Inngangur að forritun í Python

Forritun fyrir byrjendur

Valborg Sturludóttir

First printing, Sept 2020

Efnisyfirlit

Fyrsti hluti

1. Inngangur

1.1 Tilgangur bókarinnar

Þessi bók fjallar um þau undirstöðu atriði sem þarf að kynna til að ná tökum á forritun í Python. Höfundi finnst mikilvægt að kenna námsefnið með íslenskum hugtökum þar sem ætlunin er að nota hana í kennslu í íslenskum framhaldsskólum. Ef nemendur ætla að leggja fyrir sig tölvunarfræði í framhaldssnámi er nauðsynlegt að búa yfir ríkulegu íðorðasafni, þess þá heldur ef nemandi hyggst framfleyta fræðunum. Hugtök verða þó líka sett fram á ensku því lesandi gæti óskað að fletta upp íterefni sem meira er til af á netinu á ensku en íslensku.

Uppbyggining er þannig að fyrri hlutinn snýr að því að kynna lesandann fyrir grunn virkni Python; málskipan, lykilhugtök og lykilorð, gagnatýpur, lykkjur og föll. Seinni hlutinn snýr svo að því að beita þekkingu úr fyrri hlutanum í hlutbundinni forritun. Þar eru kynntir til sögunnar klasar og aðferðir sem lesandinn útfærir upp á eigin spýtur. Ekki er búist við neinni fyrri kunnáttu við lestur þessarar bókar, hún á að geta staðið fyrir sínu án þess að lesandinn búi yfir nokkurri þekkingu á sviði tölvunarfræða eða forritunar. Ef slík þekking er fyrir hendi gæti lesandanum þótt ágætt að fara hratt í gegnum fyrri hluta bókarinnar og einbeita sér að verkefnum úr seinni hlutanum. Í gegnum bókin fylgjum við svo þremur verkefnum sem verða þyngri og flóknari eftir því sem fleiri hugtök eru kynnt til sögunnar.

þremur??

1.2 Hvers vegna Python?

Ástæður þess að Python er gott mál til þess að byrja á að skoða eru eftirfarandi: [1].

1. *Málskipanin* er mjög svipuð mannlegu máli svo það er auðvelt að læra hvernig eigi að „tala" við tölvuna.
2. Python er *kvikt tagað* forritunarmál, það þýðir að notandinn þarf ekki að gefa upp hvers konar *gagnatýpur* er unnið með. Þetta gerir það að verkum að notandinn þarf ekki að læra urmull af lykilorðum áður en byrjað er að forrita.
3. Python er ekki alveg *hlutbundið* forritunarmál, sem gerir það að verkum að notandinn þarf ekki að læra hvernig á að beita hlutbundinni forritun fyrr en góð undirstaða er þegar komin.

[1] Strax í þessum texta koma fyrir hugtök sem verða skýrð betur seinna, ekki missa kjarkinn.

4. Python er frítt og aðgengilegt öllum helstu stýrikerfum og einnig er hægt að forrita yfir netið í vafra og því óþarfi fyrir notandann að setja nokkuð upp sé þess óskað.
5. Python er mikið notað, algengt mál svo það er praktískt að hafa undirstöðu skilning á því.
6. Nefnt í höfuð á Monty Pyhton grínhópsins.

ernig bókin er upp

1.3 Uppsetning

.

Víðsvegar um bókina, aðallega í upphafi, má finna númeraða kóðabúta sem eru ekki teknir úr vinnubók og eru því ekki eins litakóðaðir og þeir sem eru teknir úr vinnubókum. Ástæðan fyrir því er að þessum kóðabútum er auðveldara að viðhalda heldur en skjáskotum úr vinnubókum og því er heldur vísað í bækur sem eru aðgengilegar lesendum og frumstæðari framsetning ræður heldur ríkjum hér.

1.4 Að keyra kóða

Það fyrsta sem nemendur vilja yfirleitt gera er að byrja að skrifa sinn eigin kóða. Áður en við komumst svo langt þarf að útskýra hvernig það er gert. Þessi kennslubók byggir á notkun Jupyter Notebooks með hjálp Anaconda hugbúnaðarins, sem er öflugt pakkakerfi og tólakista sem hefur upp á mikið meira en bara Jupyter að bjóða. Hægt er að nálgast Anaconda á www.anaconda.com. Einnig er hægt að keyra kóða á netinu í gegnum síður eins og www.repl.it, nota ritla (eins og notepad eða sublime) til að keyra .py skrár í skipanalínu, eða nota þyngri umhverfi eins og pycharm sem eru sérhönnuð fyrir hugbúnaðarþróun. Hér er gert ráð fyrir Juptyter umhverfinu og verður bókin öll miðuð að því.

f tómri bók

hvernig hún

Þessari bók fylgja einnig nokkrar vinnubækur úr Jupyter sem lesandinn getur nýtt sér. Hér á mynd sést hvernig tóm Jupyter vinnubók lítur út. Virkninni er skipt upp í sellur og keyrsluröð sellanna skiptir máli, við sjáum seinna mikilvægi þess að geta skipt upp kóða svona og hvers vegna þetta umhverfi er þægilegt til að byrja í. En hver sella hefur aðgang að svokölluðu skilgreiningarsvæði vinnubókarinnar en er þó sín eigin eining, því má keyra eina sellu í einu án þess að keyra allan kóðann í vinnubókinni.

ri gott að taka emi úr vinnubók að fylgja

Hér væri réttast að skoða Vinnubók 1 sem fylgdi þessari bók.

1.5 Málskipan

iklu meira rnig á að nota ekur yfirhöfuð

Málskipan (e. syntax) er hugtak sem þýðir hvernig á að skrifa kóða svo að hann þýðist í vélamál sem tölvan skilur. Málskipan eru þær reglur sem við þurfum að fara eftir þegar við forritum, þær reglur sem forritunarmálið býst við að við förum eftir. Ef við brjótum þessar reglur fáum við villu, og einhver algengasta villa sem hægt er að fá er málskipunarvilla (e. syntax error). Python er frábrugðið öðrum forritunarmálum á þann hátt að málskipanin krefst þess að kóðinn sé settur upp á ákveðinn hátt. Líkja því má við að þurfa ekki að hafa greinamerki í huga þegar við ljúkum setningum heldur setjum við setningarnar okkar á réttan stað í samræðunum.

1.5.1 Uppsetning á kóða

Þessi kóðabútur er þannig uppsettur að allar línur byrja jafnlangt til vinstri, eins og hver setning í töluðu máli stendur hver lína fyrir sínu ein og sér.

Kóðabútur 1.1: Réttur Python kóði

```
# Réttur Python kóði sem keyrist
4 + 8
5 + 6
```

```
breyta = 9 * 2
```

Þessi næsti kóðabútur hinsvegar er ekki nógu vel uppsettur, þar eru „setningar” sem virðast hanga undir öðrum og vera þeim háðar.

Kóðabútur 1.2: Rangur Python kóði

```
# Illa skrifaður Python kóði sem keyrist ekki
4 + 8
   5 + 6
breyta = 9 * 2
```

Svona inndrætti er einungis beitt ef lína á beinlínis að hanga undir línunni að ofan og tilheyrir henni. Þess vegna þarf að huga að því hvernig kóði er uppsettur. Í öðrum málum eru notuð greinamerki til að segja tölvunni að lína sé búin og að aðrar línur eigi að heyra undir eitthvað ákveðið samhengi en ekki í Python, þar er treyst á að forritarinn setji kóðann upp á máta sem hægt er að sjá að sé réttur. Dæmi um hvernig línur geta verið aðgreindar í öðrum málum:

Kóðabútur 1.3: Dæmi um annað mál sem er strangt tagað og með greinamerkjum

```
// Java
int i = 7;
i + 5;
```

Kóðabútur 1.4: Dæmi um annað mál sem byggir á afmörkuðu samhengi en með greinamerkjum

```
; Lisp
(setq x 10)
(setq y 34.567)

(print x)
(print y)
```

Í þessum tveimur frábrugnu málum sem voru tekin sem dæmi var óþarfi að setja kóðann í mismunandi línur, því greinamerkin væru nóg til að aðgreina hverja línu fyrir sig. Hins vegar er það góð venja að skrifa kóða sem er læsilegur öðru fólki. Í Java eru greinamerkin semikommur (;) en í Lisp eru línur og samhengi afmörkuð með svigum. Python byggist hinsvegar á því að forritarinn stilli öllu upp rétt með réttum inndrætti.

1.5.2 Gagnatýpur og lykilorð

Í Python eru nokkrar grunn gagnatýpur sem við munum kynnast í þessari bók. Ástæðan fyrir því að þær eru kallað grunntýpur er sú að þær fylgja með Python uppsetningunni og notandinn getur beitt þeim í samræmi við það sem þær eru færar um, sem má skoða í skjölun Python https://www.python.org/doc/. Týpa eða tag er hugtak sem þýðir að hlutur sé af einhverri ákveðinni tegund sem má framkvæma ákveðnar aðgerðir á. Lesandi þekkir muninn á orðum og tölum úr daglegu tali og veit að hægt er að framkvæma mismunandi aðgerðir á þessum mismunandi týpum, eins og hægt er að skipta út hástöfum fyrir lágstafi í orðum en ekki tölum og hægt er að hefja tölur í veldi en ekki orð. Að sama skapi eru til aðgreinanlegar týpur sem tölvan kann skil á og leyfir ákveðnar aðgerðir á. Í fyrri hluta þessarar bókar verða gerð skil á tveimur talnatýpum (heiltölum og fleytitölum), strengjum, listum, orðabókum (einnig kallaðar hakkatöflur) og boolean gildum. Í seinni hlutanum bæstast svo við sett .

sett?

Lykilorð eru orð sem eru frátekin og birtast þau græn í Jupyter vinnubók. Hver gagnatýpa hefur eitt lykilorð og eru einnig nokkur innbyggð föll í Python, sem við kynnumst fljótlega, með frátekin

orð. Forðast skal að yfirskrifa þessi lykilorð, en gerist það þá er auðvelt að laga það í Jupyter. Hver vinnubók hefur sinn kjarna til að vinna á og það eina sem þarf að gera í aðstæðum það þar sem innbyggt orð er allt í einu farið að þýða eitthvað annað þá dugir að endurræsa kjarnann.

2. Tölur og breytur

Í þessum kafla ætlum við að hefjast handa við að forrita. Það fyrsta sem við ætlum að gera er að kynnast talnatýpum og keyra kóða eins og við værum að nota reiknivél. Við könnumst við reiknivélar og hvernig þær afgreiða röð aðgerða. Nú viljum við sannreyna að þær reikniaðgerðir sem við þekkjum séu til í Python og að þegar við keyrum kóðann okkar þá verði útkoman sú sama og við áttum von á. Við viljum líka geta geymt útkomuna okkar til að nota aftur seinna, til þess þurfum við breytur (e. variables).

2.1 Tölur - talnatýpur

Í Pyhton eru í grunninn tvær týpur af tölum (en til eru tvær týpur af hvorri fyrir sig, sem snýr meira að minnisnotkun og er út fyrir svið þessarar bókar). Þær eru:

- **Heiltölur** - tölur sem eru ekki með neinum aukastaf. Á ensku eru þessar tölur kallaðar integers og er lykilorð þeirra því **int**.
- **Fleytitölur** - tölur sem eru með aukastaf, sem er fyrir aftan punkt (ekki kommu, fleytitölur eru oft kallaðar kommutölur á íslensku). Á ensku eru þessar tölur kallaðar floating point numbers og er því lykilorðið þeirra **float**.

Kóðabútur 2.1: Heiltölur og fleytitölur

```
# Heiltölur, enginn aukastafur
42
100000
-139

# Fleytitölur, aukastafur/ir fyrir aftan punkt
4.0
3.1415926
-100.98
```

2.2 Reikniaðgerðir og tákn

Grunn reikniaðgerðir eru nokkrar sem við könnumst við úr grunnskóla en aðrar eru framandi og við skulum skoða aðeins betur.

Í eftirfarandi dæmum er vert að draga fram nokkur atriði sem eru ekki augljós byrjanda. Það fyrsta er að myllumerkið (#) þýðir að allt sem kemur fyrir aftan það er *athugasemd*, athugasemdir eru engöngu til að gera kóða læsilegri fyrir fólk, þær eru hunsaðar af tölvunni þegar hún breytir kóðanum í eitthvað sem hún skilur. Einnig eru þarna bil á milli talna og tákna, það er líka til að gera kóðan læsilegri, bilin mega bara ekki vera fremst í línunni enn sem komið er.

Kóðabútur 2.2: Reikniaðgerðir

```
# Samlagning framkvæmd með +
# Þegar eftirfarandi kóði er keyrður ætti útkonan að vera 10
6 + 4

# Frádráttur framkvæmdur með -
# Þegar eftirfarandi kóði er keyrður ætti útkonan að vera 10
14 - 4

# Margföldun framkvæmd með *
# Þegar eftirfarandi kóði er keyrður ætti útkonan að vera 10
10 * 2

# Deiling framkvæmd með /
# Athugið að þetta er fleytitöludeiling sem skilar nákvæmu svari
# Þegar eftirfarandi kóði er keyrður ætti útkonan að vera 10.0
60 / 6

# Heiltöludeiling framkvæmd með //
# Athugið að þessi deiling er frábrugðin þeirri sem þið kannist við
# Hér viljum við vita hversu oft, heil tala, ein tala gengur upp íaðra og okkur
    er sama um afganginn
# Þegar eftirfarandi kóði er keyrður ætti svarið að vera 10
177 // 17

# Veldishafning framkvæmd með **
# Hér er mikilvægt, eins og með deilinguna, að hafa íhuga hvor talan kemur á
    undan.
# Fyrst kemur talan sem hefja áíveldi og svo kemur talan sem er veldisvísirinn
# Þegar eftirfarandi kóði er keyrður ætti svarið að vera 9
3 ** 2

# Leifareikningur framkvæmdur með % (e. modulus)
# Þetta er eitthvað alveg nýtt og framandi, en þóekki óskiljanlegt
# Það sem þetta reiknar er hversu mikil leif eða afgangur er eftir þegar
    heiltöludeilingu er beitt.
# Þegar eftirfarandi kóði er keyrður ætti svarið að vera 7
177 % 17
```

Í öllum þessum dæmum var verið að vinna með heiltölur, þó var útkoman úr deilingunni (stundum kölluð fullkomin deiling) fleytitala. Hvað gerist ef þessir sömu útreikningar eru gerðir með fleytitölum? Ef við myndum skipta út hverri tölu fyrir sig og setja í staðinn sömu tölu með .0 fyrir aftan þá yrðu útkomurnar þær sömu nema fleytitölur. En hvað gerist ef við breytum aðeins fyrri tölunni en ekki seinni tölunni? Þá ertu að nota ólíkar týpur og slíkt er vandmeðfarið, en í þessu

tilviki er það í lagi þar sem Python gerir þá ráð fyrir að það sé í lagi að reikna með fleytitölum og framkvæmir reikninginn eins og þú hafir verið að beita fleytitölum og niðurstaðan verður þá að sjálfsögðu fleytitala.

2.2.1 Breytur

Nú höfum við séð hvernig má keyra kóða einfaldlega eins og í reiknivél. Höldum okkur við samlíkinguna um reiknivélina til að útskýra breytur. Á hefbundinni reiknivél sem notuð er í stærðfræðitíma í framhaldsskóla er takki sem á stendur ANS. Það stendur fyrir answer og ef ýtt er á hann getur vélin geymt síðasta gildið sem hún gaf sem svar og unnið svo með það til að gefa næsta svar. Flottari vélar geta svo geymt nokkuð mörg svör en það er útfyrir gagnsemi þessarar samlíkingu. Þegar ýtt er á þennan takka er minnisvæði í reiknivélinni tekið frá og skrifað er í það gildi, sem er svo sótt þegar ANS er notað í útreikningi. Að sama skapi má láta Python úthluta minnissvæði í tölvunni fyrir þær breytur sem þið viljið geyma. Munurinn er sá að þið nefnið sjálf hvað minnisvæðið er merkt sem, eruð ekki bundin við að nota ANS og að þið eruð svo gott sem með óteljandi minnissvæði.

Að gefa minnissvæði merkingu og gildi er gert með *gildisveitingu*. Gildisveiting þýðir að nú er einhver ákveðinn merkimiði kominn með eitthvað til að geyma. Sjáum einfalt dæmi um þetta.

Kóðabútur 2.3: Breytur kynntar

```
# Hér er ég að fara að búa til breytu sem heitir val
val = 5

# Þegar ég keyri línuna fyrir ofan segi ég vélinni að hafa aðgengilegt
    minnisvæði sem ég get notað með því að skrifa orðið val, og settu íþað svæði
    gildið 5.

# Svo ég er að veita breytunni val gildið 5, þess vegar er það kallað
    gildisveiting.

# Svo get ég notað breytuna mína
# þegar þetta er keyrt fæst svarið 10
val + 5
```

Ef þú prófar þig áfram við að búa til breytur gætir þú rekist á svolítið sem hefur ekki gerst áður í vinnubók, að þegar sella inniheldur eingöngu gildisveitingu og er keyrð þá „gerist ekkert". Þetta finnst mörgum mjög skrýtið því þau vilja fá einhverja útkomu. En útkoman er sú að þú sagðir vélinni að geyma þetta, þú sagðir henni ekki að gera neitt annað. Og þá komum við niður á stórt vandamál, að tölvur eru mjög bókstaflegar og vitlausar. Þær skortir allt vit, þær reyna ekki að hafa vit fyrir þér. Þær gera nákvæmlega það sem þú baðst um. Nákvæmlega eins og þú baðst um það.

Þannig að ef ég ætlaði að segja tölvu að smyrja handa mér hnetusmjörs og sultu samloku þá þyrfti ég að segja vélinni að gera eftirfarandi í nákvæmlega þessari röð:

1. taka fram hníf
2. taka fram tvær brauðsneiðar
3. opna hnetusmjörið
4. setja beitta endann ofan í hnetusmjörið þannig að hann nái upp 50gr af hnetusmjöri
5. setja hnetusmjörið sem er á hnífnum á miðja brauðsneiðina
6. nota hnífinn til þess að smyrja hnetusmjörinu á þá hlið sem hnetusmjörið er nú þegar á, og enga aðra
7. taka fram skeið
8. opna sultuna

9. setja kúpta enda skeiðarinnar ofan í sultukrukkuna
10. taka skeiðina upp úr sultukrukkunni með kúfaða skeið af sultu
11. setja sultuna á hina brauðsneiðina
12. nota skeiðina til að smyrja sultunni yfir þá hlið brauðsneiðarinnar sem sultan er á og enga aðra hlið
13. setja brauðsneiðarnar saman þannig að hnetusmjörið og sultan snertist og hornin mætast öll.

Takið eftir að hér er gert ráð fyrir þó nokkur og ef véliin kann ekki nú þegar skil á:

1. taka fram
2. hnífur
3. opna
4. mæla 50 gr
5. smyrja
6. hlið á brauðsneið
7. miðja á brauðsneið
8. skeið
9. kúfað

Svo þó svo að þér hafi þótt þessi útskýring á samlokugerð alveg ofboðslega óþarflega nákvæm þá er ekki víst að úr þessu verði nokkur samloka. Þetta könnumst við öll við, að tölvur gera það sem þeim er sagt ekki það sem við viljum. Að því sögðu þá þurfum við að skoða breytur nokkuð betur áður en við förum að beita þeim á skilvirkan hátt.

Breytur eru skilgreindar vinstra megin við jafnaðarmerki í Python. Eins og það væri lesið, val fær gildið 5. Það væri lítið vit í því að hafa það öfugt, 5 er núna jafngilt val. Það sem við værum þá að segja tölvunni að í hver sinn sem hún vill nota heiltöluna fimm þá á hún að hætta við að nota töluna sjálfa og í staðinn vísa eingöngu í það sem er í minnissvæði merktu val. Það er alls ekki það sem við viljum.

Kóðabútur 2.4: Dæmi um gildisvetingar réttar og rangar

```
# Hér er ég að fara að búa til breytu sem heitir val
val = 5

# Hér er ég ekki að búa til breytu sem heitir val heldur er ég að segja að talan
    fimm er ekki lengur til sem ehiltala heldur gæti hún vísað íhvað sem er sem
    er geymt íminnisvæði merktu val
5 = val

# Hér bý ég til breytu sem heitir heiltala sem fær gildið 0
heiltala = 0

# Hér yfirskrifa ég lykilorðið fyrir týpuna heiltala og læt það innihalda 0
int = 0
# þetta er harðbannað og ef þetta gerist er ekki nóg að þurrka þetta út og keyra
    aftur, nú þarf að endurræsa kjarna vinnubókarinnar.
```

Nú þegar við höfum séð hvernig má skilgreina breytu viljum við vita hvernig á að nota þessa breytu. Ef við snúum okkur aftur að reiknivélasamlíkingunni um ANS takkann þá ætti eftirfarandi kóðabútur að geta sýnt með eðlislægum hætti hvernig breytur nýtast.

Kóðabútur 2.5: Að nota breytu

```
# Hér framkvæmi ég útreikning sem ég geymi íbreytunni ANS
ANS = 5**2 + (4+8.9)**2
```

```
# Segjum að þetta hafi verið endapunkturinn ílöngu algebrudæmi og nú veit ég
    hvað y er, og get þánýtt það til að finna x eins og verða vill svo oft í
    stærðfræði að x sé týnt. Gefum okkur að x = 3 * y og því fæst
x = 3 * ANS

# Nú ef við viljum reikna eitthvað út með x eigum við það til íminnissvæði
    merktu x með réttu gildi.

# Nú langar okkur til að vera viss um að við séum við vitrænt svar svo við
    biðjum tölvuna um að segja okkur hvað er geymt íbreytunni x.
print(x)
```

Strengir

Til þess að geta sýnt og notað texta þarf gagnatýpu til að halda utan um hann. Í flestum forritunarmálum, og Python er ekki undantekning, eru gögn af þeirri týpu kölluð **strengir**.

2.2.2 Descriptions and Definitions

Name Description
Word Definition
Comment Elaboration

3. In-text Elements

3.1 Theorems

This is an example of theorems.

3.1.1 Several equations

This is a theorem consisting of several equations.

Theorem 3.1.1 — Name of the theorem. In $E = \mathbb{R}^n$ all norms are equivalent. It has the properties:

$$\big|\,||\mathbf{x}|| - ||\mathbf{y}||\,\big| \leq ||\mathbf{x}-\mathbf{y}|| \tag{3.1}$$

$$||\sum_{i=1}^{n} \mathbf{x}_i|| \leq \sum_{i=1}^{n} ||\mathbf{x}_i|| \quad \text{where } n \text{ is a finite integer} \tag{3.2}$$

3.1.2 Single Line

This is a theorem consisting of just one line.

Theorem 3.1.2 A set $\mathscr{D}(G)$ in dense in $L^2(G)$, $|\cdot|_0$.

3.2 Definitions

This is an example of a definition. A definition could be mathematical or it could define a concept.

Definition 3.2.1 — Definition name. Given a vector space E, a norm on E is an application, denoted $||\cdot||$, E in $\mathbb{R}^+ = [0, +\infty[$ such that:

$$||\mathbf{x}|| = 0 \;\Rightarrow\; \mathbf{x} = \mathbf{0} \tag{3.3}$$

$$||\lambda \mathbf{x}|| = |\lambda| \cdot ||\mathbf{x}|| \tag{3.4}$$

$$||\mathbf{x}+\mathbf{y}|| \leq ||\mathbf{x}|| + ||\mathbf{y}|| \tag{3.5}$$

3.3 Notations

Notation 3.1. *Given an open subset G of $\mathbb{R}^n$, the set of functions φ are:*

1. *Bounded support G;*
2. *Infinitely differentiable;*

a vector space is denoted by $\mathscr{D}(G)$.

3.4 Remarks

This is an example of a remark.

R The concepts presented here are now in conventional employment in mathematics. Vector spaces are taken over the field $\mathbb{K} = \mathbb{R}$, however, established properties are easily extended to $\mathbb{K} = \mathbb{C}$.

3.5 Corollaries

This is an example of a corollary.

Corollary 3.5.1 — Corollary name. The concepts presented here are now in conventional employment in mathematics. Vector spaces are taken over the field $\mathbb{K} = \mathbb{R}$, however, established properties are easily extended to $\mathbb{K} = \mathbb{C}$.

3.6 Propositions

This is an example of propositions.

3.6.1 Several equations

Proposition 3.6.1 — Proposition name. It has the properties:

$$\big|\,||\mathbf{x}|| - ||\mathbf{y}||\,\big| \leq ||\mathbf{x}-\mathbf{y}|| \tag{3.6}$$

$$||\sum_{i=1}^{n}\mathbf{x}_i|| \leq \sum_{i=1}^{n}||\mathbf{x}_i|| \quad \text{where } n \text{ is a finite integer} \tag{3.7}$$

3.6.2 Single Line

Proposition 3.6.2 Let $f, g \in L^2(G)$; if $\forall \varphi \in \mathscr{D}(G)$, $(f,\varphi)_0 = (g,\varphi)_0$ then $f = g$.

3.7 Examples

This is an example of examples.

3.7.1 Equation and Text

■ **Example 3.1** Let $G = \{x \in \mathbb{R}^2 : |x| < 3\}$ and denoted by: $x^0 = (1,1)$; consider the function:

$$f(x) = \begin{cases} \mathrm{e}^{|x|} & \text{si } |x-x^0| \leq 1/2 \\ 0 & \text{si } |x-x^0| > 1/2 \end{cases} \tag{3.8}$$

The function f has bounded support, we can take $A = \{x \in \mathbb{R}^2 : |x-x^0| \leq 1/2+\varepsilon\}$ for all $\varepsilon \in]0;5/2-\sqrt{2}[$. ■

3.7.2 Paragraph of Text

■ **Example 3.2 — Example name.** Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris. ■

3.8 Exercises

This is an example of an exercise.

Exercise 3.1 This is a good place to ask a question to test learning progress or further cement ideas into students' minds. ■

3.9 Problems

Problem 3.1 What is the average airspeed velocity of an unladen swallow?

3.10 Vocabulary

Define a word to improve a students' vocabulary.
Vocabulary 3.1 — Word. Definition of word.

Part Two

4. Presenting Information

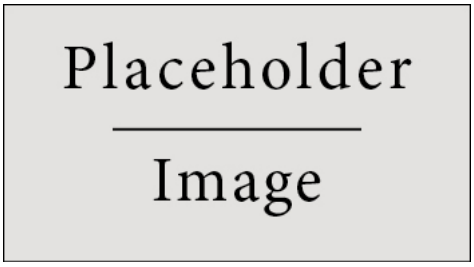
4.1 Table

Treatments	Response 1	Response 2
Treatment 1	0.0003262	0.562
Treatment 2	0.0015681	0.910
Treatment 3	0.0009271	0.296

Tafla 4.1: Table caption

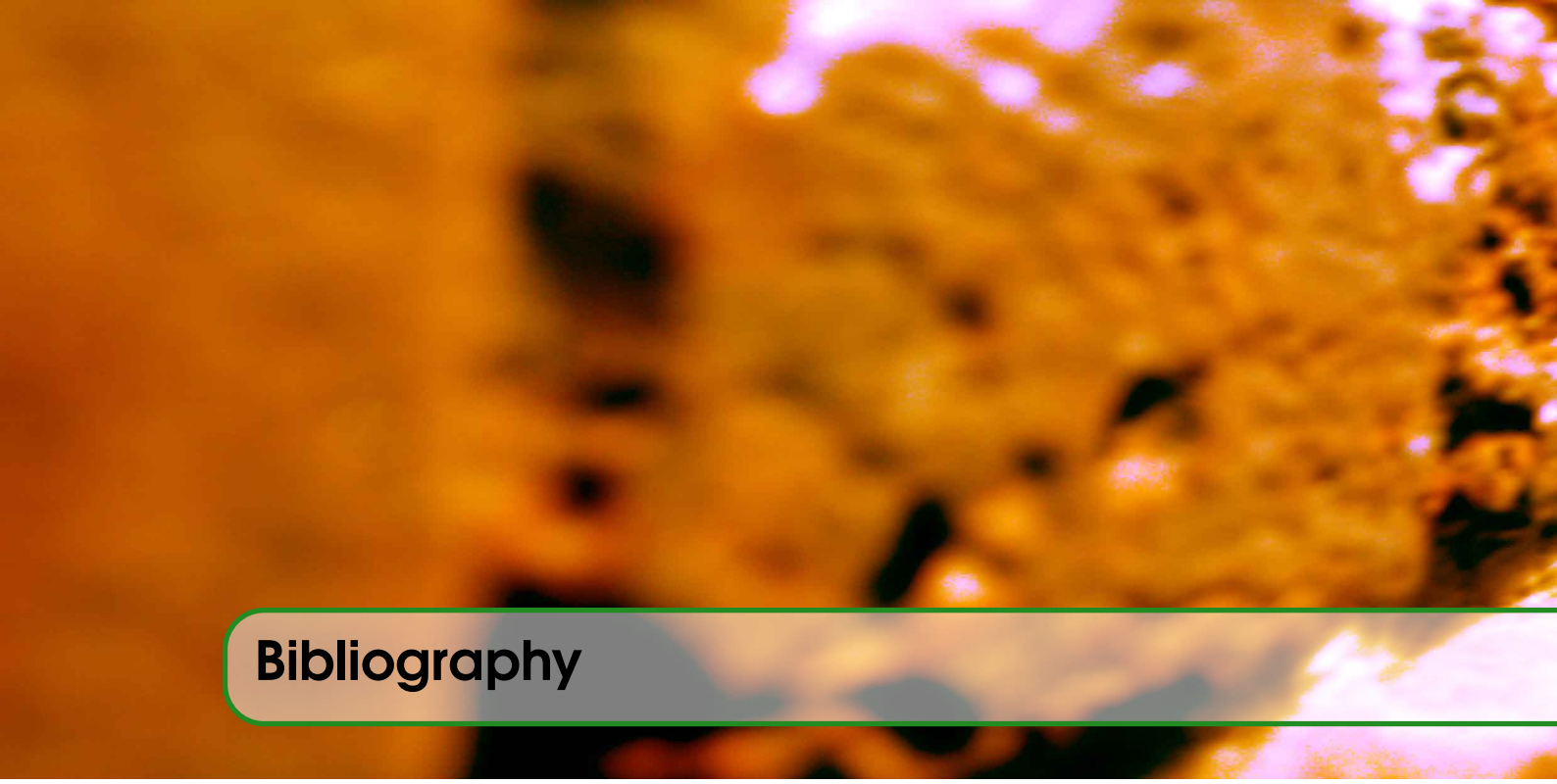
Referencing Table 4.1 in-text automatically.

4.2 Figure



Mynd 4.1: Figure caption

Referencing Figure 4.1 in-text automatically.



Bibliography

Articles

[1] James Smith. “Article title”. Í: 14.6 (mar. 2013), blaðsíður 1–8.

Books

[2] John Smith. *Book title*. 1. útgáfa. Bindi 3. 2. City: Publisher, jan. 2012, blaðsíður 123–200.

Atriðisorðaskrá