðangreind skipun breytir fyrsta stakinu í listanum yfir í streng sem lýsir þessu furðulega nafni á mánudegi.

```
1 vikudagar[0]='Mááánudagur';
```

10.2 x er bara breyta

Við getum unnið með stök í listum, alveg eins og um staka breytu væri að ræða. Við getum t.a.m. gert:

```
1 dagur_litlum_stofum = vikudagar[0].lower()
```

10.3 Fram fyrir núllið

Í listum þá getum við líka vísað í hluti afturábak. Þ.e.a.s. við getum talað um neikvæðar tölur í indexinu og þá byrjar kerfið aftast. T.a.m. getum við breytt síðasta vikudeginum í listanum hér að ofan með.

```
vikudagar[-1]='Fööööstudagur';
```

Til að ná svo í næstsíðasta stakið getum við gert: vikudagar[-2]; o.s.frv. þar til við erum komin í mínus sem er ígildi fjölda staka í listanum, en þá erum við komin aftur að upphafi hans.

10.4 Út fyrir svæði listans

Ef við vísum í tölu sem er hærri en lengd listans eða lægri en mínus lengd listans því þá fáum við neðangreinda villu. Villan segir okkur að index-ið sé fyrir utan leyfð svæði listans. Index out of range er algeng villa í listum og það er gott að hafa hana í huga við villuskoðun.

IndexError: list index out of range

10.5 Leggja saman tvo lista

Svona getum við lagt saman tvo lista og þá verður niðurstaðan bara einn stóri listi með öllum þessum stökum.

```
fuglar = litlir_fuglar + storir_fuglar;
```

10.6 Bæta í lista

-Innan í lista Með insert getum við bætt stökum við lista, hvar sem er. Við þurfum þá auðvitað að taka fram hvar við viljum að gildið komi.

```
nemendur = ['Oli', 'Kalli', 'Siggi'];
nemendur.insert(1, 'Jonatan');
['Oli', 'Jonatan', 'Kalli', 'Siggi']
```

10.7 Aftast í listann

Til að setja stak sjálfkrafa bara aftast í listann, notum við append.

```
nemendur.append('Anna');
['Oli', 'Jonatan', 'Kalli', 'Siggi', 'Anna']
```

10.8 Eyða úr lista

-Eyða eftir gildi (remove) Ef við vitum ekki hvar stakið er sem við viljum eyða þá er nóg að vita bara gildið.

```
nemendur = ['Siggi', 'Vigdis', 'Elin'];
nemendur.remove('Vigdis');
#['Siggi', 'Elin'];
```

10.9 Eyða eftir staðsetningu (pop)

Með pop getum við eytt stökum úr lista eftir staðsetningu innan listans. Taka ber fram að pop skilar því sem það eyddi til baka.

```
nemendur = ['Oli', 'Kalli', 'Siggi'];
Kalli = nemendur.pop(1);
nemendur er núna: ['Oli', 'Siggi'];
```

10.10 Hvar er þetta gildi í listanum (index)

Til að komast að því hvar eitthvað gildi er geymt í lista notum við index.

```
nemendur = ['Oli', 'Kalli', 'Siggi', 'Elin']
kalli_index = nemendur.index('Kalli');
```

10.11 Hversu oft kemur þetta stak fyrir í listanum (count)

Ef við viljum vita hversu oft eitthvað gildi kemur fyrir notum við count.

```
nemendur = ['Oli', 'Oli', 'Kalli', 'Siggi'];
nemendur.count('Siggi'); #1
nemendur.count('Vigdis'); #0
nemendur.count('Oli'); #2
```

10.12 Sneiðmengi

Stundum viljum við einungis gera eitthvað með hluta af lista. Þú kannski manst eftir því í strengjavinnslunni að þá gátum við fengið hlutstreng einmitt með því að tala við strenginn eins og hann væri listi, þá með t.a.m. [0:3] til að fá fyrstu þrjá stafina í honum. Strengir eru nefnilega undir húddinu, bara listar af stöfum. Innan í hornklofanum táknar fyrri talan frá og svo seinni talan til. Svona náum við í hlutstreng úr lista.

```
fuglar = ['Dúfa', 'Hæna', 'Geirfugl', 'Örn', 'Fálki'];
```

Ef við viljum ná bara í fyrstu 3 stökin:

```
fyrstu_thrir = fuglar[0:3];
```

Svo ef við viljum fá 3 stök, byrjandi á öðru staki.

```
sneidmengi = fuglar[1:4];
```

10.13 Sneiðmengi með for lykkjum

Strengjasneiðmengi í Python hafa virkni sem endurnýtir skilninginn sem við fengum á því að læra for lykkjur. Það eru þrjár tölur, og alveg eins og í for lykkjum þá táknar fyrsta talan, upphafsgildið. Miðjutalan táknar svo á hvaða staki við endum og síðasta talan er hversu mikið við breytum tölunni milli hverrar ítrunar. T.a.m. myndi eftirfarandi kóði skila til baka, öðru hverju staki í listanum, byrjandi í staki númer 3.

```
sneidmengi = nemendur[3:1:2];
```

10.14 Þegar tölunni er sleppt, er talan 1

Ofangreint dæmi hefði allt eins geta verið skrifað án þess að taka fram miðjutöluna. Málið er að í Python er kemur talan 1 svo oft fyrir að við getum sleppt henni og þá gefur kerfið sér að viðkomandi tala sé 1. T.a.m. hafa eftirfarandi tvær línur sömu merkinguna.

```
sneidmengi = nemendur[3:1:2];
sneidmengi = nemendur[3::2];
```

10.15 Getum eitt í einu

Þegar við viljum taka ákveðna sneið úr lista og ítra svo í gegnum niðurstöðuna, þá er hægt að gera bæði samtímis með Python. Þ.e.a.s. búa til sneiðina og ítra í gegnum hana. Við mælum ekki með því að forrita þannig því þarna er verið að gera tvennt samtímis, sem hækkar flækjustigið. En lágt flækjustig er helsta einkenni hágæða hugbúnaðar.

10.16 Tölulega

Þegar við röðum tölulegum lista í Python er um tvær leiðir að ræða, að raða listanum beint (sort) og búa til auka eintak af listanum (sorted). Sorted er þá fall sem skilar til baka öðru eintaki af listanum sem við getum svo unnið með aftur.

10.17 Tölulega hækkandi

numbers = [2, 1, 4, 3]; numbers.sort(); [1, 2, 3, 4];

```
numbers = [2, 1, 4, 3];
numbers.sort();
[1, 2, 3, 4];
```

10.18 Tölulega lækkandi

Til að raða lista lækkandi gerum við svona.

```
numbers = [2, 1, 4, 3];
numbers.sort(reverse=True);
[4, 3, 2, 1];
```

10.19 Röðun sem auka eintak (sorted)

Svo ef við viljum fá röðunarniðurstöðuna sem auka eintak af listanum þá grípum við bara niðurstöðuna með sorted, svona:

```
numbers_sorted=sorted(numbers);
```

10.20 Í stafrófsröð

```
nemendur=['Oli', 'Kalli', 'Siggi'];
nemendur=sorted(nemendur);
#['Kalli', 'Oli', 'Siggi'];
```

10.21 Í öfugri stafrórfsröð

```
nemendur=['Harpa', 'Oli', 'Kalli'];
nemendur=sorted(nemendur,reverse=True);
#['Oli', 'Kalli', 'Harpa'];
```

10.22 Snúa listanum við

Til að snúa lista á hvolf notum við reverse.

```
listi.reverse();
```

10.23 Lengd lista

Til að fá hversu mörg stök eru í viðkomandi lista notum við len().

```
student_quantity=len(student_names);
```

10.24 Um leit í listum

Ef við erum með lista yfir kattarnöfn og við viljum finna kött að nafni Kalli í listanum, þá þurfum við að ítra í gegnum listann þar til við rekumst á viðkomandi nafn. Þetta er ekki mikið vandamál á meðan listarnir eru litlir en segjum að við séum með lista yfir 1 milljón ketti og viljum nú vísa í einhvern ákveðinn kött, þá er þetta orðið ansi tímafrekt. Við getum í raun vænst þess að þurfa að ítra í gegnum hálfan listann því stundum finnum við viðkomandi stak mjög snemma og stundum seint en að jafnaði er þetta um hálfur listinn.

10.25 Off-by-one villur

Það að listar byrji á núll en við mennirnir byrjum alltaf að telja á einum er hálf óþægilegt til að byrja með fyrir okkur mennina og eru villur sem tengjast þessum ruglingi kallaðar off-by-one villur. Þær eru algengar og oft er gott að hafa þennan möguleika í huga þegar við erum að skoða tölvukerfi. Vandamálið gæti verið off-by-one villa.

10.26 Tög Stakanna

Í Javascript og Python mega tög í listum vera hvernig sem er. Við getum haft streng í indexi 0, svo tölu í indexi 1 o.s.frv. Í C++ verða hins vegar öll stök í lista að vera af sama tagi, sem þarf að vera tekið fram við skilgreiningu listans. Það er frekar bindandi. Eftirfarandi er dæmi um hvernig hægt er að blanda saman staktögum í lista í Python.

```
fuglar = [3, 'Dúfa', 2, 'Hrafn'];
```

10.27 Ítra í gegnum lista

Við getum ítrað í gegnum öll stök í lista með:

```
for element in some_list:
```

En við getum einnig notað `for` með `range`, eins og lýst var í lykkjukaflanum. Þá þurfum við fyrst að vita hversu langur listinn er, sem við getum gert með `len()`, eins og einnig var sýnt hér að ofan.

```
fjoldi_staka=len(einhver_listi);
for stak_numer in range(0,fjoldi_staka,1):
    forritsblokk;
```

10.28 Spássían

Í Python þarf spássían/indent alltaf að vera rétt. Þegar við erum innan í for blokk, þá þarf að indenta einu sinni fyrir það. Við indentum með því að ýta á tab. Stundum breyta ritlar tab sjálfkrafa yfir í millibil, sem lítur þá alveg eins. Það getur leitt til villu sem erfitt er að greina. Lausnin er þá að þurrka út þessi millibil og skella einu tab í staðin.

10.29 Tölfræði

Stundum viljum við framkvæma tölfræðigreiningu á röð af tölum. Til eru föll sem leyfa okkur að framkvæma tölfræðigreininguna á listum. Sendum við þá listann sem færibreytu inn í föllin, og fáum til baka svarið. Eftirfarandi eru dæmi um þess háttar föll.

```
min();
max();
sum();
```

10.30 Að búa til nýja tilvísun í lista

Við erum ekki enn farin að læra hlutbundna forritun en þurfum pínkulítið að fara fram úr okkur hér þar sem listar eru hlutir. Ef við reynum að taka afrit af lista með neðangreindu þá erum við í rauninni að taka afrit af hlut. En hlutaafritun virkar alltaf í grunninn þannig að einungis tilvísun í viðkomandi hlut verður afrituð. Því er ofangreindur kóði einungis að búa til tvær tilvísanir í sama hlutinn, en ekki tvo hluti sem slíka.

```
listia=listib;
```

10.31 Að smíða nýjan lista úr öðrum

En hvernig afritum við þá lista? Jú, með listasneiðum eins og neðangreint dæmi sýnir.

```
listia=listib[:];
```

10.32 Óbreytanlegir listar (tuple)

Við getum búið til lista og með þeim getum við breytt öllum gildunum sem eru í listanum en stundum viljum við búa til lista sem er óbreytanlegur. Þess háttar listi kallast tuple í Python og er skilgreindur með því að nota sviga. Ef einhver reynir svo að breyta gildunum í tuple kemur bara villa. Það er þó hægt að endurskilgreina allt tuple-ið í einu og það virkar bara. Að öðru leyti virka tuple eins og listar. Eftirfarandi lína skilgreinir eitt stykki tuple með tveimur gildum.

```
faedingarar=(1977,1988);
```

10.33 Um sviga fyrir óbreytanleika lista

Sennilega hefði verið betra að gera lista óbreytanlega með því að bæta orðinu constant fyrir framan við skilgreiningu þeirra. Það væri skýrara og ekki að sjá að neinu væri tapað. Í flestum forritunarmálum tákna svigar að um fall sé að ræða og því virðist það ruglingslegt og það að ósekju að láta sviga tákna óbreytanleika í Python-listum.

10.34 Einkvæmur listi (set)

Í Python er til gagnatag sem heitir set en set er bara listi þar sem hvert gildi má bara koma fyrir einu sinni. Við förum ekki í þá að sinni.

10.35 Listar af listum

Eins og fram hefur komið eru listar röð af stökum. Þessi stök geta svo verið af hvaða tagi sem er, tala, strengur eða hlutur. Listar eru reyndar sjálfir hlutir, sem þýðir að í staki í lista, getur verið annar listi. Þannig að við getum búið til lista þar sem stökin eru svo aftur sjálf listar.

Reyndar geta stökin í þeim listum svo verið afturlistar o.s.frv. en listar þar sem stökin eru svo afturlistar kallast tvívíðir listar. Þess háttar listi er í rauninni bara eins og tafla. Eftirfarandi er dæmi um hvernig hægt er að búa til lista sem hefur stök sem sjálf vísa í annan lista.

```
litlir_fuglar = ['Dúfa', 'Hæna']
midlungs_fuglar = ['Hrafn']
storir_fuglar = ['Örn', 'Fálki']
allir_fuglar = [litlir_fuglar, midlungs_fuglar, storir_fuglar]
```

Nú getum við vísað í listann `litlir_fuglar` með:

```
allir_fuglar[0]
```

Til að vísa t.d. í „Hænu“, sem er annað stakið (í sæti númer 1) í litlu fuglunum — (munið að við byrjum að telja frá 0 í listum) — þá gerum við:

```
allir_fuglar[0][1]
```

Nú ætlum við að breyta „Hænu“ í „hrossagauk“:

```
allir_fuglar[0][1] = 'hrossagaukur'
```

Svona getum við haft lista innan í listum. Þeir geta verið af ótakmarkaðri dýpt. Það þarf þó alltaf að passa upp á flækjustigið — það getur verið ruglingslegt að tala um eitthvað sem er af dýpt 3 eða meira. Við mennirnir eigum frekar erfitt með að skilja margvíða lista, og því er ágætis pæling að reyna að fara ekki dýpra en tvær víddir.

11 Dictionary (Orðabækur)

Dictionary, eða orðabók, er gagnagerð sem geymir stök sem pör af lykli og gildi. Við getum notað lykilinn til að nálgast viðkomandi gildi með sóknartíma 1, óháð því hversu stórt dictionary-ið er. Við þurfum þó alltaf að vita lykilinn til að sækja gildið.

Eftirfarandi kóði býr til dictionary sem heldur utan um aldur katta, þar sem nafnið á hverjum ketti er lykill:

Nú getum við sótt aldur ákveðins kattar með því að nota nafnið sem lykil.

11.1 Um leit í dictionary

Til að finna hluti í dictionary þurfum við bara að vera með lykilinn að því sem við leitum að. Þar með verður leitartíminn 1 — óháður stærð dictionary-sins. Leit í listum eykst hins vegar í beinu hlutfalli við stærð listans, þar sem leita þarf í gegnum öll stök til að finna viðkomandi.

Þessi stöðugi sóknartími er mikilvægur ábati sem verður æ mikilvægari eftir því sem gagnamengið stækkar.

Galli við dictionary er að við getum ekki ítrað í gegnum það í fyrirfram ákveðinni röð. Einnig taka dictionary upp meira minni en listar sem geyma sama magn af gögnum. Ástæðan fyrir þessu er tæknileg og verður ekki útskýrð nánar hér.

11.2 Bæta við

Nú er kominn nýr köttur, hann heitir Óli og er 3 ára. Til að bæta honum við dictionary-ið gerum við.

11.3 Eyða

Til að fjarlægja eintak úr dictionary notum við del skipunina.

11.4 Upphafstilla

Að upphafsstilla dictionary og tæma það, er sama skipunin. Hún er eftirfarandi.

```
kettir = {};
```

11.5 Ná í alla lykla

```
allir_lykjar = kettir.keys();
```

11.6 Ná í öll gildi

```
oll_gildi = kettir.values();
```

11.7 Öryggt get

Nú mætir einhver á skrifstofuna til okkar, og segist vilja ná í köttinn sinn Ellu. Þá þurfum við að kanna hvort kötturinn Ella sé til í dictionaryinu okkar. Ef við keyrum þá

```
current_cat = kettir["Ella"];
```

Þá munum við fá villu því þessi lykill er ekki til. Til að forðast það að fá villu þegar maður leitar eftir lyklum sem eru ekki til getum við notað get fallið.

```
current_cat = kettir.get("Ella");
```

Og þá skilar fallið bara strengnum "None"í staðinn fyrir að stöðva keyrslu forritsins. Eftir þetta getum við kannað hvort einhverju hafi verið skilað með

```
if current_cat:
    print("Já, hann er hér.")
if not current_cat:
    print("Því miður, við höfum ekki þann kött.");
```

11.8 Íta í gegnum dictionary

Við þurfum oft að ítra í gegnum öll gildin í dictionary. Það ber að taka fram að ekki er í boði nein auðveld leið til að ítra í gegnum dictionary í fyrir fram ákveðinni röð. Til þess þurfum við að nota annaðhvort lista eða dictionary þar sem hvert gildi vísar sjálft í þann lykil sem kemur næst. Svona ítrum við í gegnum dictionary og prentum út öll gildin.

```
for ckey in some_dictionary:
    cvalue=some_dictionary[ckey];
    print(cvalue);
```

Áður en þú byrjar á næstu verkefnum þarftu að búa til eitt lítið orðabókardictionary með a.m.k. 10 lyklum/þýðingum.

11.9 Að íta í gegnum dictionary í ákveðinni röð

Það er eðli dictionary, að vera ekki raðað upp í ákveðinni röð. Því er það mótsögn að ætlast til þess að fara í gegnum það í fyrir fram ákveðinni röðun. Python býður þó upp á leið til að ítra í gegnum dictionary-ið, með því að breyta því í lista, raða listanum, og ítra svo í gegnum listann allt í einu. En það er ruglingslegt og ekki góð leið til að forrita. Betra væri að búa til lista úr dictionary-inu, raða honum upp og ítra í gegnum hann eins og það sem hann er þá orðinn.. listi.

```
#skilgreinum dictionaryið
favorite_languages={
    'jen':'python',
    'sarah':'c',
    'edward':'perl',
    'phil':'cpp'
    };
#búum til raðaðan lista úr dictionaryinu.
names_sorted=sorted(favorite_languages.keys());
#Ítrum í gegnum þennan raðaða lista
for cname in names_sorted:
    print(cname);
```

11.10 Dicitonary af...

Eins og við höfum séð þá getur dictionary innihaldið strengi og tölur en dictionary getur einnig innihaldið hluti eins og lista og... önnur dictionary. Við getum búið til dictionary þar sem hvert gildi er aftur dictionary. Það sama á við um lista.

Við getum búið til lista þar sem hvert stak er svo aftur listi. Við getum ítrað í gegnum dictionary af listum og fyrir hvert stak dictionary-sins, ítrað í gegnum viðkomandi lista. Þetta verður ansi fljótt ansi flókið.

Við munum því taka því rólega í bili að vaða út í þetta. Fyrst er gott að öðlast góðan skilning á hlutbundinni forritun, sem við förum yfir seinna í þessu námskeiði, en það einfaldar alla svona vinnslu gríðarlega.

12 Skjöl

Nú erum við að fara að skrifa forritskóða sem getur átt samskipti við skjöl. Þar sem Unix lítur á allt sem skjal eru samskipti við þau ekki bara leið til að geyma upplýsingar í minni heldur einnig til að eiga samskipti við aðrar tölvur annars staðar í heiminum í gegnum svokölluð websockets, sem og tala við jaðartækin. Í þessu námskeiði munum við bara líta á skjöl sem leið til að geyma gögn. En það er gott að vera meðvituð um það að þetta er grunnurinn að miklu meiri fræðum, og mikilvægt sem slíkt. Í tölvum er til tvenns konar minni, varanlegt minni og innra minni.

12.1 Varanlegt Minni

Varanlegt minni er minni sem virkar þannig að það varðveitir gögnin sín jafnvel þó að það missi samband við rafmagn. Til þessa þarf að geyma gögnin í einhverju öðru en rafmagninu sjálfu. En þar sem rafeindir ferðast mjög hratt miðað við efnislega hluti þá er varanlegt minni hægvirkara en innra minni. Varanlegt minni ræður þó yfirleitt við meira gagnamagn. Dæmi um varanlegt minni er t.a.m. harðir diskar, SSD-drif, USB-kubbar o.s.frv.

Hvaða form varanlegt minni tekur á sig er alltaf að þróast. Um 1980 voru það kassettur, svo 5,25"mjúkir diskar, svo 3,5"plastdiskar (floppy disks), geisladiskar, harðir diskar, USB kubbar, DVD diskar og þar fram eftir götunum.

Einingar varanlega minnisins kallast skjöl, skjölin eru svo sjálft bara tilvísanir í svokallaðar nóður. Við förum ekki út í nóður í þessu námskeiði, við lítum bara á skjölin sem grunneiningarnar í kerfinu. Í Unix (sem er allt nema Windows) er allt skjal. Möppur eru skjöl, skjöl eru skjöl og meira að segja jaðartækin sjálf eru skjöl.

12.2 Innra minni

Innra minnið er tölvukubbur í tölvunni sem kallast RAM (Ready Access Memory). Í hvert skipti sem rafmagnið fer af, tæmist þetta minni. Þegar við erum að ræsa tölvu þá er eitt stærsta verkefnið fyrir tölvuna að afrita þessi forrit sem við viljum hafa í gangi, úr varanlega minninu yfir í innra minnið.

Það eru til mismunandi tegundir af innra minni, mismunandi vinnsluhraði, sóknarhraði o.þ.h. Innra minni er hins vegar einsleitara en varanlegt minni að því leytinu til að þetta er allt bara tölvukubbar sem maður sér yfirleitt ekki.

Einingar innra minnisins eru svokallaðar minnisaddressur. Ef þú reynir að prenta tilvik af klasa, prentast yfirleitt bara innra minnis addressan sem viðkomandi tilvik er geymt í akkúrat þá. Hafðu það í huga því þú munt vinna mikið með tilvik af klösum og þá þarftu að þekkja hvernig þau haga sér og líta út við útprentun. Útfærsluatriði innra minnisins eru yfirleitt falin notandanum og þurfum við því ekki að vita mikið um það að svo stöddu.

12.3 Að keyra forrit (varalega → innra minni)

Þegar við keyrum forrit þá er forritið í upphafi einungis geymt í varanlega minninu. Tölvan þarf því að afrita forritið úr varanlega minninu yfir í innra minnið. Í innra minninu er svo forritið keyrt í gegnum örgjörvann, köllum við það þá ferli, process eða segjum að það sé í *keyrslu*. Hér á eftir koma helstu Python aðgerðir með skjöl.

12.4 Opna skjal

Neðangreindur Python-kóði býr til tengil við skjal. Skjalbreytan verður að Python-hlut sem sér um aðgengi að skjalinu sjálfu. Taktu eftir annarri færibreytunni sem send er inn í open(), en hún ákvarðar um hvers konar tengingu við skjalið er að ræða. Tengingarnar eru útskýrðar hér að neðan.

```
1 skjal=open('skjal.txt', 'w');
2 r=read
3 w=write
4 a=append (bætir við skjalið)
5 x=öruggt write (skrifar bara ef skjalið er ekki þegar til)
```

12.5 Lesa Skjal(r)

Neðangreindur kóði opnar skjal og les innihald þess inn í breytuna skjal_texti.

```
1 skjal=open('skjal.txt','r');
2 skjal_texti=skjal.read();
```

12.6 Skrifa í skjal (`w`)

Neðangreindur kóði opnar tengil í skjal og notar hann til að skrifa texta í skjalið. Taktu eftir að `write()` fallið bætir ekki við línuskilum sjálfkrafa. Þess vegna þurfum við að bæta `"\n"` aftast við það sem við viljum skrifa.

```
skjal=open('skjal.txt','w');
skjal.write("einhver texti\n");
```

12.7 Loka skjali

Þegar þú ert búin/n að nota skjalahlutinn geturðu lokað honum með.

```
skjal.close();
```

12.8 Varðandi samskipti innra minnis og varanlegs

Þegar við skrifum í skjal með write er látið eins og þetta sé að gerast í rauntíma. Eins og yfirleitt þá er raunveruleikinn aðeins flóknari. Það sem raunverulega gerist er að tölvan skráir viðkomandi gögn á lista yfir hluti sem hún skrifar í varanlegt minni í næstu skrifatrennu. Næsta skrifatrenna mun svo eiga sér stað þegar skjaltenglinum er lokað eða þegar forritið hættir keyrslu. Ef við viljum láta skjal_hlut skrifa án tafar allt sem hann hefur geymt notum við svokallað *flush()*, svona.

```
skjal.flush();
```

Þegar við búum til hlut í forritskóða sem tengir yfir í skjal í varanlega minninu, þá verðum við að passa okkur. Þessi forritsbútur okkar er í innra minninu og þar gerist allt á gríðarlegum hraða miðað við varanlega minnið. Innra minnið getur rústað varanlega minninu ef það passar sig ekki. Segjum að þú keyrir lykkju sem skrifar í skjal og lokar tenglinum í hverri ítrun. Þarna er kominn kóði sem skrifar í varanlega minnið á gríðarhraða og pínir það til að gera þetta jafnóðum. Ef varanlega minnið er t.a.m. harður diskur mun nálin á disknum þurfa að fara á nýja staðsetningu fyrir hverja einustu ítrun. Þetta mun þýða að nálin mun vera alveg á fullu að færa sig, skrifa og svo færa sig aftur. Þú getur eyðilagt harða diskinn með þessu.

Buffer-virknin, að skrifa ekki strax í skjalið heldur hafa ákveðnar skrifatrennur, er til að minnka hættu á m.a. þessu. Það eru til dæmi þar sem tíminn sem tekur að keyra ákveðið forrit hefur farið úr 10 mínútum niður í 10 sekúndur bara með því að buffera skrifin, svo ekki sé nú talað um álagið á harða diskinn.

12.9 Ein lína í einu

Ein lína í einu Við getum lesið línur skjals inn í lista með eftirfarandi kóða.

```
skjal=open('skjal.txt','r');
skjal_listi=skjal.read().splitlines();
print skjal_listi[0];
```

12.10 Að lesa x marga bita (read number)

Sjálfgefin hegðun read() er að lesa bara inn allt skjalið. Ef þú vilt les að ákveðnu magni geturðu notað eftirfarandi viðmót.

```
skjal=open('skjal.txt','r');
skjal_bitar=skjal.read(12);
print skjal_bitar;
```

12.11 Að lesa línu í einu

Við getum líka lesið eina línu í einu úr skjali með eftirfarandi kóða.

```
skjal=open('skjal.txt','r');
skjal_lina=skjal.readline();
print skjal_lina;
```

12.12 Skjalstígar

Stígarnir í Unix eru mjög einfaldir. Það er ekki flókið að ákvarða hvort stígur sé frá rótarkerfi eða hvort það sé frá undirmöppu. Ef þú ert að vinna með innri verkefni þá geturðu líka unnið með kóða sem notar skráarstíg fyrir kerfi sem ekki er Linux (t.d. Windows). Ef þú ert að breyta kóða fyrir annað kerfi en Linux, þá geturðu notað mát sem virkar í báðum kerfum.

```
import os
os.path.join('mappa', 'skjal.txt');
```

13 Klassar

Við búum í heimi sem er gerður úr hlutum, eða þannig hugsum við a.m.k. Þessir hlutir geta svo verið af mismunandi tegundum. Það geta verið menn, bílar, hús, stólar o.fl.

Allir hlutir eru af einhverju tagi. Af hvaða tagi hlutur er, ákvarðar hvernig hann er og hagar sér í heiminum. Sem dæmi þá er einhver einn ákveðinn maður, tilvik af hugtakinu *maður*. Um leið og við vitum að einhver hlutur er maður þá vitum við margt um viðkomandi hlut. Við vitum að viðkomandi hefur 2 augu, 2 eyru, nef, munn, höfuð og þar fram eftir götunum. Það sama á við um alla aðra hluti, þeir eru allir tilvik af klösum sem ákvarða eigindi þeirra. Af hvaða tagi hlutur er, ákvarðar hvernig hann er og hagar sér í heiminum.

13.1 Dæmi um bíla

Tökum bíla sem dæmi. Það er allt úti í bílum í þessum heimi. Ef við ætlum að forrita heim þá förum við ekkert að forrita hvern einasta bíl í þessum heimi okkar. Það væri óskilvirkt og við værum alltaf að skrifa sama kóðann aftur og aftur fyrir hvern bíl. Það sem við gerum í staðinn er að skilgreina bara í eitt skipti fyrir öll hvað það þýðir að vera bíll. Svo búum við til tilvik af bílum og þá taka viðkomandi einingar á sig öll þau gögn og alla þá virkni sem bílar almennt hafa.

Í venjulegu máli eru þetta hugtök og eintök. Í tölvunarfræðinni köllum við hugtökin klasa, og eintökin tilvik. Við skrifum þá klasa og búum að því loknu til tilvik af viðkomandi klasa. Yfirleitt er sér skjal fyrir hvern klasa. Að hugsa svona kallast að hugsa hlutbundið og er oft skynsamleg leið að þróa hugbúnað. Ábatinn af því að hugsa hlutbundið verður almennt meiri eftir því sem tölvukerfin verða stærri.

13.2 Member

Hlutir hafa skilgreinda svokallaðar membera. En það eru breytur/föll sem eru hangandi á öllum tilvikum viðkomandi hugtaks.

13.3 Berytur

Member breyta gæti t.a.m. verið bílnúmer á bílum. Bílnúmer myndi þá vera member breyta í klasanum *bíll*. Og væri eflaust strengur þegar út í það er farið.

13.4 Föll

Klasar hafa þá líka member föll. Í flestum forritunarmálum erum föll tilgreind með sviga, svona: `()`. Það er því við sendum stundum færibreytur inn í föllin, og fara þær þá inn í svigann. Eitt slíkt í klasanum/hugtakinu *bílar* gæti t.a.m. verið:

```
skrúfa_niður_rúðu().
```

Allir bílar gætu þá kallað á fallið með því að segja bara nafnið sitt og svo viðkomandi fall, svona:

```
raudi_billinn.skrufa_nidur_rudu();
```

Klasinn *bíll* gæti t.a.m. haft eftirfarandi member breytur og föll.
Breytur:

```
    hjól1
    hjól2
    hjól3
    hjól4
    mótor
    stýri
    bílnúmer
    staðsetning x
    staðsetning y
```

Föllin:

```
    opna_hurð()
    keyra_rúðuþurrkur()
    starta()
    skrúfa_niður_rúðu()
```

Við myndum svo útfæra öll þessi member föll í klasaskjalinu.

13.5 Kóðin

Eftirfarandi kóði skilgreinir hugtakið/klasann *bíll*, býr til tvö eintök af bílum, setur bílnúmer á báða bílana og prentar svo út bílnúmer rauða bílsins.

```
# -*- coding: UTF-8 -*-
#leyfir okkur að nota íslenska stafi.
#hér skilgreinum við hugtakið bíll.
class bill():
    #hér skilgreinum við aðgerðir sem hægt er að framkvæma á alla bíla.
    #self er hér tilvísun í það eintak af klasanum sem er að kalla í þetta fall
    hverju sinni.
    #hér er member fall bíls, sem gerir ekkert annað
    #en að setja gildi í member breytu bílsins.
    def setja_bilnumer(self,bilnumer):
        self.bilnumer=bilnumer;
#búum til tvö eintök af bílum.
raudi_billinn = bill();
blai_billinn = bill();
#Setjum bílnúmer á bílana okkar.
raudi_billinn.setja_bilnumer('p1736');
blai_billinn.setja_bilnumer('g4822');
#færibreyturnar hér 'p1736' og 'g4822' fara í memberbreytuna self.bilnumer.
#eins og sést í setja_bilnumer(), sem við erum að kalla á.
#það þýðir að þessi tilvik hafa núna viðkomandi gildi í breytunni sinni, bilnumer.
#prentum út númerið á rauða bílnum.
print("Rauði bíllinn hefur númerið: "+raudi_billinn.bilnumer);
```

13.6 Smiðir

Í klösum er hægt að tilgreina svokallaða smiði en smiður er forritsbútur sem smíðar ný eintök af viðkomandi hugtaki.

```
# -*- coding: UTF-8 -*-#leyfir okkur að nota íslenska stafi.

class Person:
    #Í Python er smiðurinn alltaf __init__().
    #Ef starf er ekki tilgreint verður það sjálfkrafa stillt á "none".
    #Kaup verður svo still á 0 ef því er ekki gefið gildi.
    def __init__(self, nafn, starf=None, kaup=0):
        #hér hengjum við færibreyturnar á nýja tilvikið.
        self.nafn=nafn;
        self.starf=starf;
        self.kaup=kaup;

#búum til persónurnar Jón og Úla.
#hér erum við að kalla á init fallið til að búa þær til.
```

```
jon=Person('Jon Jonsson');
oli=Person('Oli Sigmundsson', starf='dev', kaup=100000);

#prentum nafn og kaup þeirra.
print("Starfsmadur1: "+jon.nafn, jon.kaup);
print("Starfsmadur2: "+oli.nafn, oli.kaup);
```

13.7 Erfir

Eðli hluta er oft innbyrðis tengt. T.a.m. eru jeppar bara lítið breyttir bílar. Í staðinn fyrir að skilgreina jeppa alveg frá grunni þá segjum við að jeppar séu eins og bílar, nema bara... og svo segjum við að hvaða leyti jeppar eru öðruvísi en bílar. Það er minna mál en að segja hvað jeppar eru alveg frá grunni. Þarna endurnýtum við þekkingu sem við þegar höfum á bílum til að skilgreina jeppa. Þetta kallast erfðir. Við látum jeppa erfa bíla og að svo stöddu þurfum við bara að segja hvað einkennir jeppa umfram bíla.

Annað gott dæmi væri sjúkrabílar. Þeir eru eins og bílar nema bara að þeir eru hvítir. Þeir hafa á sér Rauða Kross merkið o.s.frv.

13.8 Grunn klasinn

```
class hus():
    def __init__(self,fermetrar):
        self.fermetrar=fermetrar;
    def prenta(self):
        print("Fermetrar: ", self. fermetrar);
```

Við getum notað þennan klasa sem grunn klasa. Skrifum nú klasana fyrir hús.

13.9 Mismunandi tegundir klasa

```
class Hol(losunarskrin):
    def __init__(self):
        self.fermetrar=50;
        self.prenta();
```

14 Almennt um hugbúnaðarþróun

Þegar við erum að skrifa forrit þá lendum við í ýmsum aðstæðum sem er í rauninni ekkert einstakar fyrir Python heldur er um að ræða hugsun eða speki sem á við um hugbúnaðarþróun almennt. Hér á eftir verður farið yfir nokkur slík atriði.

14.1 Um breytuheiti

Þegar við erum að búa til breytur þá er mikilvægt að heitin segja notandanum eins mikið um hvað breyturnar standa fyrir og hægt er, eins hratt og mögulegt er með góðu móti. Einnig er mikilvægt að breyturnar séu ekki að ljúga á neinn hátt. Vönduð breytuheiti eru gulls-ígildi og einn af vanmetnari eiginleikum þeirra.

14.2 Breytuheiti sem ber að forðast

Þegar við ákveðum breytuheiti þá ber að forðast eftirfarandi breytuheiti

- 1 (einn) og l (stafurinn ell)
- i og t
- O (bókstafurinn) og 0 (talan)

Ástæðan er augljós, þessi breytuheiti eru ekki bara óskýr og segja manni ekkert um hvað þau standa fyrir. Þau eru beinlínis hættuleg þar sem auðvelt er að rugla saman við hvort annað. Það er engin ástæða til að veita notandanum engar upplýsingar um hvað er að gerast, og stilla honum þar að auki upp til að labba ofan í þá holu að rugla þeim saman við hinn möguleikann.

15 Setja upp Raspberry Pi

Til að byrja með Raspberry Pi þarftu eftirfarandi:

- Aflgjafa
- Ræsimiðil (t.d. MicroSD-kort með nægu geymsluplássi og góðum leshraða)

Þú getur sett upp Raspberry Pi annaðhvort sem gagnvirka tölvu með skjáborði (e. desktop), eða sem höfuðlausa tölvu (e. headless), sem er aðeins aðgengileg yfir netið.

Til að stilla Raspberry Pi höfuðlausan þarftu ekki nein viðbótarjaðartæki. Þú getur forstillt hýsingarnafn, notandareikning, nettengingu og SSH-aðgang þegar þú setur upp stýrikerfið.

Ef þú vilt hins vegar nota Raspberry Pi með skjá og innsláttarbúnað, þarftu eftirfarandi aukabúnað:

- Skjá
- Snúru til að tengja Raspberry Pi við skjáinn
- Lyklaborð
- Mús

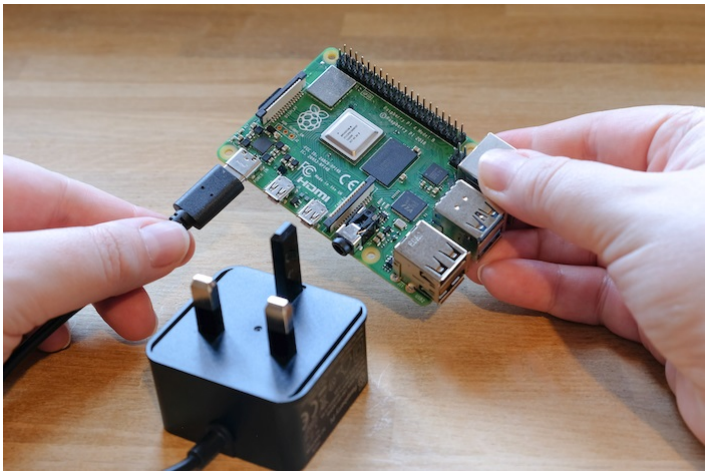
15.1 Alfgjafi

Eftirfarandi tafla sýnir USB-PD aflstillingu sem þarf til að knýja ýmsar Raspberry Pi gerðir. Þú getur notað hvaða hágæða aflgjafa sem er sem veitir rétta aflstillingu.

Stingdu aflgjafanum þínum í tengið merkt „POWER IN“, „PWR IN“ eða „PWR“. Sumar Raspberry Pi gerðir, eins og Zero serían, eru með USB-tengi með sama formstuðli og rafmagnstengi. Vertu viss um að nota rétta tengið á Raspberry Pi þínum!

Fyrirmynd	Raspberry Pi aflgjafi
Raspberry Pi 5	5V / 3A 27W USB-C aflgjafi
Raspberry Pi 4 Model B	5V / 3A 15W USB-C
Raspberry Pi 3 (allar gerðir)	5V / 2,5A 12,5W Micro USB
Raspberry Pi 2 (allar gerðir)	5V / 3,5A 12,5W Micro USB
Raspberry Pi 1 (allar gerðir)	5V / 3,5A 12,5W Micro USB
Raspberry Pi Zero (allar gerðir)	5V / 3,5A 12,5W Micro USB

Tafla 15.1: Raspberry Pi aflgjafar og mælt aflgjafa straumur



Mynd 15.1: Setja í samband við rafmagn

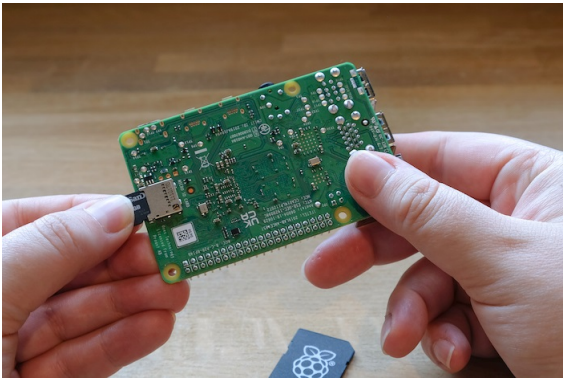
15.2 Boot media

Raspberry Pi-gerðir skortir geymslu um borð, svo þú verður að útvega hana. Þú getur ræst Raspberry Pi frá stýrikerfismynd sem er uppsett á hvaða studdu miðli sem er: microSD-kort eru almennt notuð, en USB-geymsla, netgeymsla og geymsla tengd PCIe HAT eru einnig fáanleg. Hins vegar styðja aðeins nýlegar Raspberry Pi-gerðir allar þessar fjölmiðlagerðir.

Allar Raspberry Pi-neytendagerðir síðan Raspberry Pi 1 Model A+ eru með microSD-rauf. Raspberry Pi þinn ræsir sjálfkrafa úr microSD-raufinni þegar raufin inniheldur kort.

15.3 Mælt er með SD kortum

Við mælum með því að nota SD-kort með að minnsta kosti 32GB geymsluplássi fyrir Raspberry Pi OS uppsetningar. Fyrir Raspberry Pi OS Lite mælum við með



Mynd 15.2: Setja í samband við rafmagn

að minnsta kosti 16GB. Þú getur notað hvaða SD-kort sem er sem er minna en 2TB. Stærð yfir 2TB er ekki studd eins og er vegna takmarkana í MBR. Eins og með alla aðra ræsimiðla muntu sjá betri afköst á SD-kortum með hraðari les- og skrifhraða. Ef þú ert ekki viss um hvaða SD-kort þú átt að kaupa skaltu íhuga opinberu SD-kort Raspberry Pi. Önnur stýrikerfi hafa aðrar kröfur. Skoðaðu skjölin fyrir stýrikerfið þitt fyrir getuþörf. Við mælum með því að nota SD-kort með að minnsta kosti 32GB geymsluplássi fyrir Raspberry Pi OS uppsetningar. Fyrir Raspberry Pi OS Lite mælum við með að minnsta kosti 16GB. Þú getur notað hvaða SD-kort sem er sem er minna en 2TB. Stærð yfir 2TB er ekki studd eins og er vegna takmarkana í MBR (e.*master boot record*). Eins og með alla aðra ræsimiðla muntu sjá betri afköst á SD-kortum með hraðari les- og skrifhraða. Ef þú ert ekki viss um hvaða SD- kort þú átt að kaupa skaltu íhuga opinberu SD- kort Raspberry Pi. Önnur stýrikerfi hafa aðrar kröfur. Skoðaðu skjölin fyrir stýrikerfið þitt fyrir getuþörf.

15.4 Lyklaborð

Þú getur notað hvaða USB tengi sem er á Raspberry Pi þínum til að tengja lyklaborð með snúru eða USB Bluetooth móttakara.

Þú getur notað hvaða USB-tengi sem er á Raspberry Pi þínum til að tengja lyklaborð með snúru eða USB Bluetooth-móttakara.

15.5 Mús

Þú getur notað hvaða USB tengi sem er á Raspberry Pi þínum til að tengja mús með snúru eða USB Bluetooth móttakara.

Mynd 15.3: Tengja lykaborð við USB

Mynd 15.4: Mús teng við USB

15.6 Skjár

Raspberry Pi gerðir eru með eftirfarandi skjátengingu:

Athugið: Engar Raspberry Pi gerðir styðja myndband yfir USB-C (DisplayPort alt mode).

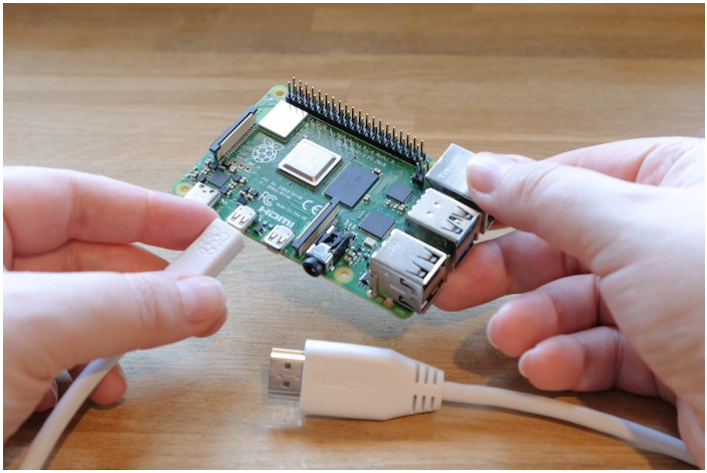
Ef Raspberry Pi þinn er með fleiri en eitt HDMI tengi skaltu tengja aðalskjáinn þinn í tengið merkt HDMI0 .

Flestir skjáir eru ekki með micro eða mini HDMI tengi. Hins vegar geturðu notað micro-HDMI-til-HDMI snúru eða mini-HDMI-til-HDMI snúru til að tengja þessi tengi á Raspberry Pi þínum við hvaða HDMI skjá sem er. Fyrir skjái sem styðja ekki HDMI skaltu íhuga millistykki sem þýðir skjáúttak frá HDMI yfir í

Fyrirmynd	Sýna úttak
Raspberry Pi 5	2× micro HDMI
Raspberry Pi 4	2× micro HDMI, 3,5 mm TRRS tengi
Raspberry Pi 3	HDMI, hljóð 3,5 mm
Raspberry Pi 2	HDMI, hljóð 3,5 mm
Raspberry Pi 1 Model B+	HDMI, 3,5 mm
Raspberry Pi 1 Model A+	HDMI, 3,5 mm
Raspberry Pi Zero (allar gerðir)	lítill HDMI

Tafla 15.2: Úttak fyrir mismunandi Raspberry Pi gerðir

tengi sem skjárinn þinn styður.



Mynd 15.5: Tengt viðskjá

15.7 Hjóð

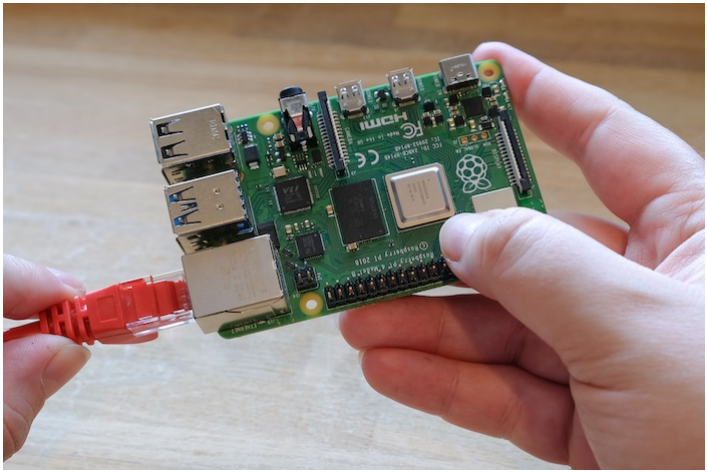
Allar Raspberry Pi gerðir með HDMI, micro HDMI eða mini HDMI styðja hljóðúttak yfir HDMI. Allar Raspberry Pi gerðir styðja hljóð yfir USB. Allar Raspberry Pi gerðir með Bluetooth styðja Bluetooth hljóð. Öll afbrigði af Raspberry Pi 1, 2, 3 og 4 eru með 3,5 mm auka TRRS tengi, sem gæti þurft mögnun fyrir nægjanlegt úttaksrúmmál.

15.8 Netkerfi

Eftirfarandi Raspberry Pi gerðir eru með Wi-Fi og Bluetooth tengingu: • Flaggskipsmódel síðan Raspberry Pi 3 Model B

- Allar Zero W módel
- Allar Pico W gerðir
- Compute Modules stillt með þráðlausum (fáanlegt síðan CM4)

Viðskeytið „Model B“ gefur til kynna afbrigði með Ethernet tengi; „Módel A“ gefur til kynna engin Ethernet tengi. Ef Raspberry Pi þinn er ekki með Ethernet tengi geturðu samt tengst við þráð nettengingu með USB-til-Ethernet millistykki.



Mynd 15.6: RJ-45 Netkapal tendur

15.9 Setja upp stýrikerfi

Til að nota Raspberry Pi þinn þarftu stýrikerfi. Sjálfgefið er að Raspberry Pis leitar að stýrikerfi á hvaða SD-korti sem er sett í SD-kortaraufina.

Það fer eftir Raspberry Pi-gerðinni þinni, þú getur líka ræst stýrikerfi úr öðrum geymslutækjum, þar á meðal USB-drifum, geymslu sem tengd er með HAT og netgeymslu.

Til að setja upp stýrikerfi á geymslutæki fyrir Raspberry Pi þinn þarftu:

- Tölvu sem þú getur notað til að mynda geymslutækið í ræsitæki
- Leið til að tengja geymslutækið þitt við þá tölvu

Flestir Raspberry Pi-notendur velja microSD-kort sem ræsibúnað.

Við mælum með því að setja upp stýrikerfi með Raspberry Pi Imager.

Raspberry Pi Imager er tól sem hjálpar þér að hlaða niður og skrifa myndir á macOS, Windows og Linux. Imager inniheldur margar vinsælar stýrikerfismyndir fyrir Raspberry Pi. Myndavél styður einnig að hlaða myndum niður beint frá Raspberry Pi eða þriðja aðila eins og Ubuntu. Þú getur notað Imager til að forstilla skilríki og fjaraðgangsstillingar fyrir Raspberry Pi þinn.

Myndavélin styður myndir sem eru pakkaðar í. img sniði sem og gámasnið eins og.zip.

Ef þú hefur enga aðra tölvu til að skrifa mynd í ræsitæki gætirðu sett upp stýrikerfi beint á Raspberry Pi af Internetinu.

15.10 Setum upp image

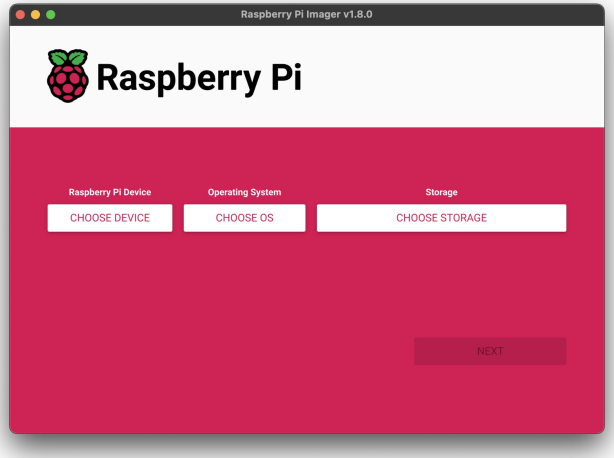
Þú getur sett upp Imager á eftirfarandi hátt:

- Sæktu nýjustu útgáfuna af raspberrypi og keyrðu uppsetningarforritið.
- Settu það upp frá útstöð með því að nota pakkastjórann þinn.

```
sudo apt install rpi -imager.
```

Þegar þú hefur sett upp Imager skaltu ræsa forritið með því að smella á Raspberry Pi Imager táknið eða keyra

```
rpi -imager
```



Mynd 15.7: Valmynd 1

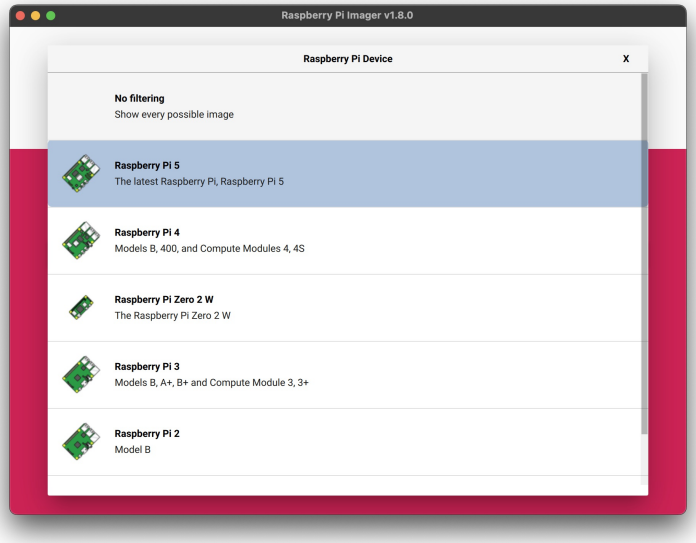
Smelltu á Veldu tæki og veldu Raspberry Pi líkanið þitt af listanum.

Næst skaltu smella á Veldu stýrikerfi og velja stýrikerfi til að setja upp. Myndavél sýnir alltaf ráðlagða útgáfu af Raspberry Pi OS fyrir líkanið þitt efst á listanum.

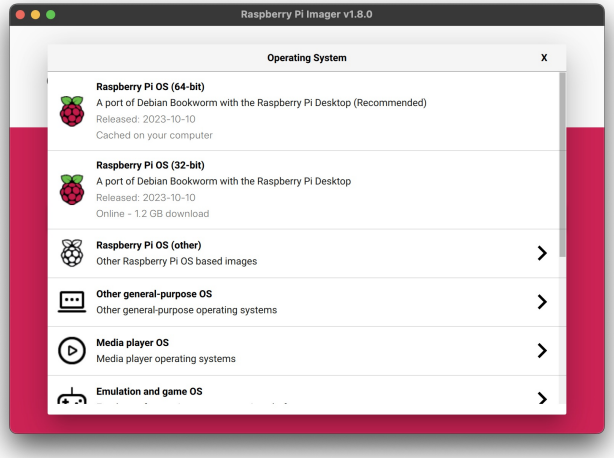
Tengdu valinn geymslutæki við tölvuna þína. Tengdu til dæmis microSD kort í með ytri eða innbyggðum SD kortalesara. Smelltu síðan á Veldu geymslu og veldu geymslutæki. Viðvörun Ef þú ert með fleiri en eitt geymslutæki tengt við tölvuna þína, vertu viss um að velja rétt tæki! Þú getur oft greint geymslutæki eftir stærð. Ef þú ert ekki viss skaltu aftengja önnur tæki þar til þú hefur auðkennt tækið sem þú vilt taka mynd af.

Næst skaltu smella á Next .

Í sprettiglugga mun Imager biðja þig um að beita sérsniðnum stýrikerfi. Við mælum eindregið með því að stilla Raspberry Pi þinn í gegnum stýrikerfisstillingarnar.

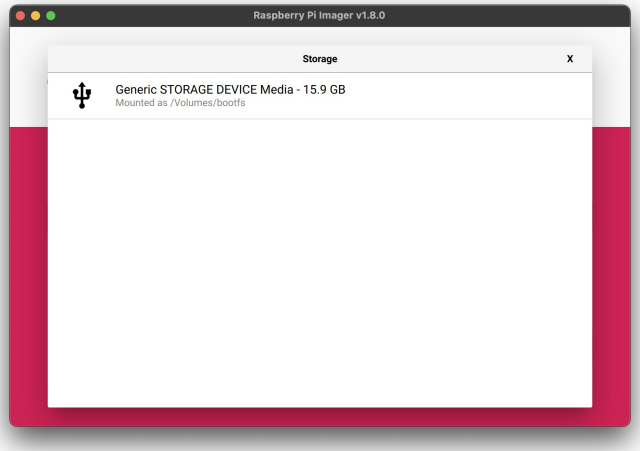


Mynd 15.8: Valmynd 2



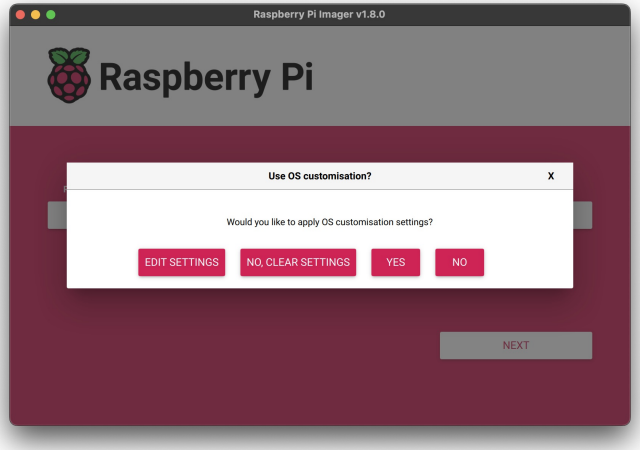
Mynd 15.9: Valmynd 3

Smelltu á hnappinn Breyta stillingum til að opna stýrikerfisaðlögun . Ef þú stillir Raspberry Pi ekki í gegnum stýrikerfisstillingar mun Raspberry Pi OS biðja



Mynd 15.10: Valmynd 4

þig um sömu upplýsingar við fyrstu ræsingu meðan á stillingarhjálpinni stendur .



Mynd 15.11: Valmynd 5

Þú getur smellt á **Nei** hnappinn til að sleppa sérstillingu stýrikerfisins.

15.11 OS Aðlögun

Sérstillingarvalmynd stýrikerfisins gerir þér kleift að setja upp Raspberry Pi fyrir fyrstu ræsingu. Þú getur forstillt: • Notendanafn og lykilorð

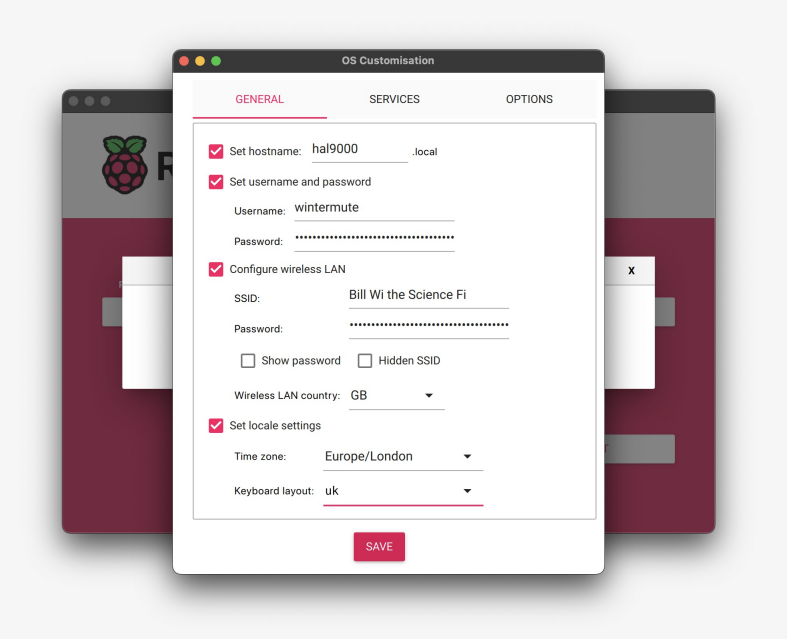
- Wi-Fi skilríki
- Hýsingarheiti tækisins
- Tímabeltið
- Lyklaborðinu þínu
- Fjartenging

Þegar þú opnar fyrst stýrikerfisvalmyndina gætirðu séð hvetja sem biður um leyfi til að hlaða Wi-Fi skilríkjum frá hýsingartölvunni þinni. Ef þú svarar „já“ mun Imager fylla út Wi-Fi skilríki frá netinu sem þú ert tengdur við. Ef þú svarar „nei“ geturðu slegið inn Wi-Fi skilríki handvirkt.

Hýsingarnafnvalkosturinn skilgreinir hýsilnafnið sem Raspberry Pi sendir út á netið með því að nota **mDNS**.

Þegar þú tengir Raspberry Pi við netið þitt geta önnur tæki á netinu átt samskipti við tölvuna þína með *<hostname>.local eða <hostname>. lan* .

Notandanafn og **Lykilorð** valkostur skilgreinir notandanafn og lykilorð á admin notandareikningnum á Raspberry Pi þínum.

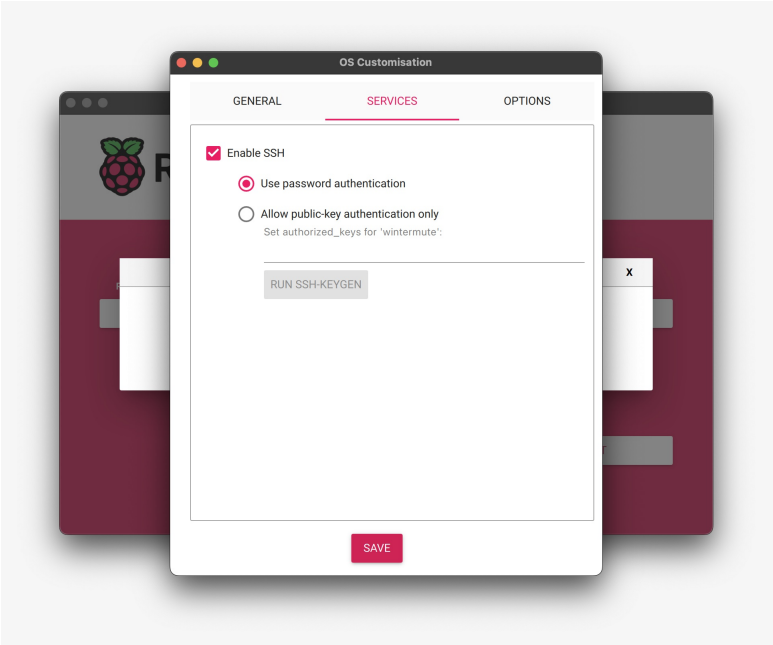


Mynd 15.12: Nafn,Notandi og lykilorð

Þráðlausa staðarnetsvalkosturinn

Gerir þér kleift að slá inn SSID (nafn) og lykilorð fyrir þráðlausa netið þitt. Ef netið þitt sendir ekki út SSID opinberlega, ættir þú að virkja „Falið SSID“ stillinguna. Sjálfgefið er að Imager notar landið sem þú ert í sem „Þráðlaust staðarnetsland“. Þessi stilling stjórnar Wi-Fi útsendingartíðnum sem Raspberry Pi þinn notar.

Sláðu inn skilríki fyrir þráðlausa staðarnetsvalkostinn ef þú ætlar að keyra höfuðlausan Raspberry Pi. Staðbundin stillingarvalkosturinn gerir þér kleift að skilgreina tímabelti og sjálfgefið lyklaborðsuppsetningu fyrir Pi þinn.

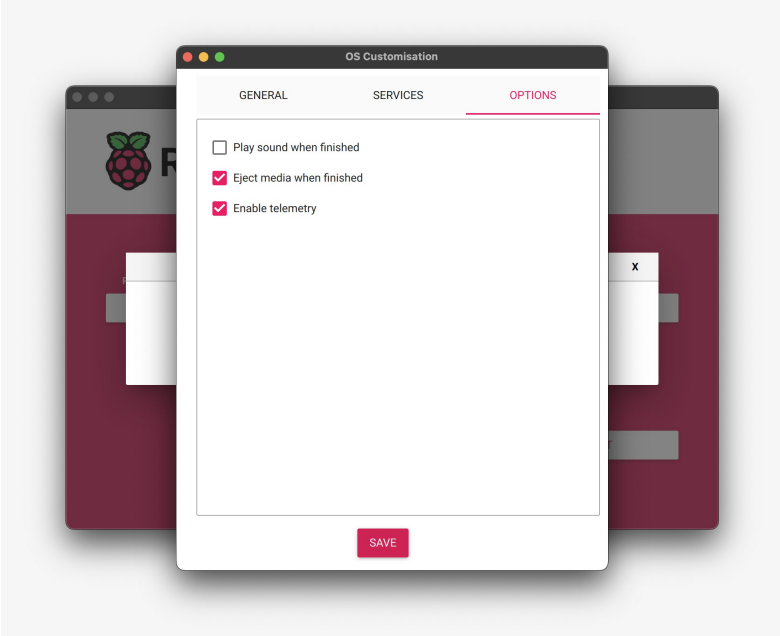


Mynd 15.13: Kveykja á SSH

Þjónusta flipinn inniheldur stillingar til að hjálpa þér að tengjast Raspberry Pi fjarstýrt. Ef þú ætlar að nota Raspberry Pi lítillega yfir netið þitt skaltu haka í reitinn við hliðina á Virkja SSH . Þú ættir að virkja þennan valkost ef þú ætlar að keyra hauslausan Raspberry Pi.

Veldu auðkenningarvalkostinn fyrir lykilorð til að SSH inn í Raspberry Pi þinn í gegnum netið með því að nota notandanafnið og lykilorðið sem þú gafst upp á almennum flipanum fyrir sérstillingu stýrikerfisins.

Veldu Leyfa eingöngu auðkenningu almenningslykils til að forstilla Raspberry Pi þinn fyrir SSH auðkenningu án lykilorðs með því að nota einkalykil úr tölvunni sem þú ert að nota núna. Ef þú ert þegar með RSA lykil í SSH uppsetningunni þinni , notar Imager þann opinbera lykil. Ef þú gerir það ekki geturðu smellt



Mynd 15.14: Valkvæt möguleikar

á Run SSH- keygen til að búa til opinbert/einka lykilpar. Myndavél mun nota nýstofnaða opinbera lykilinn.

Aðlögun stýrikerfis felur einnig í sér **Valkostavalmynd** sem gerir þér kleift að stilla hegðun Imager meðan á ritun stendur. Þessir valkostir gera þér kleift að spila hávaða þegar Imager lýkur við að staðfesta mynd, aftengja geymslumiðil sjálfkrafa eftir staðfestingu og slökkva á fjarmælingum.

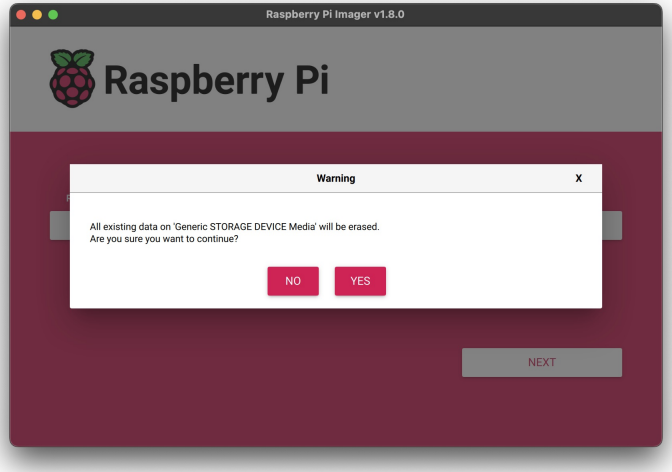
15.12 Skrifaðu

Þegar þú hefur lokið við að slá inn stýrikerfisstillingar, smelltu á Vista til að vista sérstillinguna þína. Smelltu síðan á Já til að nota stýrikerfisstillingar þegar þú skrifar myndina í geymslutækið. Að lokum skaltu svara já við Ęrtu viss um að þú viljir halda áfram?"sprettiglugga til að byrja að skrifa gögn í geymslutækið.

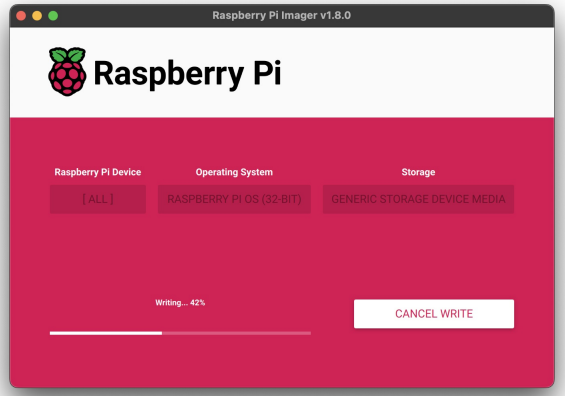
Ef þú sérð stjórnandakvaðningu sem biður um leyfi til að lesa og skrifa á geymslumiðilinn þinn skaltu veita Imager leyfi til að halda áfram.

Fáðu þér kaffibolla eða spjala við Kennaran sigga :D Þetta gæti tekið nokkrar mínútur.

Ef þú vilt lifa sérstaklega hættulega geturðu smellt á Hætta við staðfestingu til að sleppa staðfestingarferlinu. Þegar þú sérð sprettigluggann „Skrifað tókst“

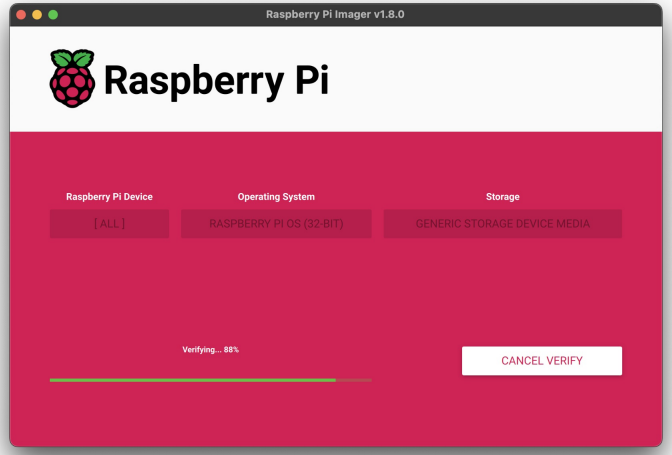


Mynd 15.15: Veljum já hér



Mynd 15.16: OS er set á SD-kort

hefur myndin þín verið skrifuð að fullu og staðfest. Þú ert nú tilbúinn til að ræsa Raspberry Pi úr SD-kort!



Mynd 15.17: Tilbúið og velja Continue

15.13 Settu upp í gegnum netið

Network Install gerir Raspberry Pi kleift að setja upp stýrikerfi á geymslutæki með því að nota útgáfu af Raspberry Pi Imager sem hlaðið er niður í gegnum netið. Með Network Install geturðu sett upp stýrikerfi á Raspberry Pi þínum án sérstakrar SD kortalesara og engrar tölvu fyrir utan Raspberry Pi þinn. Þú getur keyrt Network Install á hvaða samhæfu geymslutæki sem er, þar á meðal SD kort og USB geymslu. Netuppsetning keyrir aðeins á flaggskipsmódelum síðan Raspberry Pi 4B og lyklaborðsmódel. Ef Raspberry Pi þinn keyrir eldri ræsiforrit gætirðu þurft að uppfæra ræsiforritið til að nota Network Install. Netuppsetning krefst eftirfarandi:

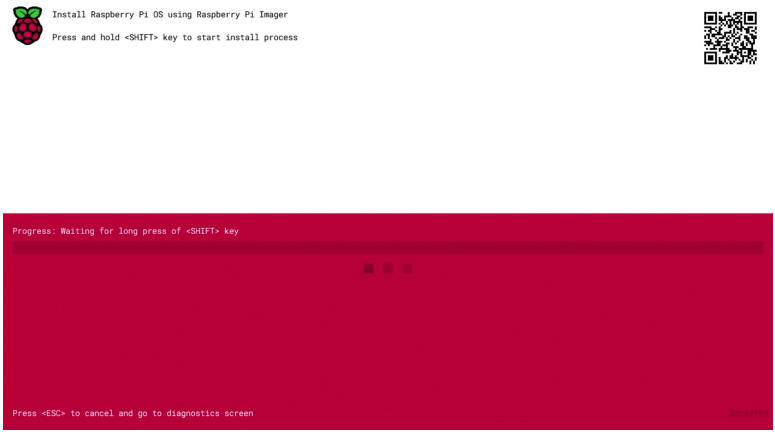
- Raspberry Pi líkan sem keyrir fastbúnað sem styður netuppsetningu
- Skjár
- Lyklaborð
- Þráðlaus nettenging

Til að ræsa Network Install skaltu kveikja á Raspberry Pi á meðan þú ýtir á og heldur inni SHIFT takkanum í eftirfarandi stillingum:

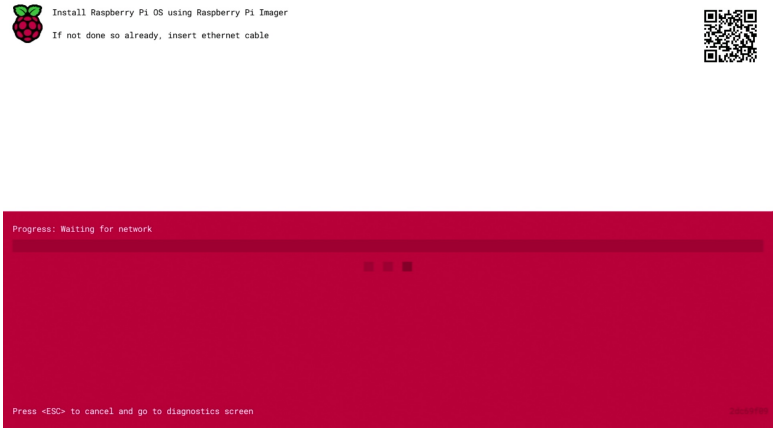
- Ekkert ræsanlegt geymslutæki
- Meðfylgjandi lyklaborð
- SD-kort eða USB-geymsla

Ef þú hefur ekki þegar tengt Raspberry Pi við internetið skaltu tengja það með Ethernet snúru.

Þegar þú ert tengdur við internetið mun Raspberry Pi þinn hala niður Raspberry



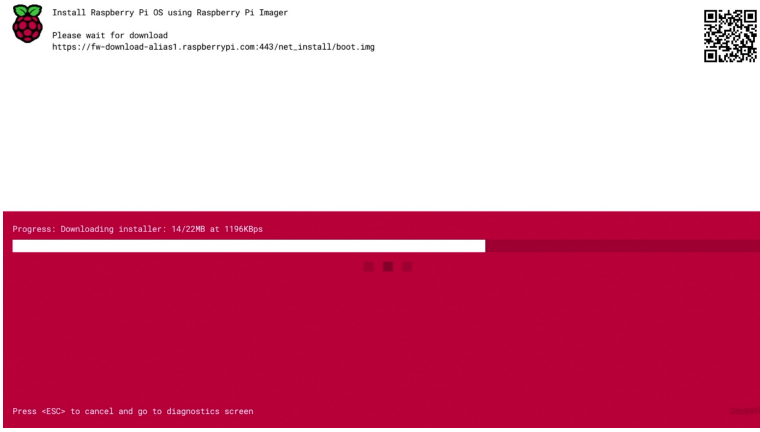
Mynd 15.18: Valkvæt möguleikar



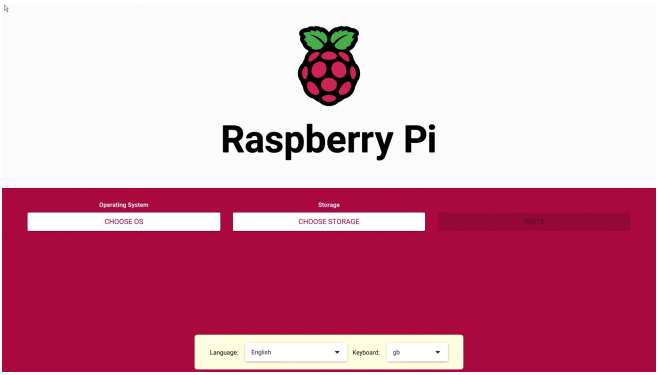
Mynd 15.19: Valkvæt möguleikar

Pi uppsetningarforritinu. Ef niðurhalið mistekst geturðu endurtekið ferlið til að reyna aftur.

Þegar þú hefur lokið við að hlaða niður Raspberry Pi Installer mun Raspberry Pi þinn sjálfkrafa ræsa Raspberry Pi Imager. Fyrir frekari upplýsingar um að keyra Raspberry Pi Imager,



Mynd 15.20: Valkvæt möguleikar



Mynd 15.21: Valkvæt möguleikar

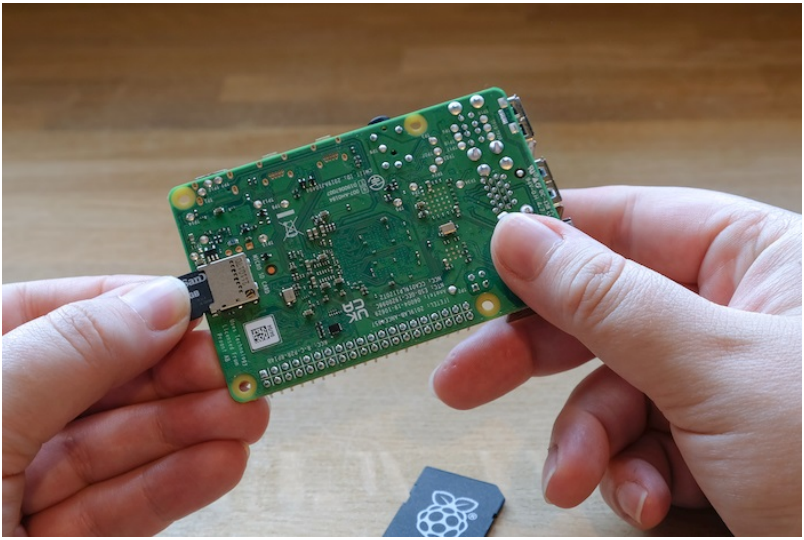
15.14 Settu upp þinn Raspberry Pi :D

Eftir að hafa sett upp stýrikerfismynd skaltu tengja geymslutækið þitt við Raspberry Pi þinn.

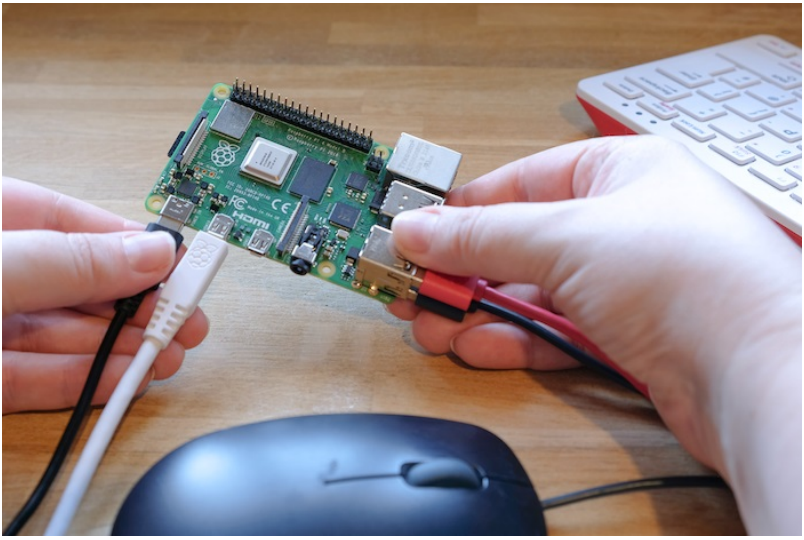
Taktu fyrst aflgjafa Raspberry Pi úr sambandi til að tryggja að slökkt sé á Raspberry Pi á meðan þú tengir jaðartæki. Ef þú settir upp stýrikerfið á microSD-korti geturðu tengt það í Raspberry Pi-kortaraufina núna. Ef þú settir upp stýrikerfið á einhverju öðru geymslutæki geturðu tengt það við RaspberryPi núna.

Settu síðan önnur jaðartæki í samband, eins og músina, lyklaborðið og skjáinn.

Að lokum skaltu tengja aflgjafann við Raspberry Pi þinn. Þú ættir að sjá stöðuljósið kvikna þegar kveikt er á Pi þínum. Ef Pi þinn er tengdur við skjá ættirðu að sjá ræsiskjáinn innan nokkurra mínútna.



Mynd 15.22: Valkvæt möguleikar



Mynd 15.23: Valkvæt möguleikar

15.15 Stillingar við fyrstu ræsingu

Ef þú notaðir stýrikerfisaðlögun í Imager til að forstilla Raspberry Pi þinn, til hamingju! Tækið þitt er tilbúið til notkunar. Haltu áfram í næstu skref til að

læra hvernig þú getur nýtt Raspberry Pi þinn vel.

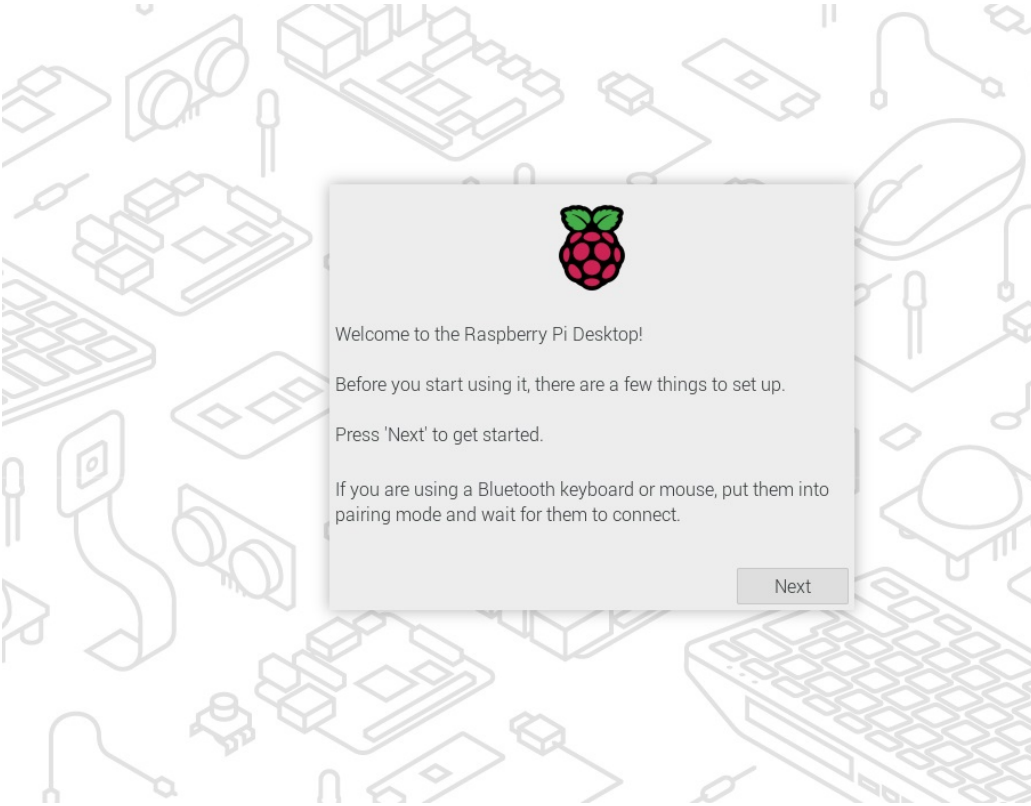
Ef Raspberry Pi þinn ræsist ekki innan 5 mínútna skaltu athuga stöðuljósið. **Ef það blikkar, Sjá Kafla *LED warning flash codes***

1. Ef Pi þinn neitar að ræsa skaltu prófa eftirfarandi mótvægisskref:

2. ef þú notar annað ræsitæki en SD-kort skaltu prófa að ræsa af SD-korti

3. endurmyndaðu SD-kortið þitt ; vertu viss um að klára allt staðfestingarskrefið í Imager uppfærðu ræsiforritið á Raspberry Pi þínum, endurmyndaðu síðan SD-kortið þitt

Ef þú velur að sleppa stýrikerfisaðlögun í Imager mun Raspberry Pi keyra uppsetningarhjálp við fyrstu ræsingu. Þú þarft skjá og lyklaborð til að fletta í gegnum töframanninn; mús er valfrjáls.



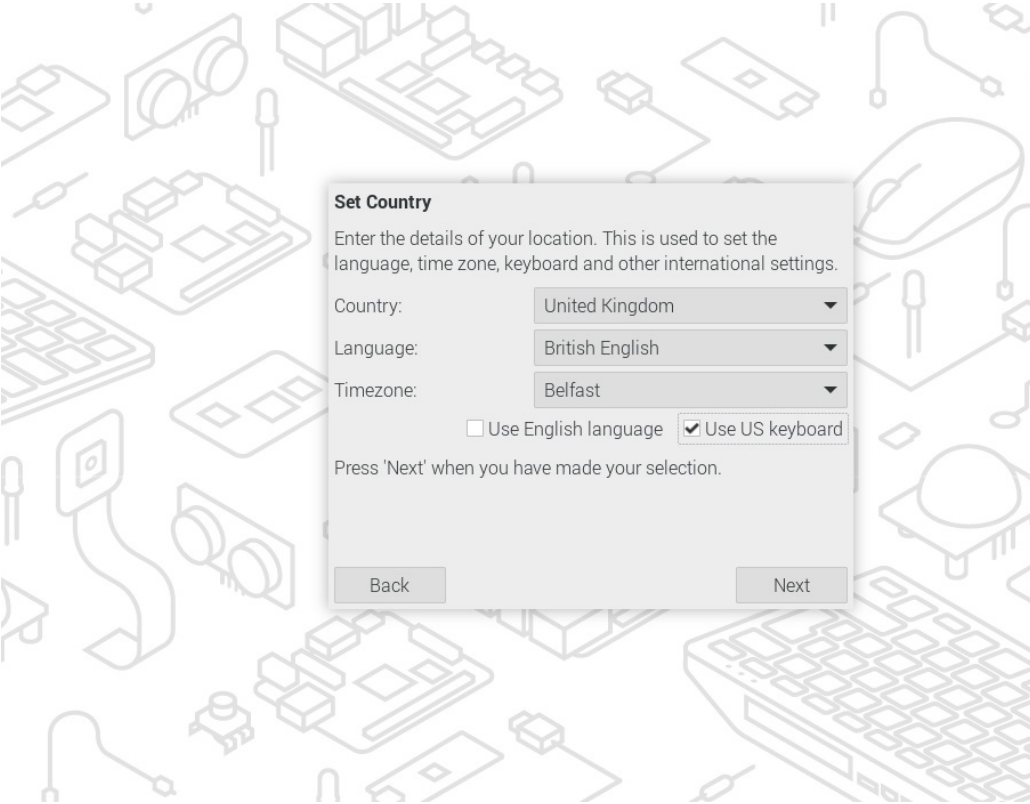
Mynd 15.24: Valkvæt möguleikar

15.16 Bluetooth

Ef þú ert að nota Bluetooth lyklaborð eða mús mun þetta skref leiða þig í gegnum pörun tækja. Raspberry Pi þinn mun leita að pöruðum tækjum og tengjast fyrsta tækinu sem það finnur fyrir hvern hlut. Þetta ferli virkar með innbyggðum eða ytri USB Bluetooth millistykki. Ef þú notar USB millistykki skaltu tengja það við áður en þú ræsir Raspberry Pi.

15.17 Staðsetining

Þessi síða hjálpar þér að stilla land þitt, tungumál og tímabelti og lyklaborðsuppsetningu.



Mynd 15.25: Tími og tímabelti

15.18 Notandi

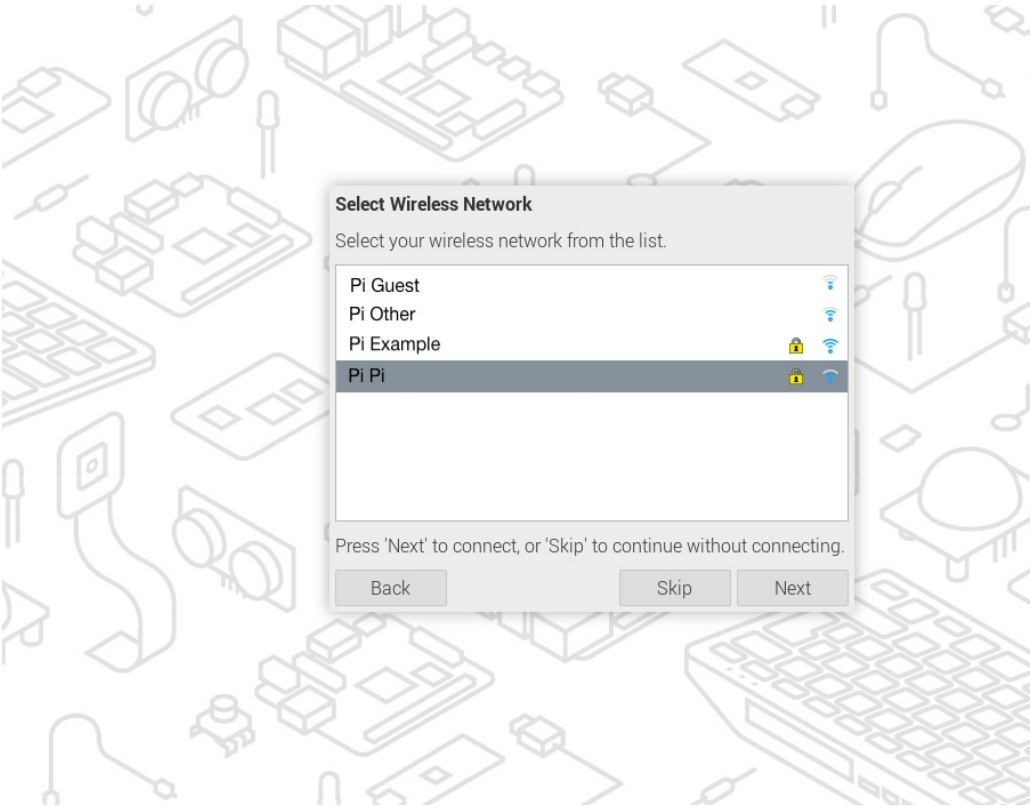
Þessi síða hjálpar þér að stilla notandanafn og lykilorð fyrir sjálfgefna notendareikninginn. Sjálfgefið er að eldri útgáfur af Raspberry Pi OS setja notandanafnið á „pi“. Ef þú notar notandanafnið „pi“, forðastu gamla sjálfgefna lykilorðið „raspberry“ til að halda Raspberry Pi öruggum.

Mynd 15.26: Notendanaf og lykilorð

15.19 Wi-Fi

Þessi síða hjálpar þér að tengjast Wi-Fi neti. Veldu valinn netkerfi af listanum.

Ef netið þitt krefst lykilorðs geturðu slegið það inn hér.



Mynd 15.27: Velja Wi-FI

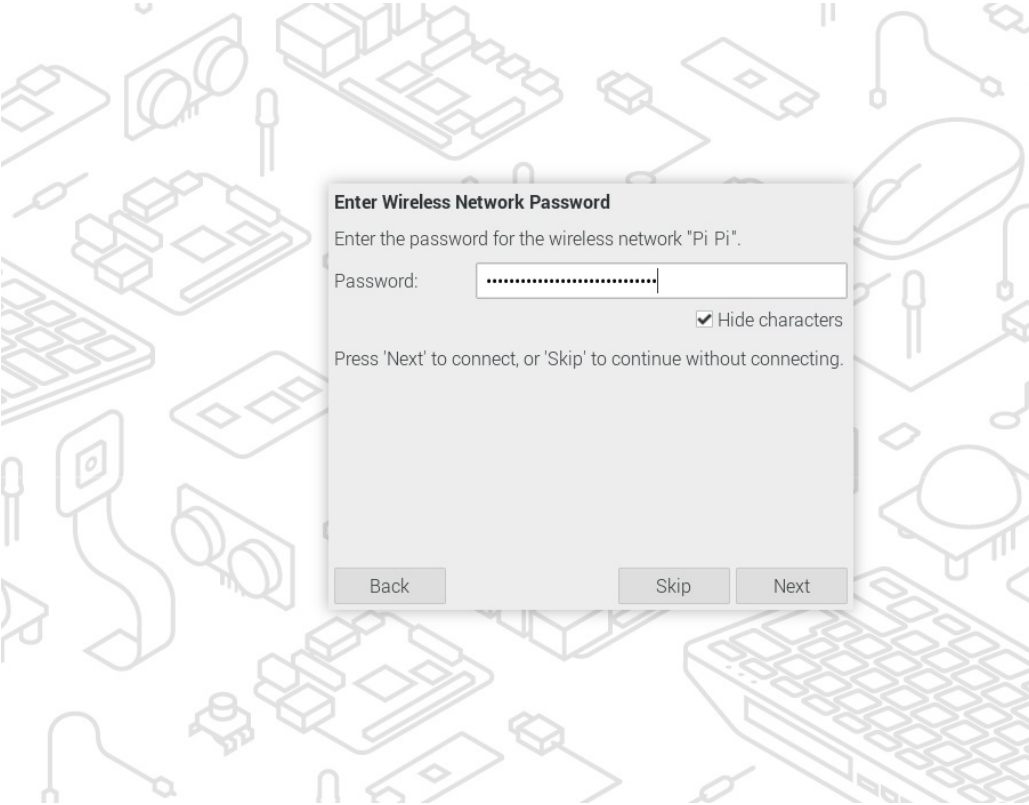
15.20 Borwser

Þessi síða gerir þér kleift að velja Firefox eða Chromium sem sjálfgefinn netvafra. Þú getur valfrjálst fjarlægt vafrann sem þú setur ekki sem sjálfgefinn.

15.21 Hugbúnaðaruppfærslur

Þegar Raspberry Pi þinn hefur aðgang að internetinu hjálpar þessi síða þér að uppfæra stýrikerfið og hugbúnaðinn í nýjustu útgáfur. Meðan á hugbúnaðaruppfærsluferlinu stendur mun töframaðurinn fjarlægja vafra sem ekki er sjálfgefinn ef þú valdir að fjarlægja hann í valskrefinu fyrir vafra. Það getur tekið nokkrar mínútur að hlaða niður uppfærslum.

Þegar þú sérð sprettiglugga sem gefur til kynna að kerfið þitt sé uppfært skaltu



Mynd 15.28: Lykilorð á Wi-Fi

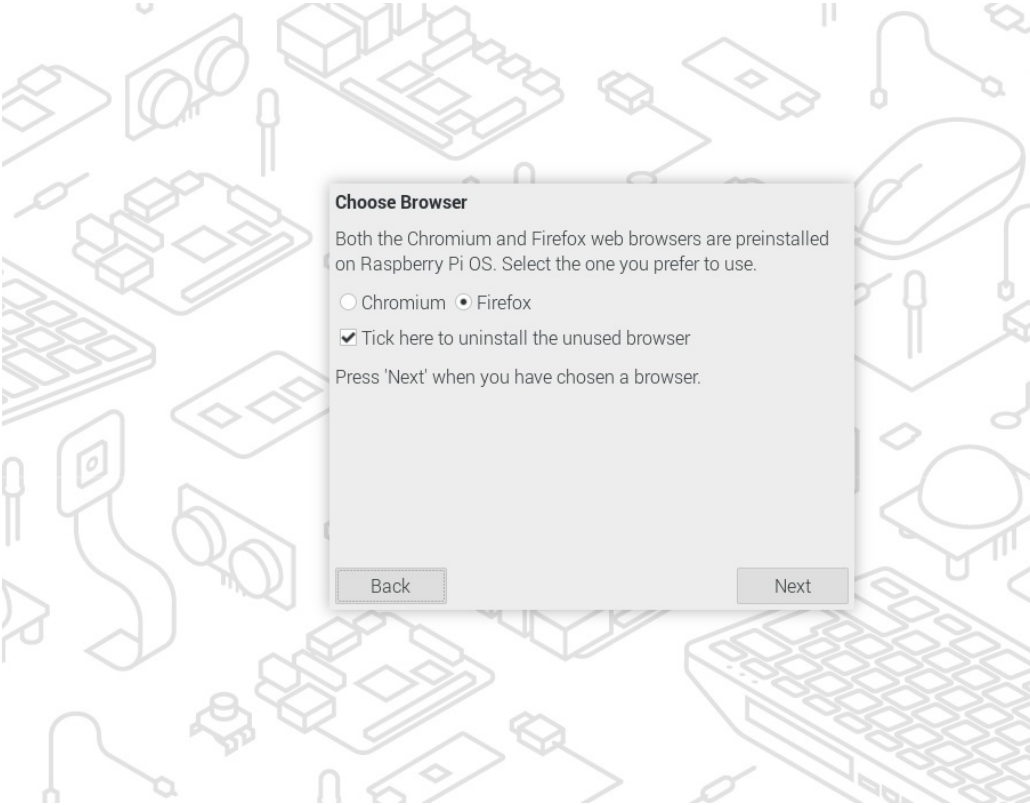
smella á OK til að halda áfram í næsta skref.

15.22 Enduræsing

Í lok stillingarhjálparinnar skaltu smella á Endurræsa til að endurræsa Raspberry Pi. Raspberry Pi þinn mun beita stillingum þínum og ræsa á skjáborðið.

15.23 Tilbúið

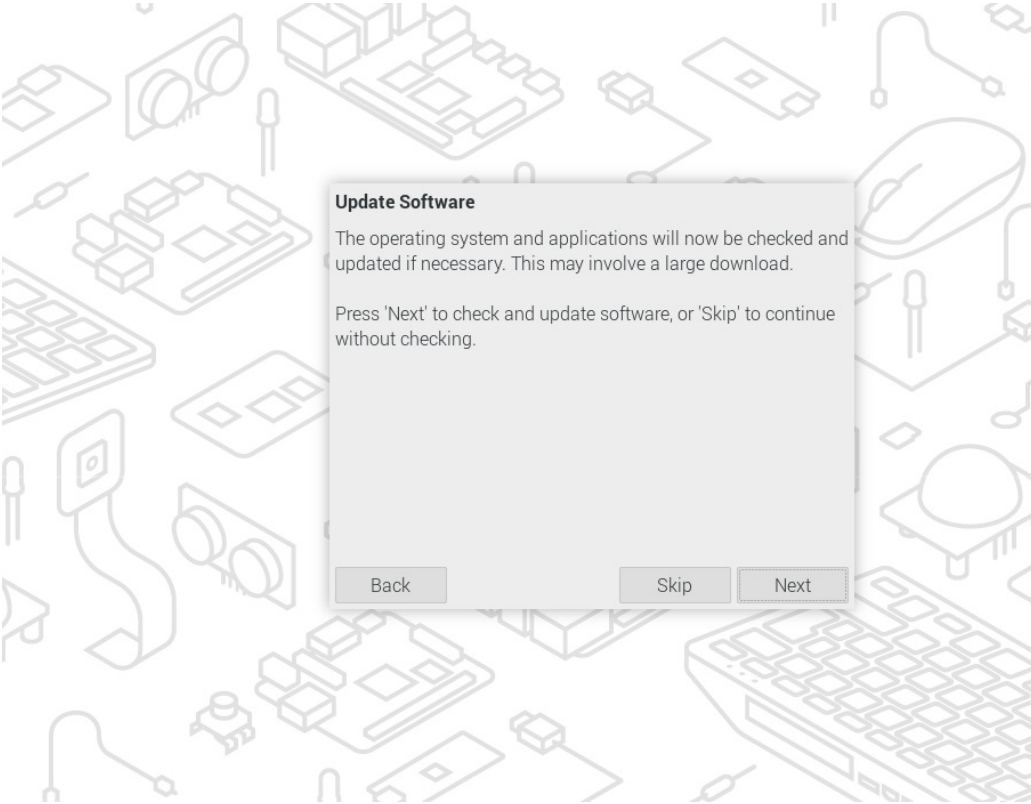
Nú þegar Raspberry Pi þinn er settur upp og tilbúinn til notkunar, hvað er næst? Raspberry Pi OS kemur með mörgum nauðsynlegum forritum fyrirfram uppsett svo þú getur byrjað að nota þau strax. Ef þú vilt nýta þér önnur forrit sem okkur finnst gagnleg, smelltu á hindberjatáknið efst í vinstra horninu á skjánum.



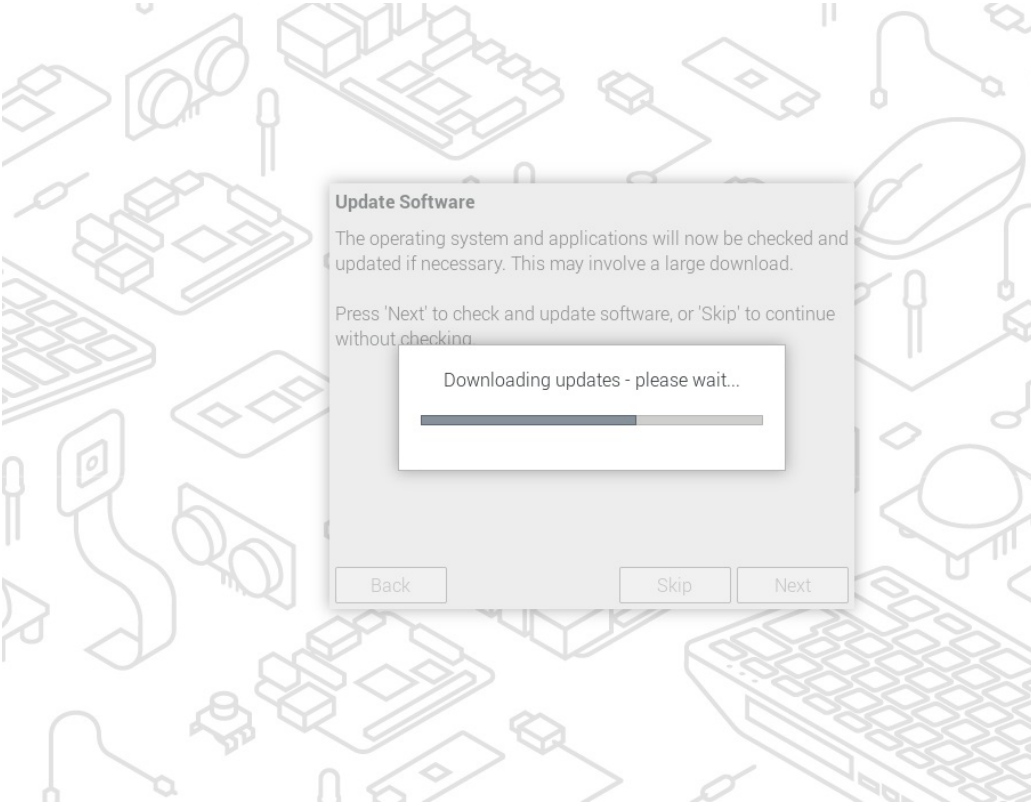
Mynd 15.29: Veja vafra

Veldu Preferences > Recommended Software í fellivalmyndinni og þú munt finna pakkastjórann. Þú getur sett upp fjölbreyttan hugbúnað sem mælt er með hér ókeypis.

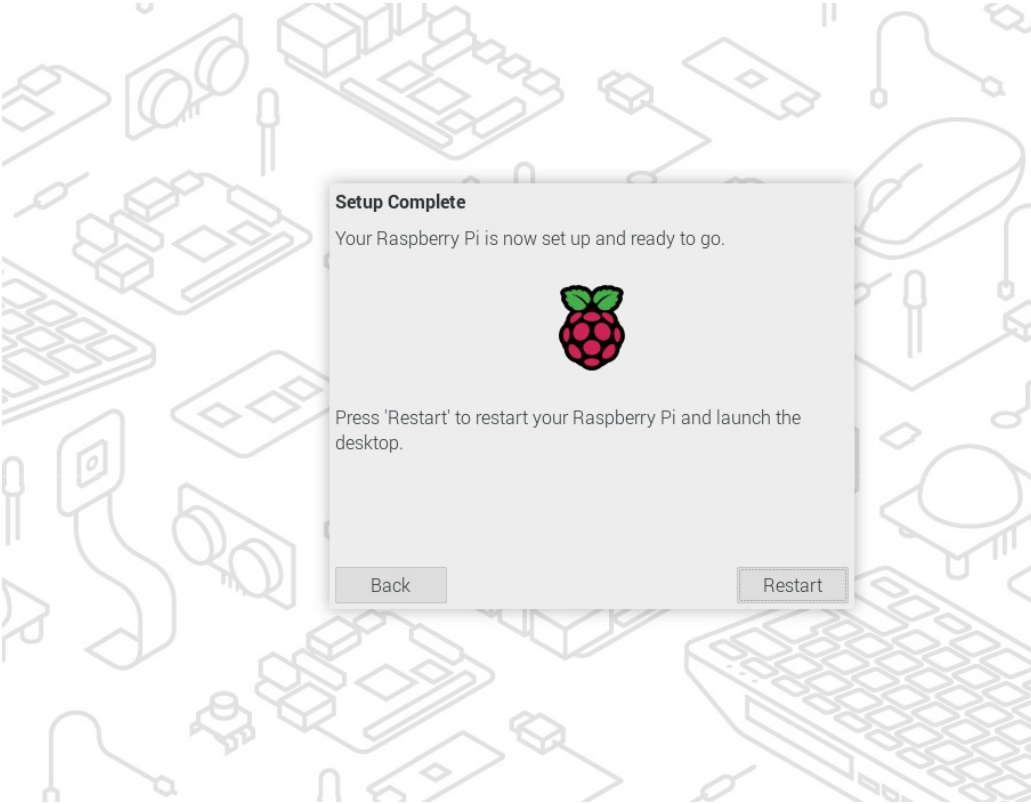
Til dæmis, ef þú ætlar að nota Raspberry Pi sem heimilistölvu gætirðu fundið LibreOffice gagnlegt til að skrifa og breyta skjölum og töflureiknum Þú getur líka gert Raspberry Pi aðgengilegri með forritum eins og skjástækkunargleri og Orca skjálesara, sem er að finna undir Universal Access.



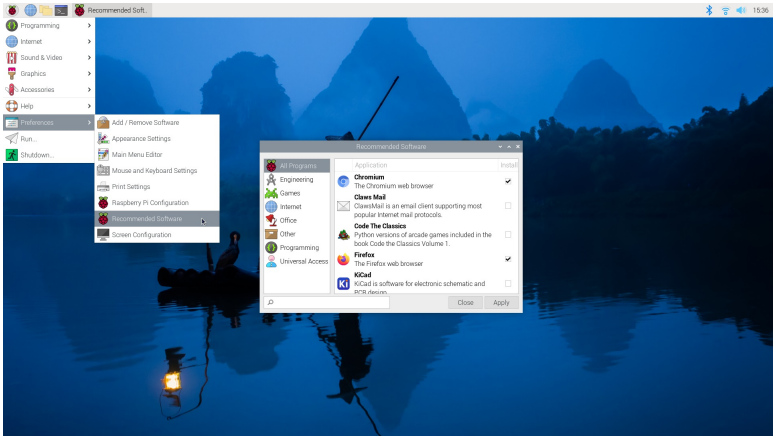
Mynd 15.30: Viltu uppfæra núna eða seina?



Mynd 15.31: Velja Wi-FI



Mynd 15.32: Endursæum til að klára



Mynd 15.33: Velja Wi-FI

16 GPIO

GPIO er hugbúnaðarforrit (klasi) sem sér um samskipti inn- og útganga Raspberry Pi við örgjörvann. Það þarf að setja hann upp sérstaklega ef ætlunin er að nota tölvuna til að stýra ljósum eða öðrum ytri tækjum.

Uppsetning

Innsetningin fer fram í skipanaglugganum með eftirfarandi skrefum (skipanir eru feitletraðar í kóðadæminu hér að neðan):

Náðu í GPIO pakkann af https://pypi.python.org/pypi/RPi.GPIO og vistaðu hann í `pi`-möppunni.

Opnaðu LX Terminal og keyrðu eftirfarandi skipanir:

```
sudo apt-get install python-dev
tar zxf nafn_skrarinnar
cd nafn_skrarinnar
sudo python setup.py install
```

Til að viðhalda stýrikerfinu með uppfærslum:

```
sudo apt-get update
```

Til að vísa í 'RPi.GPIO' sem bara 'GPIO':

```
import RPi.GPIO as GPIO
```

Það eru tvær leiðir til að vísa í pinnana í 40 pinna tenginu á tölvunni:
Vísa í pinnana í réttri númeraröð (1–40):

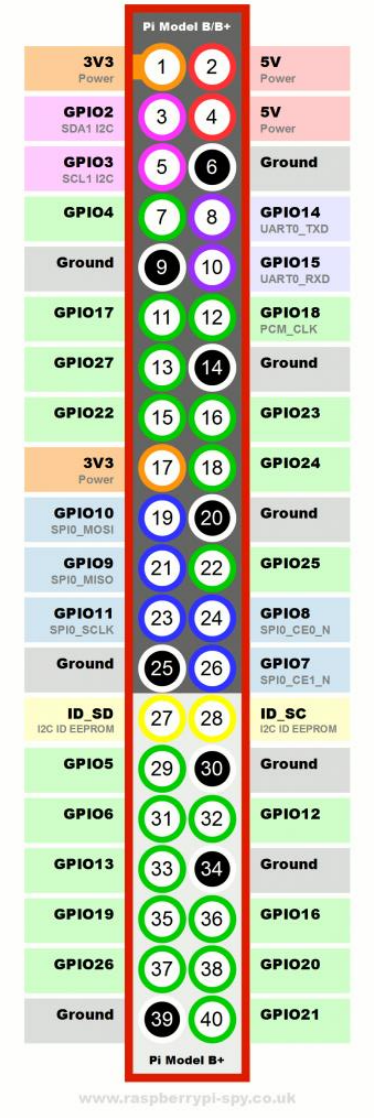
```
GPIO.setmode(GPIO.BOARD)
```

Vísa í pinnana eftir rásum:

```
GPIO.setmode(GPIO.BCM)
```

Við notum aðferðina 'GPIO.setmode(GPIO.BCM)' og notum eftirfarandi töflu til að sjá hvaða pinnum við tengjumst. Rásin 'GPIO17' tengist pinna 11 í 40 pinna tenginu, og 'GPIO18' tengist pinna 12.

myndinni um 90 gráður til hægri



Mynd 16.1: Samskipti Raspberry Pi við pinna.Heimild: ResearchGate.

17 Atburðarvöktun

Hægt er að láta forritið vakta ákveðinn inngang og framkvæma skipanir ef breyting verður á honum. Þannig getum við keyrt ákveðinn kóða í hvert sinn sem atburður á sér stað. Atburðarvöktun er yfirleitt keyrð í lykkjum Vöktunin er sett inn með skipuninni:

```
1 GPIO.add_event_detect(channel, GPIO.RISING)
```

Þessi skipun vaktar hvort ákveðinn inngangur rísi (0V uppí 3,3V)

Nota má EF (IF) skilyrði til að tékka hvort atburður hafi átt sér stað á ákveðnum inngangi og bregðast við því.

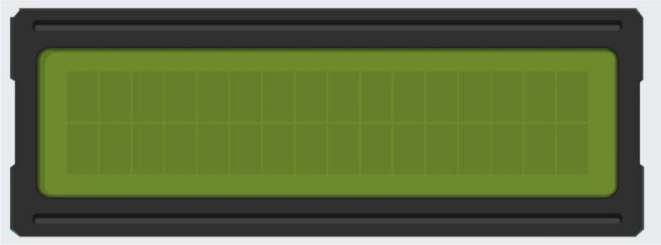
```
1 If GPIO.event_detected(channel, GPIO.RISING)
```

18 Stjórn Seguliða

Kóðadæmi

```
import RPi.GPIO as GPIO
import time

# define relay pin
relay_pin = 21

# set GPIO mode as GPIO.BOARD
GPIO.setmode(GPIO.BCM)
# setup relay pin as OUTPUT
GPIO.setup(relay_pin, GPIO.OUT)

# Open Relay
GPIO.output(relay_pin, GPIO.LOW)
# Wait half a second
time.sleep(0.5)
# Close Relay
GPIO.output(relay_pin, GPIO.HIGH)
# Wait half a second
time.sleep(0.5)
# Close Relay
GPIO.output(relay_pin, GPIO.LOW)
GPIO.cleanup()
```

19 2x16 LCD skjá



Mynd 19.1: Tilvintun vantar

LCD-skjárinn er einn af flóknustu hlutunum . Hann hefur áhrif bæði á hvernig hugbúnaðinum er stjórnað og hvernig skjárinn sjálfur virkar. Skjárinn hefur 16 litla pixla svið á 2 línum, þar sem hvert svið er 5x8 pixlar. Hverjum pixli er stjórnað af örstýringum til að sýna stafi, tölustafi og sérstafi. Þetta þýðir að hægt er að sýna allt að 32 stafi í einu án þess að þurfa að stjórna hverjum pixli fyrir sig.

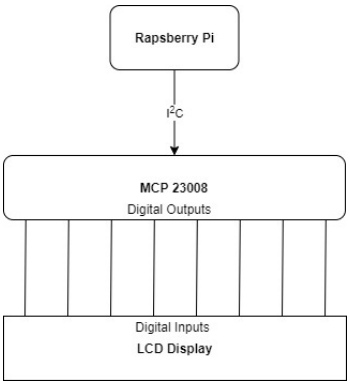
Til að auðvelda notkun eru mikilvægustu stafirnir, eins og A-Ö, 0-9 og nokkrir sérstafir, þegar forforritaðir. Notendur hafa einnig möguleika á að skrá 8 sérsniðna stafi til viðbótar. Þessi fjölbreytileiki og plássið til að birta texta og tölur gerir LCD-skjáinn mjög hentugan fyrir margvísleg forrit.

Skjánum er stjórnað með einföldum stafrænum merkjum. Þar sem Raspberry Pi hefur aðeins nokkur tiltæk stafræn inntak og úttak, er notuð I^2C eining til að bæta við fleiri stafrænum inntökum og úttökum. Þetta er sýnt á myndinni. Birtustig skjásins er stillt í með potentiometer. Þetta er hægt að stilla með skrúfjárni þar til skjárinn hefur bestu birtuskil og allt er auðvelt að lesa.

Prófum Hér að skrifa Hallo Wolrd og sjá hvort textin kemur á skjá.

```
# Example using a character LCD backpack.
import time
import Adafruit_CharLCD as LCD

# Define LCD column and row size for 16x2 LCD.
lcd_columns = 16
lcd_rows    = 2

# Initialize the LCD using the pins
lcd = LCD.Adafruit_CharLCDBackpack(address=0x21)

try:
    # Turn backlight on
    lcd.set_backlight(0)

    # Print a two line message
    lcd.message('Hello\nworld!')

    # Wait 5 seconds
    time.sleep(5.0)

    # Demo showing the cursor.
    lcd.clear()
    lcd.show_cursor(True)
    lcd.message('Show cursor')

    time.sleep(5.0)

    # Demo showing the blinking cursor.
    lcd.clear()
    lcd.blink(True)
    lcd.message('Blink cursor')

    time.sleep(5.0)

    # Stop blinking and showing cursor.
    lcd.show_cursor(False)
    lcd.blink(False)

    # Demo scrolling message right/left.
    lcd.clear()
    message = 'Scroll'
    lcd.message(message)
    for i in range(lcd_columns-len(message)):
        time.sleep(0.5)
        lcd.move_right()
    for i in range(lcd_columns-len(message)):
        time.sleep(0.5)
        lcd.move_left()

    # Demo turning backlight off and on.
    lcd.clear()
    lcd.message('Flash backlight\nin 5 seconds...')
    time.sleep(5.0)
```

```
    # Turn backlight off.
    lcd.set_backlight(1)
    time.sleep(2.0)
    # Change message.
    lcd.clear()
    lcd.message('Goodbye!')
    # Turn backlight on.
    lcd.set_backlight(0)
    # Turn backlight off.
    time.sleep(2.0)
    lcd.clear()
    lcd.set_backlight(1)
except KeyboardInterrupt:
    # Turn the screen off
    lcd.clear()
    lcd.set_backlight(1)
```



Mynd 19.2: Tilvintun vantar

20 Forrit

20.1 Forrit 1

Við tengjum ljósdíóðuna við GPIO18 sem er pinni 12 í 40 pinna tenginu. Við notum engan inngang til að starta þessu heldur gerir forritið það sjálft. Venjuleg 5mm rauð ljósdíóða þarf c.a 1,8V til að kvikna. Útgangurinn sendir út 3,3V sem er of mikið fyrir díóðuna svo við þurfum að setja inn viðnám á milli útgangs og díóðu til að fella spennuna úr 3,3V í 1,8V eða um 1,5V. Díóðan tekur 10mA (0,010A) straum og við reiknum stærð viðnámsins með:

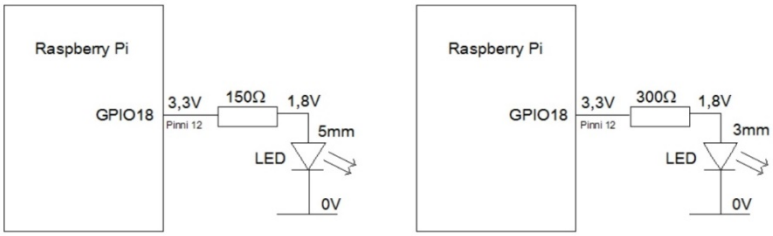
$$U = (3.3V - 1.8V) = 1.5V, \quad I = 0.010A$$

Setjum gildi inn í jöfnuna:

$$R = \frac{1.5V}{0.010A} = 150\,\Omega$$

Ef notuð er 3 mm LED má gera ráð fyrir 5 mA straumi (0,005 A) og viðnám $R = 300\,\Omega$.

Tengingin verður þá eftirfarandi:



Mynd 20.1: Eiríkur Guðmundsson — TNT 303

Forritskóðinn er eftirfarandi:

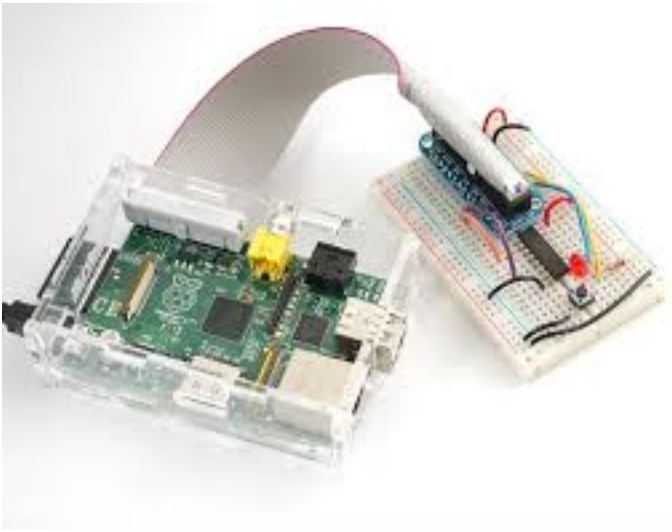
Athugið: `#` táknar athugasemd (e. comment) í Python. Línur sem byrja á `#` eru ekki keyrðar með forritinu, heldur eru þær settar inn sem hjálpartæki fyrir forritarann og til upplýsingar fyrir aðra sem lesa kóðann.

Aðeins textinn sem er **ekki** með # í byrjun er keyrður (RUN).

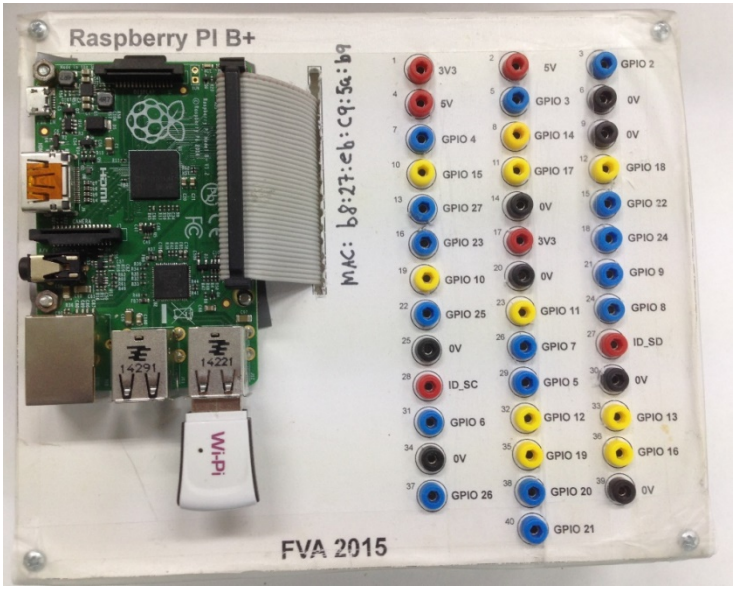
```
# ------------- Forrit 1 ---------------------------------------
#-------------- Kveikt á díóðu ----------------------------------
#---------- Raspberry Pi  Hardware Projects eftir Dogan Ibrahim --
#  Þetta forrit kveikir á díóðu
Import RPi.GPIO as GPIO      # til að geta notað GPIO í stað RPi.GPIO
GPIO.setmode(GPIO.BCM)       # Ákveðum hvaða númeringu við notum
    fyrir I/O
GPIO.setup(18, GPIO.OUT)     # Rás 18 (pinni 12) er útgangur
GPIO.output(18, 1)       # Rás 18 (Pinni 12)er sett í on (1). Díóða
    lýsir
GPIO.cleanup()  # Hreinsum til eftir okkur (Þetta lifir ekki, vantar
    tíma
```

Þessa tengingu er auðvelt að framkvæma á svokölluðu brauðbretti. Hægt er að tengja Raspberry Pi auðveldlega við svona brauðbretti með 40 pinna borða svipað og við finnum í gömlum tölvum og eru t.d. notaðir til að tengja geisladrif við móðurborð tölvu. Svona kit er hægt að kaupa á netinu.

Uppsetningu eins og á myndinni hér að ofan er meira mál að búa til en auðveldar allar tengingar og gerir þær sýnilegri. Það getur verið flókið að rekja tengingar á brauðborði. Hún er því tilvalin í kennslu. Það er líka þægilegt í kennslu að vera með tilbúnar inn og útgangse iningar.



Mynd 20.2: Hægt að kaupa svona set á neitnu



Mynd 20.3: Smíðað í rafdeilt við fjölbrautaskóla vesturlands

20.2 Forrit 2

Annað forritið okkar er einfalt. Það lætur ljósdíóðu (LED) blikka 10 sinnum með 1 sek millibili . Forritið sem keyrir þetta verkefni lítur svona út.

```
# ---------------------- Forrit 2 -------------------------------
#------------------- Blikkandi díóða -----------------------------
#---- Raspberry Pi  Hardware Projects eftir Dogan Ibrahim -------
#  Þetta forrit lætur díóðu blikka 10 sinnum með 1 sek millibili
import RPi.GPIO as GPIO      # til að geta notað GPIO í stað RPi.GPIO
import time          # Náum í fall sem sér um tímatalningu fyrir okkur
GPIO.setmode(GPIO.BCM)       # Ákveðum hvaða númeringu við notum
    fyrir I/O
GPIO.setup(18, GPIO.OUT)     # Rás 18 (pinni 12) er útgangur

for counter in range (0,10):         # counter er teljari fyrir
    lykkju sem
keyrir 10 skipti
    GPIO.output(18, 1)                   # Rás 18 (Pinni 12)er sett í
        on (1).
Inndráttur 4 bil
    time.sleep(1)        # Bíðum í 1 sek. Rás 18 er enn í on
    GPIO.output(18, 0)       # Rás 18 (Pinni 12)er sett í off (0).
    time.sleep(1)        # Bíðum í 1 sek. Rás 18 er enn í off
GPIO.cleanup()       # Hreinsum til
```

Allt sem er dregið inn um 4 bil á eftir for skipuninni tilheyrir for lykkjunni . Hún keyrir 10 sinnum. Í hvert skipti kviknar og slokknar á díóðunni einu sinni. Við getum líka notað breytur í forritinu til að gera það skiljanlegra.

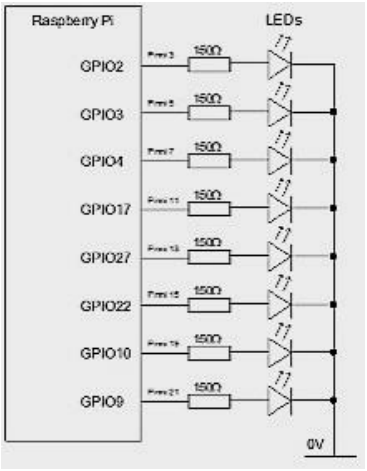
```
# ---------------Forrit 2 með breytum-------------------------------
#----------------- Blikkandi díóða -----------------------------
#---- Raspberry Pi  Hardware Projects eftir Dogan Ibrahim ----------
#  Þetta forrit lætur díóðu blikka 10 sinnum með 1 sek millibili
Import RPi.GPIO as GPIO      # til að geta notað GPIO í stað RPi.GPIO
Import time          # Náum í fall sem sér um tímatalningu fyrir okkur
LED = 18             # Skilgreinum töluna 18 sem LED. LED er breyta
ON = 1           # Skilgreinum töluna 1 sem ON. ON er breyta
OFF = 0          # Skilgreinum töluna 0 sem OFF. OFF er breyta
GPIO.setmode(GPIO.BCM)       # Ákveðum hvaða númeringu við notum
    fyrir I/O
GPIO.setup(LED, GPIO.OUT)    # Rás 18 (pinni 12) er útgangur

for counter in range (0,10):     # counter er teljari fyrir lykkju
    sem keyrir
10 skipti
    GPIO.output(LED, ON)# Rás 18 (Pinni 12)er sett í on (1).
        Inndráttur 4 il
    time.sleep(1)        # Bíðum í 1 sek. Rás 18 er enn í on
    GPIO.output(LED, OFF)    # Rás 18 (Pinni 12)er sett í off (0).
    time.sleep(1)        # Bíðum í 1 sek. Rás 18 er enn í off
GPIO.cleanup()       # Hreinsum til
```

Forritið er læsilegra svona.

20.3 Forrit 3

Í næsta verkefni ætlum við að tengja 8 díóður við tölvuna og láta þær blikka í röð, neðan frá og upp.Tengingin er svona.

```
#---------------------------  Forrit 3 ----------------------
#------------------------ 8 blikkandi LED --------------------
#------- Raspberry Pi  Hardware Projects eftir Dogan Ibrahim --
Import RPi.GPIO as GPIO
Import time
# Búum til grúppu sem inniheldur BCM rásirnar.
# Þannig getum við vísað í númer díóðunnar sem á að vera
# kveikt í stað þess að vera með 8 útganga skilgreina í
# hverri umferð lykkjunnar
# LEDs[1] = 9, LEDs[2] = 10  o.s.frv.
# Röðin er sú sama og í tengingunum á myndinni
# 9 er neðsta díóðan og 2 sú efsta
LEDs = [9,10,22,27,17,4,3,2]
ON = 1
OFF = 0
GPIO.setmode(GPIO.BCM)
for i in range (0,8):
    GPIO.setup(Leds[i], GPIO.OUT) # skilgreinum pinnana sem útganga
# Kveikjum og slökkvum á díóðunum 10 sinnum með 1 sek. millibili
for i in range (0,10): # keyrum 10 sinnum í gegnum lykkjuna
    For j in range (0,8):           # tvöföld lykkja
        GPIO.output(LEDs[j], ON)
    time.sleep(1)
    For j in range (0,8):
        GPIO.output(LEDs[j], OFF)
    time.sleep(1)
```



Mynd 20.4: Eiríkur Guðmundsson TNT 303

20.4 Forrit 4 Umferðaljós

```
"""
This Raspberry Pi Pico MicroPython code was developed by newbiely.com
This Raspberry Pi Pico code is made available for public use without any restriction
For comprehensive instructions and wiring diagrams, please visit:
https://newbiely.com/tutorials/raspberry-pico/raspberry-pi-pico-traffic-ligh
"""

import machine
import time

# Define pin numbers (you can change these to match your wiring)
PIN_RED = 19    # GPIO19 on Raspberry Pi Pico
PIN_YELLOW = 21 # GPIO21 on Raspberry Pi Pico
PIN_GREEN = 22  # GPIO22 on Raspberry Pi Pico

# Define times in seconds (MicroPython uses seconds for time.sleep)
RED_TIME = 4    # RED time in seconds
YELLOW_TIME = 4 # YELLOW time in seconds
GREEN_TIME = 4  # GREEN time in seconds

# Setup pins as output
red = machine.Pin(PIN_RED, machine.Pin.OUT)
yellow = machine.Pin(PIN_YELLOW, machine.Pin.OUT)
green = machine.Pin(PIN_GREEN, machine.Pin.OUT)

# Main loop
while True:
    # Red light on
    red.value(1)    # turn on red
    yellow.value(0) # turn off yellow
    green.value(0)  # turn off green
    time.sleep(RED_TIME) # keep red light on for the defined period

    # Yellow light on
    red.value(0)    # turn off red
    yellow.value(1) # turn on yellow
    green.value(0)  # turn off green
    time.sleep(YELLOW_TIME) # keep yellow light on for the defined period

    # Green light on
    red.value(0)    # turn off red
    yellow.value(0) # turn off yellow
    green.value(1)  # turn on green
    time.sleep(GREEN_TIME) # keep green light on for the defined period
```

20.5 Skynjar fjarðlægð

```
from machine import Pin, Timer
import time

# Define the pins for the trigger and echo
trig = Pin(1, Pin.OUT) # The Raspberry Pi Pico pin (GP1) connected to TRIG pin of the ultrasonic sensor
echo = Pin(0, Pin.IN)  # The Raspberry Pi Pico pin (GP0) connected to ECHO pin of the ultrasonic sensor

def read_distance():
    # Ensure the trigger pin is low for a clean pulse
    trig.value(0)
    time.sleep_us(2)

    # Send a 10 microsecond pulse to start the measurement
    trig.value(1)
    time.sleep_us(10)
    trig.value(0)

    # Measure the duration of the echo pulse
    while echo.value() == 0:
        signal_off = time.ticks_us()
    while echo.value() == 1:
        signal_on = time.ticks_us()

    # Calculate the duration of the echo pulse
    time_passed = signal_on - signal_off

    # Calculate the distance in centimeters
    distance = (time_passed * 0.0343) / 2

    return distance

# Check the sensor every 1 seconds
while True:
    distance = read_distance()
    print('Distance:', distance, 'cm')
    time.sleep(1)
```

21 LED warning flash codes

Ef Raspberry Pi tekst ekki að ræsa sig af einhverjum ástæðum, eða þarf að slökkva á honum, mun LED í mörgum tilfellum blikka ákveðinn fjölda sinnum til að gefa til kynna hvað gerðist. Ljósdíóðan mun blikka í nokkur löng blikk (0 eða fleiri) og gefa síðan stutta blikka til að gefa til kynna nákvæma stöðu. Í flestum tilfellum mun mynstrið endurtaka sig eftir tveggja sekúndna bil.

Long flashes	**Short flashes**	**Status**
0	3	Generic failure to boot
0	4	start*.elf not found
0	7	Kernel image not found
0	8	SDRAM failure
0	9	Insufficient SDRAM
0	10	In HALT state
2	1	Partition not FAT
2	2	Failed to read from partition
2	3	Extended partition not FAT
2	4	File signature/hash mismatch - Pi 4 and Pi 5
3	1	SPI EEPROM error - Pi 4 and Pi 5
3	2	SPI EEPROM is write protected - Pi 4 and Pi 5
3	3	I2C error - Pi 4 and Pi 5
3	4	Secure-boot configuration is not valid
4	3	RP1 not found
4	4	Unsupported board type
4	5	Fatal firmware error
4	6	Power failure type A
4	7	Power failure type B

Tafla 21.1: Raspberry Pi Boot Error Codes

22 Verkefni í python

22.1 Breytur (strengur , heiltölur, kommutölur)

Í þessu verkefni eigið þið að skrifa lítið forrit sem spyr um 2 hluti og skrifar þá út. Breytur (strengur , heiltölur, kommutölur) – innsláttur

1.1 Nafn og aldur. Forritið spyr um nafn og aldur notandans, svo er það sett inn í setningu:

```
nafn = input("Hvað heitirðu? ")
aldur = input("Hvað ertu gamall/gömul? ")
print(f"Velkomin/n á Python námskeið {nafn}, {aldur} ára.")
```

1.2 Þrjár tölur – summur, margfeldi, og frádráttur Forritið biður notandann um að slá inn þrjár tölur og prentar síðan út summu, margfeldi og frádrátt talnanna:

```
tala1 = int(input("Sláðu inn tölu1: "))
tala2 = int(input("Sláðu inn tölu2: "))
tala3 = int(input("Sláðu inn tölu3: "))
summa = tala1 + tala2 + tala3
margfeldi = tala1 * tala2 * tala3
frádráttur = tala1 - tala2 - tala3
print(f"Fyrsta talan er: {tala1}")
print(f"Önnur talan er: {tala2}")
print(f"Þriðja talan er: {tala3}")
print(f"Summa talnanna er: {summa}")
print(f"Margfeldni talnanna er: {margfeldi}")
print(f"Frádráttur talnanna er: {frádráttur}")
```

1.3 Rúmmál kassa Forritið biður um þrjár hliðarlengdir og reiknar rúmmál kassans:

```
hæð = int(input("Sláðu inn hæð kassans: "))
lengd = int(input("Sláðu inn lengd kassans: "))
breidd = int(input("Sláðu inn breidd kassans: "))
rúmmál = hæð * lengd * breidd
print(f"Rúmmál kassans er: {rúmmál}")
```

1.4 Breyta Celsíus í Fahrenheit Forritið tekur hitastig í gráðum á Celsíus og reiknar það í Fahrenheit:

```
celsius = float(input("Sláðu inn hitastig í Celsíus: "))
fahrenheit = (1.8 * celsius) + 32
print(f"Hitastig í Fahrenheit er: {fahrenheit}")
```

1.5 Samanburður á tveimur tölum Forritið biður um tvær tölur og ber þær saman:

```

tala1 = int(input("Sláðu inn tölu1: "))
tala2 = int(input("Sláðu inn tölu2: "))
if tala1 > tala2:
    print("Fyrsta talan er stærri en önnur talan.")
elif tala1 < tala2:
    print("Fyrsta talan er minni en önnur talan.")
else:
    print("Tölurnar eru jafnar.")
```

1.6 Klukkustundir og mínútur Forritið breytir heildarfjölda klukkustunda í klukkustundir og mínútur:

```

klukkustundir = int(input("Sláðu inn fjölda klukkustunda: "))
stundir = klukkustundir // 60
minutur = klukkustundir % 60
print(f"{klukkustundir} klukkustundir jafngilda {stundir}
    klukkustundum og {minutur}
mínútum.")
```

1.7 Dagar í ár, mánuði og daga Forritið breytir fjölda daga í ár, mánuði og daga:

```

dagar = int(input("Sláðu inn fjölda daga: "))
ar = dagar // 365
mánuðir = (dagar % 365) // 30
afgangs_dagar = (dagar % 365) % 30

print(f"{dagar} dagar jafngilda {ar} ári, {mánuðir} mánuðum og
    {afgangs_dagar}
dögum.")
```

1.8 Aldursflokkar

Forritið biður um aldur og flokkar notandann í aldurshóp:

```
aldur = int(input("Sláðu inn aldur: "))
if 13 <= aldur <= 19:
    print("Þú ert unglingur.")
elif 20 <= aldur <= 64:
    print("Þú ert fullorðinn.")
elif aldur >= 65:
    print("Þú ert eldri borgari.")
else:
    print("Þú ert barn.")Sláðu inn tölu2: 5
```

1.8 Aldursflokkar Forritið biður um aldur frá notanda og prentar út hvort viðkomandi sé unglingur (13-19 ára), fullorðinn (20-64 ára) eða eldri borgari (65 ára eða eldri).

```
aldur = int(input("Sláðu inn aldur: "))
if 13 <= aldur <= 19:
    print("Þú ert unglingur.")
elif 20 <= aldur <= 64:
    print("Þú ert fullorðinn.")
else:
    print("Þú ert eldri borgari.")
```

1.9 Prósenta réttra svara Forritið biður notanda um fjölda réttra svara og heildarfjölda prófatriða og reiknar prósentu réttra svara.

```
rett_svor = int(input("Sláðu inn fjölda réttra svara: "))
heildar_fjoldi = int(input("Sláðu inn heildarfjölda prófatriða: "))
prosent = (rett_svor / heildar_fjoldi) * 100
print(f"Þú fékkst {prosent}% rétt úr prófinu.")
```

1.10 Margfeldi af annarri tölu Forritið biður um tvær heiltölur og prentar hvort önnur þeirra sé margfeldi af hinni.

```
tala1 = int(input("Sláðu inn tölu1: "))
tala2 = int(input("Sláðu inn tölu2: "))
if tala1 % tala2 == 0:
    print(f"{tala1} er margfeldi af {tala2}.")
else:
    print(f"{tala1} er ekki margfeldi af {tala2}.")
```

1.11 Matarinnkaup – krónur í evrur

Forritið biður um upphæð matarinnkaupa í íslenskum krónum og reiknar kostnaðinn í evrum (1 evra = 150 ISK).

```
isk = float(input("Sláðu inn upphæð matarinnkaupa í ISK: "))
eur = isk / 150
print(f"Kostnaðurinn í evrum er {eur} EUR.")
```

1.12 Meðaltal tveggja talna Forritið biður um tvær heiltölur og reiknar út meðaltal þeirra.

```
tala1 = int(input("Sláðu inn fyrri tölu: "))
tala2 = int(input("Sláðu inn seinni tölu: "))
medaltal = (tala1 + tala2) / 2
print(f"Meðaltal talnanna {tala1} og {tala2} er {medaltal}.")
```

1.13 Flatarmál rétthyrnings Forritið biður um lengd og breidd rétthyrnings og reiknar flatarmál hans.

```
lengd = int(input("Sláðu inn lengd rétthyrningsins: "))
breidd = int(input("Sláðu inn breidd rétthyrningsins: "))
flatarmal = lengd * breidd
print(f"Flatarmál rétthyrningsins er: {flatarmal}")
```

1.14 Flatarmál hrings

Forritið biður um geisla hrings og reiknar flatarmál hans.

Formúlan er: `flatarmál = π * radius^2`

Dæmi: Reikna flatarmál hrings

```
import math
radius = float(input("Sláðu inn geisla hringsins: "))
flatarmal = math.pi * radius ** 2
print(f"Flatarmál hringsins er: {flatarmal}")
```

22.2 Skilyrðissetningar (if – else)

2.1Forritið á að biðja um tölu Ef talan er stærri en 100 á að prentast: „Talan er stærri en 100“. Annars á að prentast: „Talan er 100 eða minni“.

2.2 Forritið umbreytir tommum í sentimetra eða öfugt
Forritið spyr hvort umbreyta eigi tommum í sentimetra eða sentimetrum í tommur. Forritið spyr síðan um aðra stærðina en reiknar hina. Í einni tommu eru 2.54 sentimetrar.

2.3 Forritið spyr um tvær heiltölur Forritið skrifar síðan hvor talan er minni. Ef tvisvar er slegin inn sama talan á forritið að skrifa: “Tölurnar eru jafn stórar”.

2.4 Forritið spyr um númer mánaðar Forritið skrifar síðan út hvað mánuðurinn heitir. Ef slegin er inn tala sem er 0 eða minni eða 13 og hærri á að skrifast út: “Rangur innsláttur”.

2.5 Forritið spyr um þrjár heiltölur. Það má ekki slá inn sömu töluna tvisvar sinnum. Forritið skrifar síðan hver talnanna er minnst.

2.6
Notið sama dæmi og í lið a í verkefni 1. Spyrjið nú einnig um kyn notanda (kk eða kvk). Ef notandi svarar kvk á að prentast: Velkomin á forritunarnámskeið [nafn], [aldur] ára. Ef notandi svarar kk á að prentast: Velkominn á forritunarnámskeið [nafn], [aldur] ára. Ef notandi svarar hvorki kk né kvk á að prentast „Rangur innsláttur“.

22.3 Lykkjur (for/while)

3.1 Notið `for` lykkju til að prenta allar tölur á bilinu 1 til 50 (báðar meðtaldar).

3.2 Forritið les inn tvær heiltölur frá lyklaborði og prentar út á skjáinn allar tölur á bilinu á milli talnanna. Ef tölurnar eru jafnháar eða önnur einum hærri en hin, birtið þá villuboð um að engar tölur séu á bilinu.

3.3 Forritið biður notanda um að slá inn upphafstölu og lokatölu. Forritið birtir allar oddatölur frá upphafstölu til og með lokatölu. T.d. ef notandi slær inn töluna 2 og 7, birtast tölurnar: 3, 5, 7.

3.4 Búið til forrit sem leggur saman allar tölur á ákveðnu bili. T.d. ef bilið er frá 5 upp í 11 á niðurstaðan að vera $5 + 6 + 7 + 8 + 9 + 10 + 11 = 56$. Notandinn á að ákveða hvaða bil er valið.

3.5 Búið til forrit sem skrifar út eina línu í margföldunartöflu. Forritið spyr hvaða línu eigi að skrifa. Ef t.d. er beðið um línu 3, ætti að skrifast:

```
3 sinnum taflan:
3 x 1 = 3
3 x 2 = 6
3 x 3 = 9
3 x 4 = 12
3 x 5 = 15
3 x 6 = 18
3 x 7 = 21
3 x 8 = 24
3 x 9 = 27
3 x 10 = 30
```

3.6 Forritið reiknar flatarmál þríhyrnings. Forritið spyr um grunnlínu og hæð þríhyrnings og skrifar síðan út flatarmál þríhyrningsins:

$$\text{Flatarmál} = \frac{\text{grunnlína} \times \text{hæð}}{2}.$$

Forritið spyr hvort endurtaka eigi þetta eða ekki og endurtekur þar til notandinn ákveður að hætta (notið `while` lykkju).

3.7 Forritið biður notanda um heiltölu n. Það reiknar út summu allra talna frá 1 til n. Dæmi: Ef notandi slær inn 5, þá reiknar forritið:

$$1 + 2 + 3 + 4 + 5 = 15.$$

3.8 Búið til forrit sem prentar út allar tölur frá 1 til 100 sem eru margfeldi af 3 en ekki af 5. Dæmi: 3, 6, 9, 12, ...

3.9 Forritið les inn heiltölu n og prentar út alla deilendur n, þ.e. allar tölur sem ganga upp í n. Dæmi: Ef $n = 12$, þá ætti forritið að skrifa: 1, 2, 3, 4, 6, 12.

3.10 Forritið biður notanda um tvær heiltölur og reiknar margfeldi þeirra. Ef önnur talan eða báðar eru neikvæðar, skrifar forritið að ekki sé hægt að reikna margfeldi neikvæðra talna.

3.11 Búið til forrit sem les heiltölu frá notanda og prentar út margföldunartöflu frá 1 til 10 fyrir þessa tölu. Dæmi: Ef notandi slær inn 4, ætti forritið að prenta:

```
4 x 1 = 4
4 x 2 = 8
...
4 x 10 = 40
```

3.12 Forritið les inn heiltölu frá notanda og prentar út allar tölur frá 1 upp að talan sjálf. Forritið prentar út hvern fjórða staf í stað hvern staf. Dæmi: Ef notandi slær inn 12, þá birtist: 4, 8, 12.

3.13 Forritið les inn heiltölu n og prentar síðan stjörnur ($*$) í n línur, þar sem fyrsta línan inniheldur eina stjörnu og hver lína hefur eina stjörnu meira en sú fyrri. Dæmi: Ef $n = 5$, þá birtir forritið:

```
*
**
***
****
*****
```

3.14 Forritið les inn heiltölu frá notanda og reiknar út summu allra slétttalna á bilinu frá 1 til þessarar tölu. Dæmi: Ef notandi slær inn 6, þá ætti forritið að reikna út: $2 + 4 + 6 = 12$.

3.15 Búið til forrit sem biður notanda um heiltölu og prentar síðan út allar tölur sem eru margfeldi af 5 og eru minni en eða jafnar tölunni sem var slegin inn. Dæmi: Ef notandi slær inn 23, þá birtast tölurnar: 5, 10, 15, 20.

3.16 Forritið les inn heiltölu frá notanda og prentar síðan fjölda skrefa til að komast á 0 með því að draga 1 frá tölunni fyrir hvert skref þar til talan er orðin 0. Ef talan er neikvæð, birtir forritið að talan sé nú þegar minni en 0.

22.4 Listar - random tölur

4.1 Búðu til lista sem inniheldur 5 heiltölur (sem þú velur). Prentaðu út tölu nr. 3 í listanum.

4.2 Notaðu random.randomint() til að búa til random tölu á bilinu 1-6 (eins og 6-hliða teningi hafi verið kastað). Prentaðu töluna.

4.3 Settu 5 random tölur á bilinu 1-6 í lista (for lykkja sem fer 5 umferðir). Notaðu append() aðferðina til að bæta tölunum við listann. Prentaðu listann eina tölu í einu.

4.4 Skrá skal nöfn keppenda í víðavangshlaupi. Fyrst er spurt um fjölda keppenda, þá eru nöfnin tekin inn í lista (með for lykkju), að lokum skal birta nöfn keppenda í réttri stafrófsröð.

4.5 Líkja á eftir því að tveimur teningum, teningur1 og teningur2, sé kastað fimmtíu sinnum. Það er tilviljunarkennt hvað kemur upp í hverju kasti en talan þarf samt að vera á bilinu 1-6. Summa teninganna í hverju kasti er síðan geymd í listanum kast. Eftir að öll köst eru búin skal birta:

1. Listann í heild sinni.
2. Listann í heild sinni raðaðan frá lægstu að hæstu tölu.
3. Hver heildarsumman er í listanum.
4. Hvert meðaltal kastanna er í listanum.

22.5 5. Functions - dictionaries

5.1

Búið til fall sem tekur tölu og setur hana í annað veldi. Munið að keyra fallið eftir að búið er að skilgreina það.

```
def annadVeldi(tala):
    tala_2 = tala * tala
    # print(tala_2)
    return tala_2

tala = input("Sláðu inn tölu: ")
tala = float(tala)

veldi = annadVeldi(tala)
print(tala, "í öðru veldi er:", veldi)
```

5.2

Búið til fall sem tekur tölu og setur hana í tiltekið veldi. Fallið á að taka tvo stika (töluna og veldið), sem eru gefnir sem viðföng (arguments) þegar fallið er keyrt.

```
def Veldi(tala, veldi):
    tala_veldi = pow(tala, veldi)
    return tala_veldi

tala = input("Sláðu inn veldisstofn: ")
tala = int(tala)

veldi = input("Sláðu inn veldisvísi: ")
veldi = int(veldi)

utkoma_veldi = Veldi(tala, veldi)
print(tala, "í veldinu", veldi, "er:", utkoma_veldi)
```

`https://www.tutorialspoint.com/python3/python_dictionary.htm`

5.3

Búið til `dictionary` með nokkrum lykil-gildis pörum (*key-value pairs*). Birtið innihaldið í skjáborðinu (console) á forminu: `lykill -> gildi` í lóðréttum dálki.

```
d={'nafn':'Eiríkur','aldur':'56','kyn':'kk','starf':'kennari'}
#d.keys()   Sýnir alla lyklana í d. ['nafn','aldur','kyn','starf']
#d.values() Sýnir öll gildin í d. ['Eiríkur','56','kk','kennari']
#d.items()  Sýnir öll pörin í d. [('nafn','Eiríkur'.....)
print("Aðferð 1")
for x in d.keys():
    print(x +" => " + d[x])
print("Aðferð 2")
for key,value in d.items():
    print(key + " => " + value)
#You can get keys using iter
```

```
12 #>>> list(iter(d)) #['a', 'b']
```

5.4 Búið til dictionary sem hefur bókstaf sem key og heiltölu sem value. Má vera enska stafrófið. Dæmi: "a":1,"b":2,"c":3,"d":4,.....

Hannið forrit sem biður um nafn og skrifar út hvaða tölustafir eru á bak við bókstafina. Dæmi: karl verður 11 1 18 12

Forritið á einnig að geta lagt saman tölurnar og skrifað út summu þeirra. Dæmi: karl gefur summuna 42

Forritið á að geta beðið um tvö nöfn og sagt til um hvort nafnið hefur hærri summu.

5.5

Endurtakið forritið frá því í lið d en notið núna föll (functions) til að reikna út summurnar og til að birta tölurnar fyrir hvert nafn.

5.6 Búið til fall sem hefur sömu virkni og forritið í lið e í æfingu 4. Fallið á að taka þrjú viðföng (parameters) – hve mörgum tenginum er kastað í hvert sinn, hversu oft þeim er kastað og hve margar hliðar hver teningur hefur (gerum ráð fyrir að í hvert sinn séu allir teningarnir með jafn margar hliðar).

Búið síðan til fjögur föll sem hafa eftirfarandi virkni:

- Birtir listann í heild sinni raðaðan frá lægstu að hæstu tölu.
- Reiknar hver heildarsumman er í listanum.
- Reiknar hvert meðaltal kastanna er í listanum.

22.6 6. Reading/writing files

6.1 Í stað þess að búa til dictionary í kóðanum á að nota meðfylgjandi .json skrá. Lesið inn skrána með JSON aðferðum Python (.load()).

Búið til textaskrá (.txt) sem inniheldur einhvern texta að eigin vali. Lesið inn innihald skrárnar sem streng og reiknið summu textans á svipaðan hátt og í æfingu 5.

6.2

Búið til tóma textaskrá og skrifið inn í hana allar sléttar tölur frá 1 til 1000 (gert með for lykkju) með Python aðferðum.

1. Lesið skrána og setjið lista. Prentið út listann.
2. Finnið summu og meðaltal talnanna með tveimur aukastöfum.
3. Finnið allar tölur sem 8 gengur upp í og setjið í nýja textaskrá.
4. Prentið út upprunalegu skrána með bil milli talna og 10 tölur í línu.

```
listi = line.split('  ')       # Skiptir línunni á tvöföldu bili og
    setur í lista
listi.remove(' ')              # Fjarlægir tóm stök úr listanum
listi = list(map(int, listi)) # Breytir strengjalista í heiltölulista
```

6.3 Búið til tóma textaskrá sem innheldur prímtölurnar upp að 100. Setjið tölurnar inn með því að nota for lykkju.

1. Prentið út skrána.
2. Prentið allar tölur sem innihalda töluna 7.
3. Prentið út fjórðu hverju tölu og setjið í nýja textaskrá. Prentið út þá skrá.

23 Skynjarar

7.1 Kveikja og slökkva á LED með Raspberry Pi

Verkefni eftir: *Pétur Örn Sigurðsson*

Þú þarft:

- Raspberry Pi tölvu
- LED ljós
- 220 ohm viðnám

Skref 1: Tengja LED við Raspberry Pi
Settu mínus (-) á LED í 1B og plús (+) í 2B á prentplötu. Hliðtengdu viðnámið í plúsinn (A2) og hinn endann í -5. Tengdu jumpervíra frá Raspberry Pi:

- 5V í neðsta plús (Pin 2)
- GND í neðsta mínus (Pin 9)
- GPIO 17 í D1 (Pin 11) sem tengist LED mínus

Skref 2: Uppfæra pakkalista

```
sudo apt update
```

Skref 3: Setja upp gpiozero bókasafn

```
sudo apt install python3-gpiozero
```

Skref 4: Búa til forrit Farðu í möppuna:

```
cd Documents
```

Búðu til skrá:

```
sudo nano blikkled.py
```

Settu inn eftirfarandi kóða:

```
1
2       from gpiozero import LED
3       from time import sleep
4
5       led = LED(17)
6
7       try:
8           while True:
9               led.on()
10              print("LED ON")
11              sleep(1)
12              led.off()
13              print("LED OFF")
14              sleep(1)
15
16      except KeyboardInterrupt:
17          print("Stopp!")
```

Skref 5: Keyra forritið

```
1 python3 blikkled.py
```

LED ætti nú að blikka á 1 sekúndu fresti.

Skref 6: Stöðva forritið

Ýttu á `Ctrl + C` til að stoppa. Þá birtist:

```
1 Stopp!
```

```
from gpiozero import LED      # Flytur inn LED klasa úr gpiozero bókasafni
from time import sleep        # Flytur inn sleep fall úr time bókasafni til að bíða milli skipana

led = LED(17)                 # Býr til LED hlut sem tengist GPIO pinna 17 (physical pin 11)

try:                          # Reynir að keyra kóðann hér fyrir neðan
    while True:               # Byrjar endalausa lykkju sem keyrir aftur og aftur
        led.on()              # Kveikir á LED-inu
        print("LED ON")       # Skrifar „LED ON“ í skjáinn
        sleep(1)              # Bíð í 1 sekúndu
        led.off()             # Slekkur á LED-inu
        print("LED OFF")      # Skrifar „LED OFF“ í skjáinn
        sleep(1)              # Bíð í 1 sekúndu

except KeyboardInterrupt:     # Ef notandi ýtir á Ctrl + C til að stoppa forritið
    print("Stopp!")           # Skrifar „Stopp!“ í skjáinn og hættir forritinu
```

Mynd 23.1: Kóði á skjá

Stjórnun á LED með Flask og Raspberry Pi

Í þessu verkefni ætlum við að:

- Setja upp vefþjón með Flask
- Stjórna LED ljósi eða öðrum rofa með GPIO-pinna
- Búa til einfalda heimasíðu sem uppfærir sig sjálfkrafa þegar notandi ýtir á hnappa

Nauðsynleg forrit og pakkar

Byrjum á því að setja upp þau forrit og pakkar sem þarf. Flask – til að búa til heimasíðuna

```
sudo apt-get install python3-flask
```

GPIO stuðningur – til að stjórna pinnum á Raspberry Pi

```
sudo apt install python3-rpi.gpio
```

GPIO Zero – einfaldari leið til að nota LED og aðra rafeindahluti

```
sudo apt install python3-gpiozero
```

Það er góð venja að búa til sérstaka möppu fyrir hvert verkefni:

```
mkdir led-styring
cd led-styring
```

Í þessari möppu munum við geyma tvær gerðir af skrám:

- `.py` skrár: forrit sem keyra Flask-þjón og stjórna LED ljósi
- `.html` skrár: heimasíðan sjálf með hnöppum og útliti

Stundum getur verið gagnlegt að endurræsa Raspberry Pi til að tryggja að nýuppsettir pakkar virki rétt:

```
sudo reboot
```

Til að keyra þetta verkefni þarftu að setja upp nauðsynlega pakka:

```
sudo apt-get install python3-flask
sudo apt install python3-rpi.gpio
sudo apt install python3-gpiozero
```

```
from flask import Flask, render_template_string, jsonify
import RPi.GPIO as GPIO

app = Flask(__name__)
PIN = 6  # GPIO pinni sem tengdur er við LED

# Stillum GPIO upp
GPIO.setmode(GPIO.BCM)
GPIO.setup(PIN, GPIO.OUT)
GPIO.output(PIN, GPIO.LOW)  # Byrjar með slökkt

@app.route('/')
def index():
    html = '''
    <!DOCTYPE html>
    <html>
    <head><meta charset="utf-8"><title>LED Rofi</title></head>
    <body>
      <h2>Rofi: LED Stýring</h2>
      <p>Staða: <span id="status">...</span></p>
      <button onclick="kveikja()">Kveikja</button>
      <button onclick="slokkva()">Slökkva</button>
      <script>
        function kveikja() {
          fetch("/on").then(() => uppfaera());
        }
        function slokkva() {
          fetch("/off").then(() => uppfaera());
        }
        function uppfaera() {
          fetch("/status").then(r => r.json()).then(data => {
            document.getElementById("status").innerText = data.state
                ? "KVEIKT" : "SLÖKT";
          });
        }
        uppfaera();
      </script>
    </body>
    </html>
    '''
    return render_template_string(html)

@app.route('/status')
def status():
    return jsonify({'state': GPIO.input(PIN) == GPIO.HIGH})

@app.route('/on')
def on():
    GPIO.output(PIN, GPIO.HIGH)
    return ('', 204)
```

```
50
51 @app.route('/off')
52 def off():
53     GPIO.output(PIN, GPIO.LOW)
54     return ('', 204)
55
56 if __name__ == '__main__':
57     try:
58         app.run(host='0.0.0.0', port=5050)
59     finally:
60         GPIO.cleanup()
```

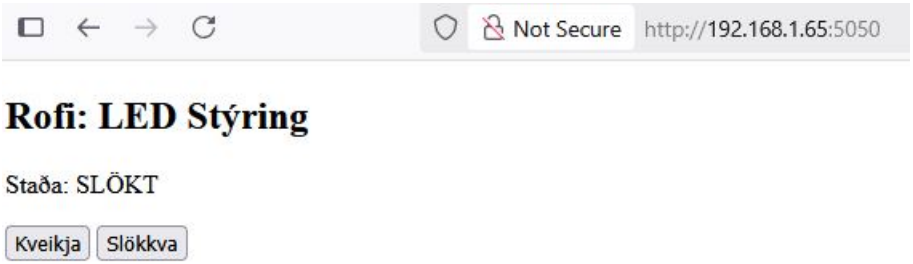
Keyrðu forritið með:

```
1 python3 app.py
```

Farðu síðan í vafra og opnaðu slóðina:

`http://raspberrypi.local:5050`

Eða notaðu IP tölu Raspberry Pi.



Mynd 23.2: útkona í vafra

24 Umferðarljós með LED-ljósdíóðum

Þetta verkefni er byggt á efni úr bókinni [1].

Í þessu verkefni lærirðu að kveikja og slökkva á mörgum LED-ljósum með því að búa til umferðarljósaröð (rauður–gulur–grænn). Þú getur líka bætt við takka síðan ef við viljum, en í þessu verkefni einbeitum við okkur eingöngu að ljósunum.

24.1 Hvað þarftu?

- Raspberry Pi (með GPIO pinna)
- Brauðbretti (breadboard)
- 3x LED-ljós (rauð, gul og græn)
- 3x viðnám (t.d. 220 Ω)
- Tengisnúrur (jumper wires)

24.2 Tenging

Tengdu **langa fótinn** (jákvætt) á hverri LED í gegnum **viðnám** yfir raufina á brauðbrettinu. Tengdu hinum megin við viðnámið við eftirfarandi GPIO pinna:

- Rauður: GPIO `25`
- Gulur: GPIO `8`
- Grænn: GPIO `7`

Tengdu **styttri fótinn** (neikvætt) á hverri LED við - brautina (neikvæða braut). Hún þarf einnig að vera tengd við `GND` á Raspberry Pi.

> **Athugið:** Viðnámið má vera annaðhvort á milli LED og GND eða LED og GPIO — það skiptir ekki máli svo lengi sem viðnámið er hluti af straumrásinni.

24.3 Skrifaðu forritið

Opnaðu ritilinn (t.d. Thonny), skrifaðu eftirfarandi kóða og vistaðu skrána sem `traffic_signal.py`.

Listing 24.1: Umferðarljósaforrit

```
from gpiozero import LED
from time import sleep

# Skilgreining á ljósunum
red = LED(25)
amber = LED(8)
green = LED(7)

# Upphafsstöðugildi: grænt ljós logar
green.on()
amber.off()
red.off()

# Endalaus lykkja fyrir umferðarljós
while True:
    sleep(10)        # Bíddu í 10 sek
    green.off()      # Slökkva á grænu
    amber.on()       # Kveikja á gulu
    sleep(1)
    amber.off()
    red.on()
    sleep(10)
    amber.on()
    sleep(1)
    green.on()
    amber.off()
    red.off()
```

24.4 Keyrðu forritið

Í Thonny

Ýttu á **F5** eða smelltu á **Run** hnappinn.

Í skipanalínu

Farðu í möppuna þar sem skráin er vistuð og sláðu inn:

```
python3 traffic_signal.py
```

24.5 Sjálfvirk VLC keyrsla á Raspberry Pi

Ef þú vilt spila myndband úr vefstreymi beint í VLC glugga á Raspberry Pi, getur þú búið til skelskrift (`.sh`) sem gerir þetta. Þessi aðferð virkar þegar þú ert með skjá og gluggaumhverfi (t.d. Raspberry Pi OS Desktop).

24.5.1 Skrifaðu skelskrift sem ræsir VLC

Opnaðu terminal og búðu til nýja skrá:

```
nano ~/spila.sh
```

Límdu eftirfarandi kóða inn í skránna:

Listing 24.2: Skriptu dæmi sem ræsir VLC með streymi

```
#!/bin/bash
export DISPLAY=:0
export XAUTHORITY=/home/siggi/.Xauthority
vlc "https://cdn3.cast-tv.com/25349/Skjar1_ABR/playlist.m3u8"
```

Vistaðu og lokaðu með:

```
Ctrl + X, síðan ýttu á Y og Enter
```

24.5.2 Gerðu skrána keyranlega

```
chmod +x ~/spila.sh
```

24.5.3 Keyrðu VLC skriptuna

Nú getur þú keyrt skriptuna með:

```
~/spila.sh
```

Þetta ræsir VLC og byrjar að spila streymið á skjánum. Hafðu í huga að þessi aðferð virkar aðeins ef X11 gluggaumhverfi er virkt, og Raspberry Pi er tengdur við skjá.

24.5.4 Athugasemd

Ef skriptan er keyrð í gegnum SSH eða á kerfi án gluggaumhverfis, getur VLC ekki opnað myndglugga og gefur villu á borð við:

```
qt.qpa.xcb: could not connect to display
```

Í slíkum tilfellum er betra að nota `cvlc` til að spila án myndar (t.d. hljóð einungis).

Heimildir

[1] Phil King. *Simple Electronics with GPIO Zero: Take Control of the Real World with Your Raspberry Pi*. 2nd ed., Raspberry Pi Press, 2025.

25 Myndaskrá