Rekursioon

11. Tahvlipraktikum

Rekursioon

- defineeritav funktsioon kutsub välja iseennast
- võimaldab kirjeldada korduvtäidetavaid protsesse
- võimsam kui iteratsioon

Faktoriaal

$$n! = n \cdot (n - 1) \cdot \ldots \cdot 3 \cdot 2 \cdot 1$$

$$0! = 1$$

Iteratiivne faktoriaal

$$n! = n \cdot (n - 1) \cdot \ldots \cdot 3 \cdot 2 \cdot 1$$

$$0! = 1$$

```
def factorial(n):
    f = 1
    for i in range(1, n + 1):
        f *= i
    return f
```

Rekursiivne faktoriaal

$$n! = n \cdot (n - 1) \cdot \ldots \cdot 3 \cdot 2 \cdot 1$$

$$0! = 1$$

```
def factorial(n):
    if n == 0:
        return 1
    else:
        return n * factorial(n - 1)
```

Rekursiivne faktoriaal

```
def factorial(n):
  if n == 0:
    return 1
  else:
    return n * factorial(n - 1)
```

```
factorial(3) ->
```

Rekursiivne faktoriaal

```
def factorial(n):
    if n == 0:
        return 1
    else:
        return n * factorial(n - 1)
```

```
factorial(3) ->
    3 * factorial(2) ->
```

Rekursiivne faktoriaal

```
def factorial(n):
  if n == 0:
    return 1
  else:
    return n * factorial(n - 1)
```

```
factorial(3) ->
  3 * factorial(2) ->
    3 * 2 * factorial(1) ->
```

Rekursiivne faktoriaal

```
def factorial(n):
  if n == 0:
    return 1
  else:
    return n * factorial(n - 1)
```

```
factorial(3) ->
  3 * factorial(2) ->
    3 * 2 * factorial(1) ->
      3 * 2 * 1 * factorial(0) ->
```

Rekursiivne faktoriaal

```
def factorial(n):
  if n == 0:
    return 1
  else:
    return n * factorial(n - 1)
```

```
factorial(3) ->
  3 * factorial(2) ->
    3 * 2 * factorial(1) ->
      3 * 2 * 1 * factorial(0) ->
      3 * 2 * 1 * 1 ->
```

Rekursiivne faktoriaal

```
def factorial(n):
  if n == 0:
    return 1
  else:
    return n * factorial(n - 1)
```

```
factorial(3) ->
  3 * factorial(2) ->
    3 * 2 * factorial(1) ->
      3 * 2 * 1 * factorial(0) ->
      3 * 2 * 1 * 1 ->
    3 * 2 * 1 ->
```

Rekursiivne faktoriaal

```
def factorial(n):
  if n == 0:
    return 1
  else:
    return n * factorial(n - 1)
```

```
factorial(3) ->
  3 * factorial(2) ->
    3 * 2 * factorial(1) ->
      3 * 2 * 1 * factorial(0) ->
      3 * 2 * 1 * 1 ->
    3 * 2 * 1 ->
  3 * 2 ->
```

Rekursiivne faktoriaal

```
def factorial(n):
  if n == 0:
    return 1
  else:
    return n * factorial(n - 1)
```

```
factorial(3) ->
  3 * factorial(2) ->
    3 * 2 * factorial(1) ->
      3 * 2 * 1 * factorial(0) ->
      3 * 2 * 1 * 1 ->
    3 * 2 * 1 ->
  3 * 2 ->
6
```

Rekursioon

- rekursiooni baas
  - rekursiivse funktsiooni keha haru, mis on ilma rekursiivse väljakutseta
  - vajalik, et rekursioon termineeruks

```
def factorial(n):
  if n == 0:
    return 1
  else:
    return n * factorial(n - 1)
```

Rekursioon

- mittetermineeruv rekursioon

```
def factorial(n):
  return n * factorial(n - 1)
```

```
factorial(3) ->
  3 * factorial(2) ->
    3 * 2 * factorial(1) ->
      3 * 2 * 1 * factorial(0) ->
        3 * 2 * 1 * factorial(-1) ->
          ...
```

Rekursioon

- rekursiivsetes funktsioonides võivad “tegevused” toimuda enne ja/või pärast rekursiivset väljakutset

```
def countdown(n):
  if n <= 0:
    print("Start!")
  else:
    print(n)
    countdown(n - 1)
```

Rekursioon

```
def countdown(n):
  if n <= 0:
    print("Start!")
  else:
    print(n)
    countdown(n - 1)
```

```
# countdown(4) trükib:
4   # countdown(3)
```

Rekursioon

```
def countdown(n):
  if n <= 0:
    print("Start!")
  else:
    print (n)
    countdown(n - 1)
```

```
# countdown(4) trükib:
4
3   # countdown(2)
```

Rekursioon

```
def countdown(n):
  if n <= 0:
    print("Start!")
  else:
    print (n)
    countdown(n - 1)
```

```
# countdown(4) trükib:
4
3
2   # countdown(1)
```

Rekursioon

```
def countdown(n):
  if n <= 0:
    print("Start!")
  else:
    print (n)
    countdown(n - 1)
```

```
# countdown(4) trükib:
4
3
2
1   # countdown(0)
```

Rekursioon

```
def countdown(n):
  if n <= 0:
    print("Start!")
  else:
    print (n)
    countdown(n - 1)
```

```
# countdown(4) trükib:
4
3
2
1
Start!
```

Rekursioon

- rekursiivsetes funktsioonides võivad “tegevused” toimuda enne ja/või pärast rekursiivset väljakutset

```
def countup(n):
  if n <= 0:
    print("Start!")
  else:
    countup(n - 1)
    print (n)
```

Rekursioon

```
def countup(n):
  if n <= 0:
    print("Start!")
  else:
    countup(n - 1)
    print (n)
```

```
# countdown(4) trükib:
Start!
1
2
3
4
```

Ülesanne 1

Kirjutada rekursiivne funktsioon, mis väljastab arve kahanevalt ning seejärel kasvavalt:

```
...
3
2
1
Start!
1
2
3
...
```

Ülesanne 2

Kirjutada rekursiivne funktsioon astendamiseks:

```
def aste(baas, eksponent):
```

Ülesanne 1 - lahendus

Kirjutada rekursiivne funktsioon, mis väljastab arve kahanevalt ning seejärel kasvavalt:

```
def counter(n):
  if n <= 0:
    print("Start!")
  else:
    print(n)
    counter(n - 1)
    print n
```

Ülesanne 2 - lahendus

Kirjutada rekursiivne funktsioon astendamiseks:

```
def aste(baas, eksponent):
  if eksponent == 0:
    return 1
  else:
    return baas * aste(baas, eksponent - 1)
```

Ülesanne 2 - lahendus

```
def aste(baas, eksponent):
    if eksponent == 0:
        return 1
    else:
        return baas * aste(baas, eksponent - 1)
```

```
aste(2, 4) ->
    2 * aste(2, 3) ->
        2 * 2 * aste(2, 2) ->
            2 * 2 * 2 * aste(2, 1) ->
                2 * 2 * 2 * 2 * aste(2, 0) ->
                2 * 2 * 2 * 2 * 1 ->
            2 * 2 * 2 * 2 ->
        2 * 2 * 4 ->
    2 * 8 ->
16
```

Ülesanne 3

Kirjutada rekursiivne funktsioon, mis väljastab kõik positiivsed paaris täisarvud alates väikseimast kuni etteantud positiivse täisarvuni.

```
def paaris(n):
```

Ülesanne 4

Kirjutada rekursiivne funktsioon, mis väljastab kõik positiivsed täisarvud alates väikseimast kuni etteantud positiivse täisarvuni, mis jaguvad teise etteantud positiivse täisarvuga.

```
def jaguvad(n, jagaja):
```

Ülesanne 5

Kirjutada rekursiivne funktsioon, mis väljastab ekraanile etteantud arvust võrdusmärkidest koosneva joonekese.

Ülesanne 6

Kirjutada rekursiivne funktsioon, mis väljastab ekraanile etteantud suurusega tärnikolmnurga.

Ülesanne 7

Kirjutada rekursiivne funktsioon, mis väljastab ekraanile etteantud suurusega ja etteantud sümbolitest koosneva kolmnurga.

Sabarekursioon

Kui kõik tegevused toimuvad enne rekursiivset väljakutset, siis sellist rekursiooni nimetatakse sabarekursiooniks.

```
def factorialTailRec(n, f):
  if n == 0:
    return f
  else:
    return factorialTailRec(n - 1, f * n)

def factorial(n)
  return factorialTailRec(n, 1)
```

Sabarekursioon

```
def factorialTailRec(n, f):
  if n == 0: return f
  else: return factorialTailRec(n - 1, f * n)

def factorial(n)
  return factorialTailRec(n, 1)
```

```
factorial(3)->
  factorialTailRec(3, 1) ->
```

Sabarekursioon

```
def factorialTailRec(n, f):
  if n == 0: return f
  else: return factorialTailRec(n - 1, f * n)

def factorial(n)
  return factorialTailRec(n, 1)
```

```
factorial(3)->
  factorialTailRec(3, 1) ->
    factorialTailRec(3 - 1, 1 * 3) ->
    factorialTailRec(2, 3) ->
```

Sabarekursioon

```
def factorialTailRec(n, f):
  if n == 0: return f
  else: return factorialTailRec(n - 1, f * n)

def factorial(n)
  return factorialTailRec(n, 1)
```

```
factorial(3)->
  factorialTailRec(3, 1) ->
    factorialTailRec(3 - 1, 1 * 3) ->
    factorialTailRec(2, 3) ->
      factorialTailRec(2 - 1, 3 * 2) ->
      factorialTailRec(1, 6) ->
```

Sabarekursioon

```
def factorialTailRec(n, f):
  if n == 0: return f
  else: return factorialTailRec(n - 1, f * n)

def factorial(n)
  return factorialTailRec(n, 1)
```

```
factorial(3)->
  factorialTailRec(3, 1) ->
    factorialTailRec(3 - 1, 1 * 3) ->
    factorialTailRec(2, 3) ->
      factorialTailRec(2 - 1, 3 * 2) ->
      factorialTailRec(1, 6) ->
        factorialTailRec(1 - 1, 6 * 1) ->
        factorialTailRec(0, 6) -> 6
```

Ülesanne 8

Kirjutada rekursiivne funktsioon, mis väljastab ekraanile etteantud jada elemendid.

Ülesanne 9

Kirjutada rekursiivne funktsioon, mis asendab jadas kõik negatiivsed elemendid nende absoluutväärtusega (kasutada sabarekursiooni).

Ülesanne 10

Kirjutada rekursiivne funktsioon, mis asendab jadas kõik negatiivsed arvud -1 -ga ja kõik positiivsed arvud 1-ga (kasutada sabarekursiooni).