Föreläsning 5 Programmeringsteknik och C DD1316

- Exekveringsfel vs logiska fel
- Kort introduktion till klasser
- Repetition av det vi lärt oss om funktioner
- Parametrar, default-parametrar
- Lokala och globala variabler
- Retursats
- None
- Abstraktion mha funktioner

Exekveringsfel och logiskt fel

- Exekveringsfel: ett fel som gör att programmet avbryts med felmeddelande innan det hinner att avslutas
- Logiskt fel: Programmet *avslutas* men felaktig utdata kommer ut (denna typ av fel är svårast att spåra).

Exempel på logiskt fel

```
def BMI(height, weight):
    return weight /(height*height)
```

Vilket av följande anrop är fel?

`BMI(1.70,65)` 22.49	`BMI(65,1.70)` 0.000402

Exemplet smurf.py

Introduktion till klasser

Funktioner

Syntax:

parametrar

```
def funktionensnamn(x, y, ...):
```

Kod som ska exekveras när funktionen anropas

Indragning är viktigt!

Exempel

```
def meny():
    print("""1. Spela
2. Visa spelregler
3. Avsluta
Välj:""", end="")
```

```
meny()
```

anrop

definition

Parametrar

Genom att parametrisera en funktion kan den göras mer generellt användbar:

```
def medel23():
  print((2+3)/2)

medel23()
```

```
def medel(a,b):
  print((a+b)/2)

medel(2, 3)
```


`return`

En funktion med parametrar som ***returnerar*** ett värde har ännu större användningsområde (är mer flexibel).

```
def medel(a,b):
  print((a+b)/2)

medel(2,3)
```

```
def medel(a,b):
  return (a+b)/2

m=medel(2,3)
print(m)
```

Funktionen `medel` till höger kan användas i kod där man inte vill ha utskrift.

Parametrar, `return` och `None`

- Indata skickas till funktioner via funktionens parametrar.
- Funktioner returnerar utdata med hjälp av return-satsen.
- Om en funktion inte har return-sats i kroppen kommer då funktionen att returnera None. *(None **betyder ingenting i python)***

Exempel

```
def medel(a,b):
    m = (a+b) / 2
    return m

x=2
y=3
z=medel(x, y)
```

Variabeln z kommer att få värdet 2.5 vilket är av typen float

```
def medel(a,b):
  m = (a+b) / 2


x=medel(2, 3)
```

Variabeln x kommer att få värdet `None`

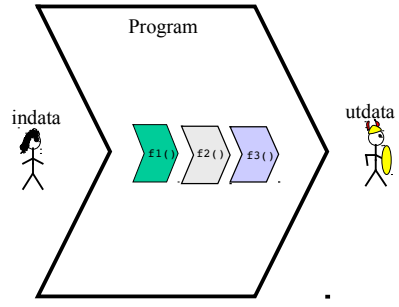
Exempel

```
def isGreater(a,b):
    return a > b

m=isGreater(2 , 3)
```

Variabeln m kommer att få värdet False vilket är boolean

Program



Exemplet impedance.py

Diskussion om nedbrytning av ett problem i oberoende funktioner samt kring abstraktion (att dölja komplexitet med funktionsanrop)

Globala och lokala variabler

En variabel kan antingen vara global eller lokal, kan alltså inte vara båda samtidigt:

- Globala variabler: åtkomliga i hela programmet och lever tills programmet avslutas:
  - Variabler utanför funktionsdefinitioner är globala.
  - I en funktionsdefinition markeras globala variabler med nyckelordet `global`.
- Lokal variabel: variabel som oftast lever en kort stund.
  - Formella parametrar
  - Lokala variabler som skapas i funktioner
  - En lokal variabel är känd endast i funktionen och har kortare livslängd än en global variabel

Exempel

Följande kod ger fel:

```
def change():
    x = x/2

x=10
change()
```

Följande kod är korrekt:

```
def change():
    global x
    x = x/2
x = 10
print(x)
change()
print(x)
```


Använd inte globala variabler!

- Vi tar upp globala variabler i kursen därför att ni bör känna till dem
- De behöver mycket sällan användas eftersom relevanta datastrukturer kan passas som argument/parametrar till funktioner
- I labbarna är de oftast ett tecken på dålig programstruktur

Defaultparametrar: Exempel

```
def correction(penColor = ”röd”):
    print (”Tentamen har rättats med ”
          + penColor + ” penna”)
```

Kan anropas på följande sätt:

- utan argument: `correction()`
- med argument: `correction(”svart”)`

Default-parametrar

- Vid definition av en funktion kan man ge formella parametrar ett default-värde. De parametrar som får ett default-värde kallas för default-parametrar.
- Argument till en funktion med default-parametrar kan utelämnas.
- Funktionen använder sig av default-parametrars värde endast när argumentet utelämnas vid anropet.
- Om man vill blanda default-parametrar med vanliga parametrar måste man se till att default-parametrar kommer *senare* i listan på formella parametrar.

Vilket av följande är fel?

```
def correctEssay(penColor=”röd”, type):
  print(type + ” har rättats med” +
  penColor + ” penna”)
```

```
def correcEssay(type, penColor=”röd”):
   print(type + ” har rättats med”+
   penColor + ” penna”)
```


Sammanfattning

- Genom att deklarera egna funktioner inför man abstraktion i sitt program, vilket gör att programmet blir lättare att läsa och förstå, och dessutom att koden blir mer överskådlig
- Man kan undvika kodupprepning genom att deklarera egna funktioner
- Se till att dina funktioner är fristående och självständiga
- Minimera beroendet mellan funktioner! Beroende mellan funktioner gör det svårt att modifiera programmet
- Undvik att använda globala variabler