Programmeerimine

5. loeng

Täna loengus

- Funktsioonid ja protseduurid
- Funktsioonide defineerimine
  - Protseduurid
  - Parameetrid
  - Väärtuse tagastamine
- Lokaalne vs. globaalne skoop

Funktsioonid ja protseduurid

- Protseduur on suhteliselt iseseisev korduvkasutatav programmiosa, mis koosneb käskudest ja mida on võimalik välja kutsuda tema nime kaudu.
- Protseduur võib olla parametriseeritud argumentidega, millede konreetsed väärtused määratakse protseduuri väljakutsel.
- Funktsioon on protseduur, mis tagastab väärtuse.
- Funktsiooni rakendus on avaldis, mille väärtust võib kasutada sarnaselt teiste (aritmeetilised-, loogilised-, jne) avaldistega.
- Ilma väärtuseta protseduuri saab kasutada lausena ning tema "toimeks" on kõrvalefekt programmi olekule.
- Objektorienteeritud keeltes (näit. Java) on protseduurid seotud mingi klassiga, misjuhul neid nimetatakse meetodideks.

Funktsioonid

- Funktsioonide defineerimine Pythonis:

  $\textbf{def}\ funcName(arg_1,\ldots,arg_n)$:
  $\quad funcBody$

- $funcName$ on funktsiooni nimi.
- arg_i on funktsiooni formaalsed parameetrid.
- $funcBody$ on funktsiooni keha, mis koosneb lausetest.
- Funktsiooni väljakutsel antakse formaalsetele parameetritele tegelike argumentide väärtused.
- Formaalsed parameetrid ja funktsiooni kehas defineeritud muutujad on funktsioonis lokaalsed; so. on nähtavad ainult selle funktsiooni kehas.

Protseduurid

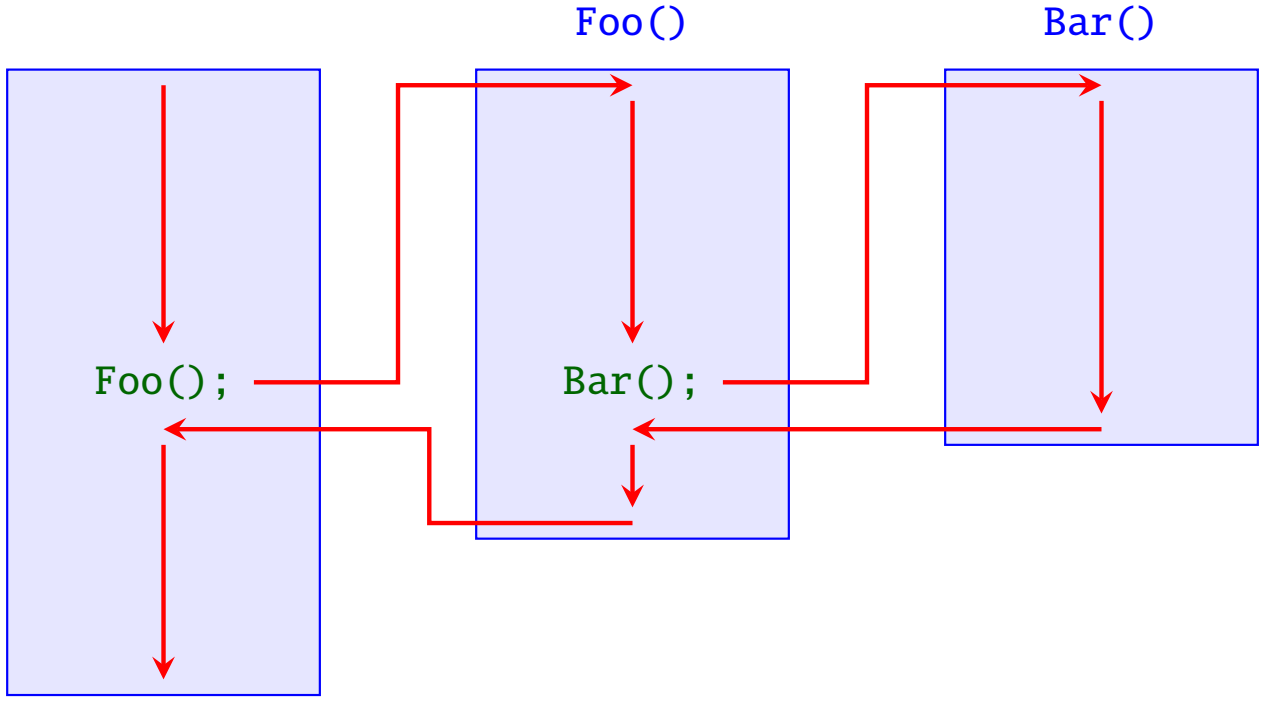
Näide: tervitus (ver. 1)

```
def tervitus ():
    print('Tere!')
    print('Kuidas käsi käib?')

tervitus ()
tervitus ()
```

```
Tere!
Kuidas käsi käib?
Tere!
Kuidas käsi käib?
```

Kontrollvoog funktsioonide väljakutsetel



Parameetrid

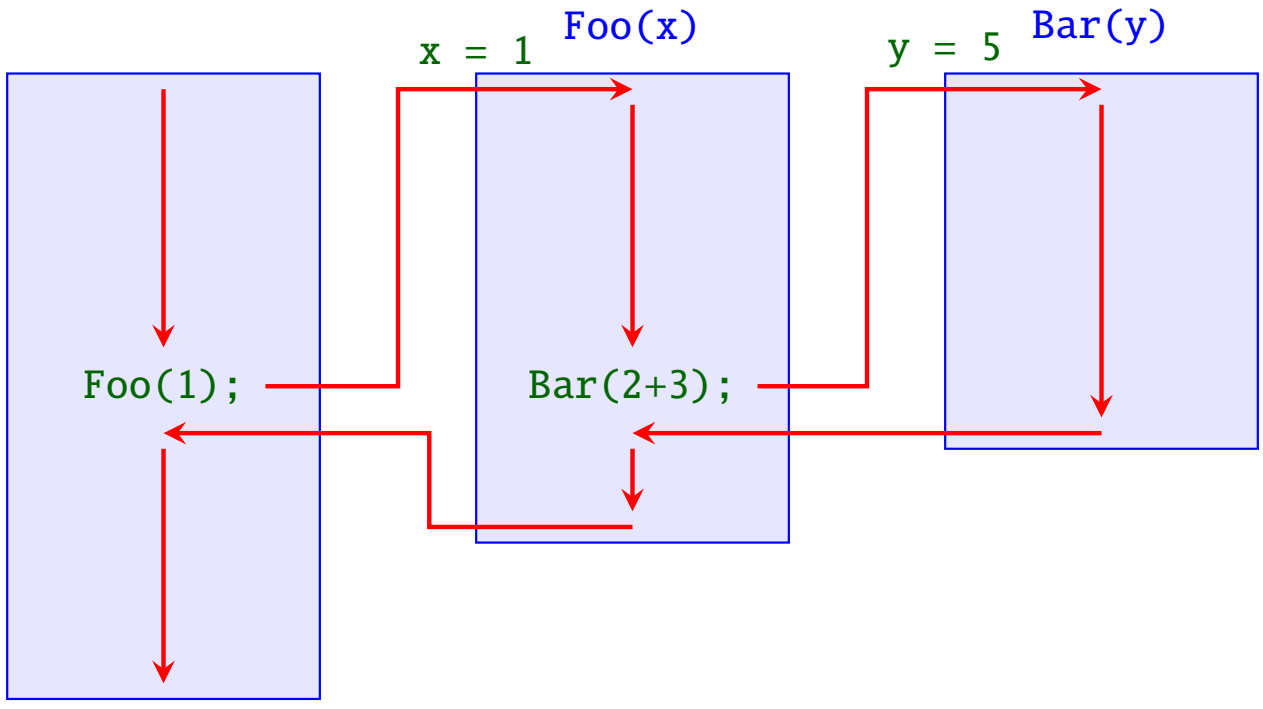
Näide: tervitus (ver. 2)

```
def tervitus(nimi):
    print('Tere ' + nimi + '!')
    print('Kuidas käsi käib?')

tervitus('Mari')
tervitus('Jaan')
```

```
Tere Mari!
Kuidas käsi käib?
Tere Jaan!
Kuidas käsi käib?
```

Parameetrite edastamine



Parameetrid

Näide: tervitus (ver. 3)

```
def tervitus(nimi, keel):
    if keel == 'et':
        print('Tere ' + nimi + '!')
        print('Kuidas käsi käib?')
    else:
        print('Hello ' + nimi + '!')
        print('How do you do?')

tervitus('Mari', 'et')
tervitus('Jaan', 'en')
```

```
Tere Mari!
Kuidas käsi käib?
Hello Jaan!
How do you do?
```

Parameetrid

Näide: tervitus (ver. 4)

```
def tervitus(nimi, keel='et'):
    if keel == 'et':
        print('Tere ' + nimi + '!')
        print('Kuidas käsi käib?')
    else:
        print('Hello ' + nimi + '!')
        print('How do you do?')

tervitus('Mari')
tervitus('Jaan', 'en')
```

```
Tere Mari!
Kuidas käsi käib?
Hello Jaan!
How do you do?
```

Väärtuste tagastamine

- **return** lause lõpetab funktsiooni töö;
- kui tal on argument, siis selle väärtus on funktsiooni väljakutse väärtuseks.

Näited

```
def liida2(arv):
    return arv+2

def summa(arv1, arv2):
    return arv1+arv2


x = liida2(3)                  # x = 5
y = summa(liida2(1), 3)        # y = 6
z = liida2(liida2(0))          # z = 4
```

Väärtuste tagastamine

- Kui **return** lausel argument puudub või funktsioonil üldse **return** lause puudub, siis on funktsiooni väljakutse väärtus **None**.

Näited

```
def tere():
    print('Tere!')
    return

def hyvasti()
    print('Hüvasti.')

x = tere()              # Trykib:   Tere!
                        # x = None
y = hyvasti()           # Trykib:   Hüvasti.
                        # y = None
```

Funktsioonid

Näide: keskmise arvutamine

```
def keskmine(arv1, arv2):
    k = (arv1 + arv2) / 2
    return k

x = keskmine(3,4)          # x = 3.5
```

Näide: absoluutväärtus

```
def absoluut(arv):
    if arv < 0:
        return -arv
    return arv

x = absoluut(-3)           # x = 3
y = absoluut(2)            # y = 2
```

Funktsioonid


```
def keskmine(arv1, arv2):
    k = (arv1 + arv2)/2
    return k
```

$arv = 7$
$arv = keskmine(arv + 1, 2)$
$arv = keskmine(arv, arv - 2)$

Funktsioonid

```
def keskmine(arv1, arv2):
    k = (arv1 + arv2)/2
    return k
```


$arv = 7$

$arv = keskmine(arv + 1, 2)$

$arv = keskmine(arv, arv - 2)$

Funktsioonid

```
def keskmine(arv1, arv2):
    k = (arv1 + arv2)/2
    return k
```

```
arv = 7
arv = keskmine(arv + 1, 2)
arv = keskmine(arv, arv - 2)
```


arv | 7

Funktsioonid


```
def keskmine(arv1, arv2):
    k = (arv1 + arv2)/2
    return k
```

arv1	8
arv2	2

$arv = 7$
$arv = keskmine(arv + 1, 2)$
$arv = keskmine(arv, arv - 2)$

arv	7

Funktsioonid

```
def keskmine(arv1, arv2):
    k = (arv1 + arv2)/2
    return k
```


arv1	8
arv2	2
k	5.0

```
arv = 7
arv = keskmine(arv + 1, 2)
arv = keskmine(arv, arv - 2)
```

arv	7

Funktsioonid

```
def keskmine(arv1, arv2):
    k = (arv1 + arv2)/2
    return k
```

```
arv = 7
arv = keskmine(arv + 1, 2)
arv = keskmine(arv, arv - 2)
```


arv 5.0

Funktsioonid


```
def keskmine(arv1, arv2):
    k = (arv1 + arv2)/2
    return k
```

arv1	5.0
arv2	3.0

```
arv = 7
arv = keskmine(arv + 1, 2)
arv = keskmine(arv, arv - 2)
```

arv	5.0

Funktsioonid

```
def keskmine(arv1, arv2):
    k = (arv1 + arv2)/2
    return k
```


arv1	5.0
arv2	3.0
k	4.0

```
arv = 7
arv = keskmine(arv + 1, 2)
arv = keskmine(arv, arv - 2)
```

arv	5.0

Funktsioonid

```
def keskmine(arv1, arv2):
    k = (arv1 + arv2)/2
    return k
```

```
arv = 7
arv = keskmine(arv + 1, 2)
arv = keskmine(arv, arv - 2)
```


arv 4.0

Funktsioonid

Näide: mitu resultaati tagastav funktsioon

```
def sumDiff(x, y):
    s = x + y
    d = x - y
    return s, d

s, d = sumDiff(5, 3)
print('Arvude 5 ja 3 summa on', s, 'ning vahe on', d)
```

Funktsioonid

- Funktsiooni keha võib alata sõnega, mis annab funktsiooni lühikirjelduse (*ingl. docstring*).
  - Vastav sõne seotakse funktsiooniobjekti atribuudiga *__doc__*.

Näide – Fahrenheiti skaalast temperatuuri teisendamine

```
def fahrenheit2Celsius(t):
    """Teisendab temperatuuri Fahrenheiti skaalast
       Celsiuse skaalasse. Näiteks:

       >>> fahrenheit2Celsius(70)
       21
    """
    return round(5*(t-32)/9)
```

Suur tänu osalemast

ja

kohtumiseni!