Föreläsning 10

DD1315

Programmeringsteknik

7,5 hp

Innehåll

- Gott & blandat (samt inbördes orelaterat)
  - Filhantering (att skriva till filer)
  - Felhantering
  - Formaterad utskrift
  - Teckenkodning
  - main()-metod

Filhantering (att skriva till filer)

- Att skriva till fil kan liknas vid att göra en skärmutskrift med *print()* med skillnaden att utmatningen istället hamnar på fil.
- För att öppna en fil att skriva till används (finns redan filen raderas den och finns den inte skapas den):
  - *utfil = open('filnamn.txt','w')*
- För att sedan skriva till denna (från dess början):
  - *utfil.write('En godtycklig sträng\n')*
- När man sedan är klar:
  - *utfil.close()*

Exempel

```
utfil = open('utfil.txt','w')

print('Mata in något, avsluta med blankrad.')
s = input('>')
while s != '':
    utfil.write(s + '\n')
    s = input('>')

utfil.close()
```

Felhantering

- Används för att undvika att ett program kraschar i olika situationer
- Det finns många fördefinierade *Error* i Python t ex
  - *ValueError*: Vid ej möjlig konvertering mellan datatyper
  - *NameError*: I uttryck innehållandes ej deklarerad variabel
  - *IndexError*: När ett element inte existerar för ett givet index i en lista
- När ett fel uppstår kan man välja att “fånga det” för att undvika att programmet kraschar (=exekveringsfel).
  - Med *try:* markerar man början av ett block som innehåller den kod som kan orsaka fel.
  - Efter detta block följer *except:* där man i ett nytt block vidtar åtgärder och fel inträffade under *try*-blocket, p s s slipper man exekveringsfel.

Exempel

```
fel = True
while fel:
    fel = False
    tempText = input('Ange temperatur? ')
    try:
        tempTal = float(tempText)
    except ValueError:
        print ('Det här är inget flyttal!')
        fel = True
```

Formaterad utskrift

- Med formaterad utskrift menas möjligheten att kunna välja bl a höger/vänsterjustering, teckenbredd et c. Oftast ren estetik i utmatningen.
- Det finns flera sätt att göra formatering:
  - med *%d %f %s*
    - ser likadan ut i flertalet språk
  - genom *str*-metoder
    - pythonspecifik

Exempel

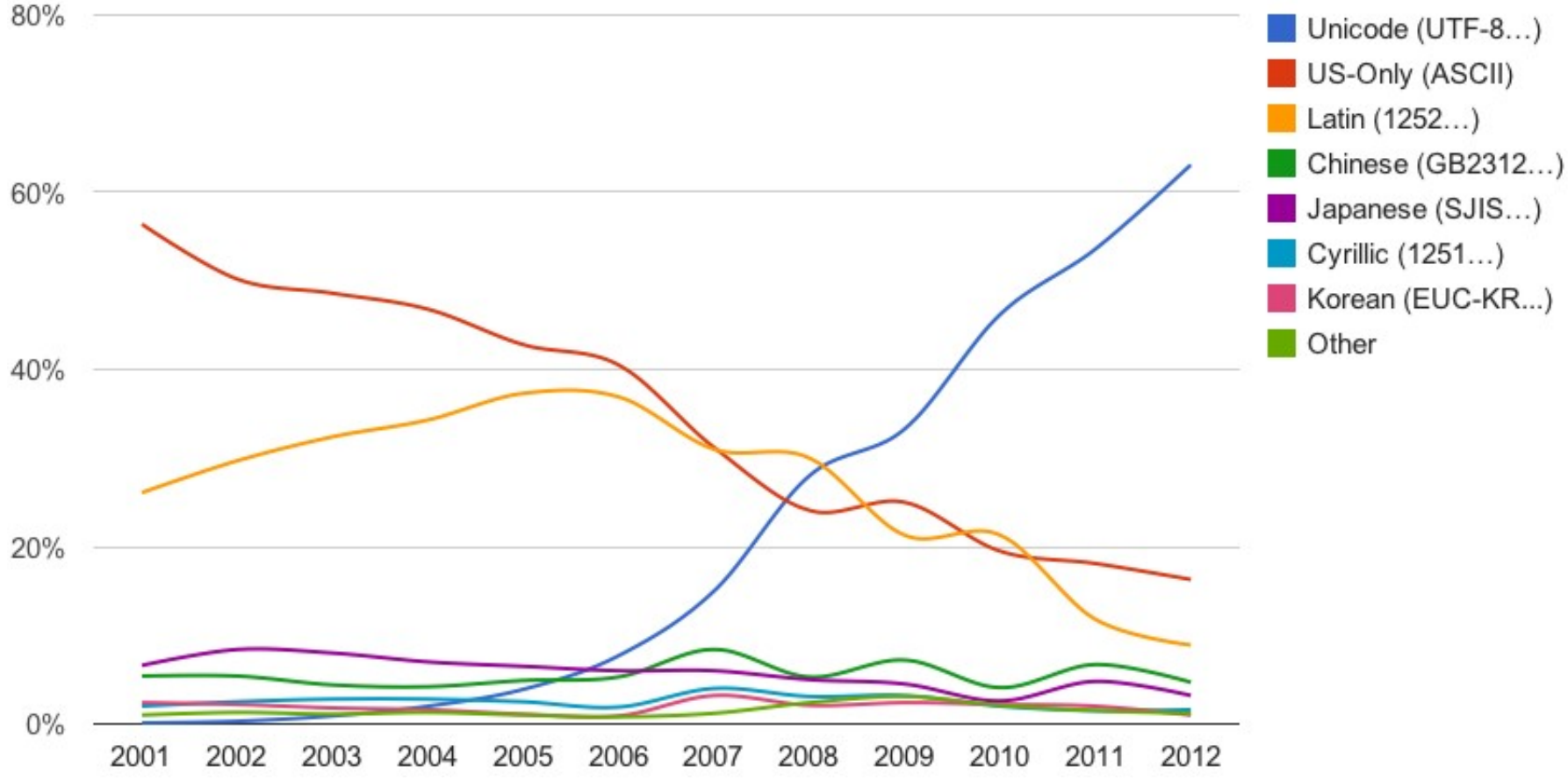
```
import math

print('Konstanten pi avrundad till 5 decimaler, vidd 10, högerjusterad: %10.5f' % math.pi)
print('Konstanten pi avrundad till 5 decimaler, vidd 10, vänsterjusterad: %-10.5f' % math.pi)
print('Heltalet 128 med vidd 10, högerjusterad: %10d' % 128)
print('Heltalet 128 med vidd 10, vänsterjusterad: %-10d' % 128)
print('Strängen %10s med vidd 10, högerjusterad' % "Meny")
print('Strängen %-10s med vidd 10, vänsterjusterad' % "Meny")

for i in range (1,11):
    print ('%-10d %10d' % (i, math.pow(2,i)) )
```

Teckenkodning

Alla tecken i filer lagras som siffror och med tiden har allt mer inkluderande (tecken på olika språk) teckenkodningssystem utvecklats

- ASCII (1960-tal)
- latin-1 (1980-tal)
- utf-8 (1990-tal och framåt)

Teckenkodning, forts



main()-metod

När man anropar en funktion ska samtliga indata till dessa skickas med som parametrar och samtliga utdata ska ingå i en *return*-sats. Detta för att funktioner ska gå att använda utan att man vet hur de är skrivna (jmf inbyggda funktionerna i python) och att koden då blir portabel. För att se till att man gör så kan man skriva en *main()*-metod där man lägger den kod som tidigare var ”huvudprogrammet” och sedan anropar denna med *main()*