Python-workshop

Välkommen till vår Python-workshop! Denna workshop är uppdelad i olika svårighetsgrader för att matcha olika personers tidigare erfarenhet av programmering/Python.

Delarna är uppdelade enligt:

A. Introduktion
B. Start för nybörjare till Python/programmering
C. Problem för dig som programmerat innan (Python)
D. Problem för dig som är mer erfaren

Man får även självklart göra något helt eget projekt under workshoparna (behöver inte ens vara Python). Det är även upp till er om ni arbetar i par eller grupp, ni kommer nog behöva dela på datorerna för att de ska räcka till. Lycka till och tveka inte att fråga om hjälp från oss!

Kapitel A.1, A.2, A.3, A.4 kan vara bra att läsa igenom även om man programmerat i Python innan då det går igenom hur man använder LiUs datorer.

Innehållsförteckning

A. Komma igång med LiUs datorer

Följ stegen nedan för att komma igång med att använda dessa datorer.

1. Öppna en terminal

Öppna en terminal genom att klicka på terminal-fönstret i vänstra hörnet eller med shortcut CTRL+ALT+T.

2. Ladda programmet VS Code

För att kunna använda text-editorn VS Code på LiUs datorer måste man först ladda in programmet. Detta görs genom följande kommando, skriv in det i terminalen du precis öppnade:

```
Unset
module add prog/vscode
```

(detta behövs göras varje gång du loggar in på datorn)

3. Skapa en fil och öppna i text-editor

Starta en VS Code och skapa en ny Python-fil i terminalen genom att skriva in följande:

```
Unset
code start.py
```

(detta skapar en ny fil kallad “start.py” i din home-folder)

4. Spara filer mellan datorer

Vi kommer fortsätta denna workshop kl 13 - 14 idag, och dessvärre har IT-avdelningen inte lyckats lösa så era konton lagrar filer över olika datorer. Därför måste ni ladda upp filer på t.ex. Google drive om ni vill spara dessa till den senare workshopen. Ni kan göra detta på valfritt sätt, fråga oss om ni behöver hjälp.

Lite mer avancerad lösning

Följande lösning kan du testa ifall du vill, annars kan det vara lättare att ladda upp på Google drive. Detta kräver att du kommer ihåg/skriver ner vilket ID (t.ex. SU10-104) datorn du använde i workshop 1 hade (se lapp på vänstra sidan på skärmen). Då kan du skriva in följande i terminalen på datorn du använder under workshop nr 2:

```
Unset
scp su[sal]-[datornamn].ad.liu.se:/mnt/$USER/* ~/
```

Där [sal] är antingen 10/11 (vilken datorsal du satt i) och datornamn är ett tre-siffrigt ID för datorn du satt vid. T.ex: satt du vid dator SU10-104 blir kommandot:

scp su[sal]-[datornamn].ad.liu.se:/mnt/$USER/ ~/*

B. Komma igång med Python

För att komma igång kan du börja med att skapa och köra ett första Python-program!

1. Skapa och starta ett program

Nu ska du få testa skriva Pythonkod! Du kan kopiera in följande i din nya fil (eller skriva något eget):

Python
```python
def main():
    print("Hello world!")


# Standard syntax för att köra main-funktionen
if __name__ == "__main__":
    main()
```

För att köra koden går du tillbaka till terminalen och skriver in följande:

Unset
```
python start.py
```

Du borde nu fått ut meddelandet “Hello world!” i terminalen, bra jobbat! Du kan nu testa koda själv, se nästa kapitel för att ta reda på hur man gör olika saker i Python. Du kan även testa på något problem från del B eller hitta på något eget program!

Syntax i Python

I följande avsnitt står det hur funktioner, if-satser, loopar och variabler skrivs i Python.

Många exempel nedan innehåller text skriven med “#” framför, detta är kommentarer och gör ingenting i programmet, utan är bara till för att skriva information till den som läser koden.

Variabler

Variabler har ingen strikt typning i Python, dvs det kan anta nästan vilket värde som helst. De definieras på följande sätt:

```python
x = 1
b = True
text = "This is some text"

numberList = [1,2,3,4,"text"] # Kan innehålla olika typer

dictionary = {
  "MafU": "D-Tour",
  "LiU": "universitet"
}
```

If-satser

If-satser är ett sätt att styra vilka delar av programmet som körs, se nedan:

```python
x = 2

if x == 2:
    print("X is 2!") # Detta printas då x=2
elif x == 3:
    print("X is 3!")
else:
    print("X is not 2")
```

Funktioner

Funktioner, kod som kan anropas och köras flera gånger, definieras på följande sätt:

Python

```python
# Funktionsdeklaration
def funktionsnamn(argument1, arg2):
    return "Argument: " + argument1 + ", argument2: " + arg2

funktionsnamn("hej", "då") # Anrop
```

Loopar

Loopar, kod som körs flera gånger, kan definieras på två olika sätt (huvudsakligen):

Python

```python
# For-loop ('i' kommer gå från 1 till och med 9)
for i in range(1,10):
    print(i)

# Foreach-loop (kommer gå igenom varje element i listan nedan)
myList = [1,2,3]
for num in myList:
    print(num)

# While-loop (kommer printa 1,2,3)
x = 0
while (x < 3):
    x += 1 # x++ finns inte i Python
    print(x)
```

C. Medelsvåra uppgifter

Nedan beskrivs ett antal problem, dessa behövs inte göras i någon speciell ordning. Försök att lösa så många du kan!

1. Summera alla jämna tal från 0-99

Skapa en loop som räknar upp från 0 upp till 99 (ej talet 100) och summera alla jämna tal till en variabel. Printa sedan denna variabel, resultatet bör vara 2450.

Tips: använd “modulo”, % för att hitta jämna tal.

2. Hitta största i en lista av siffror

Skriv en funktion som tar en lista med siffror som argument och returnerar det största talet i denna lista. Exempel:

```python
numbers = [1,3,5,9,2]

largest = findMax(numbers)

print("The largest number is:", largest) # Ska printa talet 9
```

3. Skriv ett program som skapar och skriver text till en fil

Skriv ett program som öppnar, skriver till och stänger en fil. Om du är osäker kan du Googla på hur man öppnar filer i Python.

Tips är att kolla på:

- open()
- write()
- close()
- with … as …

4. Skriv en rekursiv funktion som räknar ut fakulteten av ett tal

Skriv en funktion beräknar fakulteten av ett godtyckligt tal som skickas som argument till denna funktion. Denna funktion ska vara rekursiv, dvs anropa sig själv.

Förtydligande/repetition, fakultet brukar skrivas med utropstecken i matematik. Ex:

- 5! = 5*4*3*2*1 = 120
- 1! = 1
- 0! = 1

5. Skriv en funktion som genererar Fibonaccis talserie

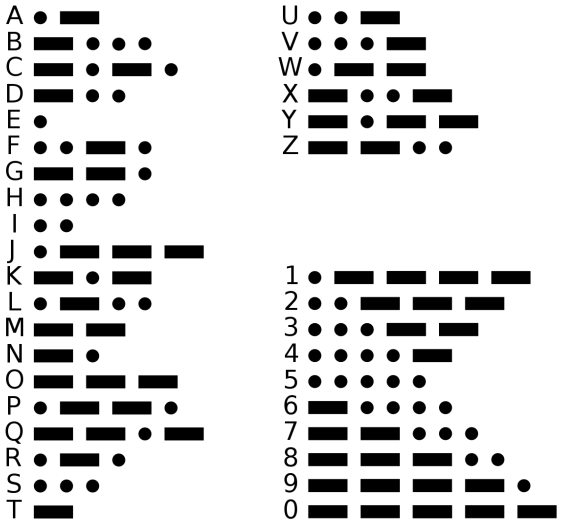
Skriv en funktion som tar in ett argument, n, och returner tal i Fibonaccis talserie upp till och med talet n som en lista. Fibonaccis talserie börjar med 0, 1 och summerar två efterföljande tal för att få nästa tal.

Ex, n = 7: [0, 1, 1, 2, 3, 5]

D. Svåra uppgifter

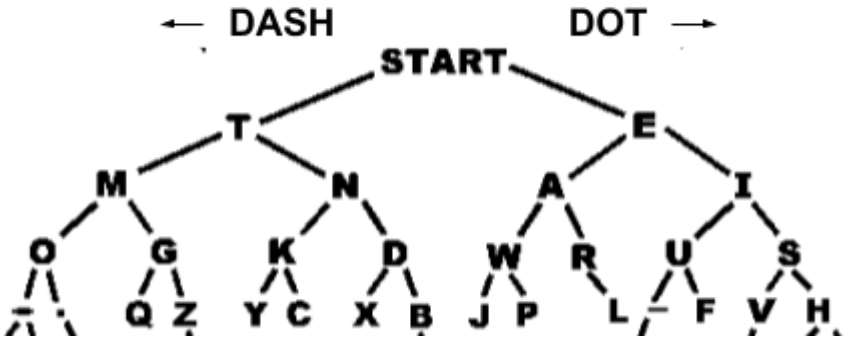
1. Skapa en funktion som översätter morsekod till bokstäver

Skapa en funktion som översätter morsekod i en sträng till bokstäver. Morsekod definieras på följande sätt där varje bokstav är uppbyggd av “dots” (.) och “dashes” (-):



För enkelhets skull kan vi säga att mellanrum i Morsekaraktärer är ett mellanslag “ “ och mellanrum i Morseord är mellanslag, slash, mellanslag “ / “.

Detta kan representeras med en trädstruktur enligt nedan, där ett steg ner till höger motsvarar “dot” och ett steg ner till vänster motsvarar “dash”.



T.ex. morsekoden “-..” (bokstaven “D”) motsvarar ett steg ner till vänster följt av två steg ner till höger, dvs att gå från START -> T -> N -> D.

Ett komplett träd går att representera som en lista i kod, förutsatt att man använder korrekta funktioner för att komma åt element i trädet.

Försök avkoda följande meddelande med ditt program:
“-.-. --- -. --. .-. .- - ..- .-.. .- - .. --- -. ... / -.-- --- ..- / ... --- .-.. ...- . -.. / .. -”

Du kan utgå från nedanstående kod för att komma igång, [länk till kodskelett med koden nedan](#).

```python
import math

charTree = [
  '-', 'T','E','M','N','A','I','O','G','K','D',
  'W','R','U','S',' ',' ','Q','Z','Y','C','X','B',
  'J','P',' ','L',' ','F','V','H'
]


def left(i):
    """ Returns index of left child"""
    return 2*i + 1


def right(i):
    """ Returns index of right child """
    return 2*i + 2


def parent(i):
    """ Returns index of parent """
    return math.floor((i-1)/2)


def isLeaf(i):
    """ Returns True if i is leaf, otherwise False"""
    if (i >= len(charTree) or charTree[i] == ' '):
        return True
    return False
```

Intressanta länkar till Morseproblemet

- Mer info om att spara träd som listor (ignorera pop-up om betyg): https://www.ida.liu.se/opendsa/Books/TDDD86F22/html/CompleteTree.html
- Morsekodsgenerator och avkodare: https://morsecode.world/international/translator.html

Fler svårare problem

Nedan finns några länkar till andra svåra problem, dessa finns dock mindre hjälp tillgänglig för, men ni kan alltid fråga oss så försöker vi lösa det åt er!

- Laborationer från första programmeringskurs: https://www.ida.liu.se/~TDDE23/planering.shtml
- Avancerade problem (ett av dessa är Morsekod-exemplet): https://www.codecademy.com/resources/blog/advanced-python-code-challenges/

Lösningsförslag

Förslag på sätt att lösa problemen finns som Pythonfiler att ladda ner för del B och C.

- Svar för del B: https://viktorronnback.github.io/mafu/svar_delB.py
- Svar för del C: https://viktorronnback.github.io/mafu/svar_delC.py