

# Dagskrá í viku 8:

## Praktísk atriði í forritun

- Miðvikudagur 26.2
  - Jafningjamat og kennslukönnun
  - Föll (stef)
  - Súðókóði
  - Athugasemdir í kóða
  - Varnarforritun og villumeðhöndlun
  - Breytur
  - Tímaverkefni

# Jafningjamat

Allir í hópnum verða að skila jafningjamati til að fá lokaeinkunn fyrir hópverkefni 1. Frestur til að svara er fram að næsta fyrirlestri, miðvikudaginn 5. mars.

[https://docs.google.com/forms/d/1z2TwoU6l\\_Qwgqdoxt4VHzjnetD53lh4rSTA4Ej-vFr8/viewform](https://docs.google.com/forms/d/1z2TwoU6l_Qwgqdoxt4VHzjnetD53lh4rSTA4Ej-vFr8/viewform)

# Kennslukönnun

- Tvö atriði sem stóðu upp úr:
  - Slæmt að vera með fólk á ólíku þekkingarstigi í sama námskeiði.
  - Námskeiðið mjög krefjandi og verkefnin heldur til of erfið.

# Val á forritunarmáli

- Flest grunnatriði í forritun eru óháð máli
- Afköst 30% meiri þegar forritarar þekkja málið vel
- Æðri mál (e. high level languages)
  - Hver lína gerir meira
  - læsileiki, einfaldleiki og áreiðanleiki meiri

# Losnum við endurtekningar

- Nóg að breyta forritinu á einum stað
- Minni hættu á villum, auðveldar viðhald

```
if hvitur_a_leik:
    # flokinn kodi kannar hvort
    # hvitur hafi valid leyfilegan leik
    # ...
else:
    # flokinn kodi kannar hvort
    # svartur hafi valid leyfilegan leik
    # ...
```

→ `if getur_leikid(leikmadur):`  
#...

↖ Setja í fallið  
`getur_leikid(litur)`

# Nafn á föllum

- Á að vera lýsandi fyrir það sem fallið gerir
  - `get_name()`, `is_empty()`
- Óskýrt nafn er vísbending um lélegan kóða
  - `part1()`, `do_calculations()`
- Heppileg lengd á nafni er 9 – 15 stafir

# Lengd falla

- Sum stef eru í eðli sínu mjög stutt
  - t.d. get/set föll í klösum
- Halda sig annars við 50 – 150 línur af kóða
  - auðar línur og athugasemdir ekki taldar með
- Rannsóknir sýna að >200 línur kalla á vandræði
  - Erfitt að hafa yfirsýn/skilning á virkni

# Hönnun á falli

- Er raunverulega þörf á fallinu?
- Skilgreina hlutverk fallsins og gefa því nafn
  - inntök, úttök, forskilyrði, eftirskilyrði
- Ákvarða hvernig á að prófa fallið
- Er hægt að nýta tilbúin forritasöfn/reiknirit?
- Hvernig á að meðhöndla villur?
- Skrifa súdókóða



# Súdókóði

- Óformleg lýsing á því hvernig reiknrit, fall, klasi eða forrit eiga að virka
- Skrifa þ.a. forritun verði næstum formsatriði
- Forðast tilvísanir í tiltekið forritunarmál

# Dæmi um súdókóða

```
FOR X = 1 to 10
  FOR Y = 1 to 10
    IF gameBoard[X][Y] = 0
      Do nothing
    ELSE
      CALL theCall(X,Y)
      increment counter
    END IF
  END FOR
END FOR
```

Sleppur fyrir horn

```
Set moveCount to 1
FOR each row on the board
  FOR each column on the board
    IF gameBoard position (row,
      column) is occupied THEN
      CALL findAdjacentTiles
        (row, column)
      INCREMENT moveCount
    END IF
  END FOR
END FOR
```

Betra

# Athugasemdir í kóða

- Oft eina lýsingin sem til er á kóðanum
- Eru oftast “up to date”
- Vísa í hönnunarskjöl/heimildir þar sem við á
- Setja inn jafnóðum, ekki í lokin
- Þumalputtaregla: 1 aths á 10 línur af kóða


# Tegundir athugasemda

- Endurtekning á kóða → `# Call Move robot`  
`aRobot.move()`
  - Bætir engu við
- Útskýring á kóða
  - Eingöngu gagnlegt þegar kóði er torskilinn.  
Lausnin felst í að bæta kóðann!
- Merkimiðar ← To be done
  - `return None` `# TBD: Meðhöndla sértilfelli`
  - Nota staðlaða merkimiða, t.d. `# ***`

# Varnarforritun

- Vernda gegn aðstæðum sem eiga ekki að geta komið upp. Rusl inn=>rusl út óásættanlegt
  - Rusl inn => ekkert út
  - Rusl inn => villuboð
  - Ekkert rusl inn
- Prófa inntök sem koma að utan
- Prófa inntök í öll stef
- Ákvarða hvernig meðhöndla eigi óleyfileg inntök

Frá notendum eða  
öðrum kerfum. T.d.  
kennitölur, símanúmer



# assert skipunin

Notað til að  
aflúsa kóða

- Prófa kóða á meðan **próun** stendur
- Prófa tilfelli sem eiga “aldrei” að geta gerst

- **Dæmi**

– **assert**  $x \geq 0$ , 'x is less than zero'  
 $x < 0$  framkallar villuna

**AssertionError: x is less than zero**

Prófa m.a. hvort  
inntök og úttök  
séu í lagi

Assert skipunin er til staðar í flestöllum forritunarmálum

# Dæmi um notkun á **assert**

- Kanna hvort skrá sé opin áður en hún er lesin
- Til að skrá for- og eftirskilyrði falla
- Kanna hvort inntak í fall sé leyfilegt

```
def finna_aldur(kennitala):  
    assert(len(kennitala) == 10)
```

# Villumeðhöndlun

- **assert** nýtist til að finna villur sem eiga aldrei að geta komið upp
- Hvernig á að tækla villur sem búast má við?
  - Fall skilar hlutlausu gildi, t.d. **None**
  - Skrifa viðvörðunarloð í skrá (log file)
  - Skila villukóða
  - Kalla á undirkerfi sem meðhöndlar villur
  - Birta notanda villuskilaboð eða stöðva forrit



# Frábrigði

- Leið til að koma villuboðum og frávíkis-  
tilfellum á framfæri til þess sem kallaði á  
kóðann

```
try:
    fh = open("testfile", "w")
    fh.write("This is my test file for exception handling!!")
except IOError:
    print "Error: can't find file or read data"
else:
    print "Written content in the file successfully"
```

← Keyrt ef villa kemur upp

← Keyrt ef ekkert kemur upp á

# Villumeðhöndlun - samantekt

- Ákvarða fyrirfram hvernig meðhöndla eigi villur (hönnunarákvörðun)
- Meðhöndla fyrirsjáanlegar villur með viðeigandi kóða, t.d. **try-except**
- Notaðu **assert** á þróunartíma, beita á tilfelli sem eiga aldrei að geta komið upp

# Breytur

- Breyta sem er ekki upphafsstillt er villuupspretta
  - Gildi breytu getur verið úrelt
  - Hluti breytunnar (t.d. klasa) hefur ekki verið upphafsstilltur
  - Getur innihaldið drasl í C/C++

# Breytur

- Upphafsstilla tilviksbreytur í klasasmíð
- Huga að teljurum (i,j,k)
  - gleymist oft að endursetja
- Hlusta á viðvaranir frá C/C++ þýðanda
- Breytunafn á að lýsa fyrirbærinu nákvæmli.
  - interest\_rate, monthly\_total, last\_payment, winner\_of\_Eurovision\_song\_contest



Of langt! Takmarka nafn  
við 10-16 tákn

# Skilgreining á breytum (Java/C++)

- Skilgreina allar breytur
  - `int i=3; double* dData = NULL;`
- Nota nafnakerfi, t.d.  
dRadius, iCount, strName, ...  
(d=double, i=int, str=string, ...)
- Kóði lesinn miklu oftár en hann er skrifaður
- Ekki spara tíma við inslátt á forriti!

# Breytunöfn sem ætti að forðast

- Forðast að nota númer í nöfnum
  - Name1, file2, total3
- Forðast stafssetningarvillur
  - Erfitt að muna
- Ekki blanda saman tungumálum
  - Best að nota ensku
- Forðast nöfn með torlesnum táknum
  - eyeChart1, eyeChartI, eyeChart1

# Nafnakerfi

- Nota þegar
  - margir forritarar vinna að sama verkefni
  - aðrir þurfa að rýna kóðann
  - langur tími (vikur, mánuðir) líða milli þess sem unnið er í forritinu
- Styttir tíma sem þarf til að setja sig inn í forrit sem aðrir skrifa

Sjá t.d. Python Style Guide

# Tímaverkefni 8

- Ef ég ætlaði mér að búa til fall sem tekur inn tölu og margfaldar hana með sjálfri sér,  $x * x$ , hvað væri skynsamlegt að skíra fallið?
- Ef ég ætla mér að lesa upp úr skrá inni í forritinu mínu, er eitthvað sem ég þarf að gera annað en að skrifa kóðann sem les úr skránni?
- Hvers vegna er ekki gott að vera með mikið af endurtekningum í kóða?



# Ítarefni

- Code Complete eftir Steve McConnell
- Python style guide
- Súðókóði: [http://users.csc.calpoly.edu/~jdalbey/SWE/pdl\\_std.html](http://users.csc.calpoly.edu/~jdalbey/SWE/pdl_std.html)
- ...