

Programmeerimine

15. loeng

Täna loengus

- Sabarekursioon vs. iteratsioon
- Erinevad rekursiooniskeemid
- Näiteid:
 - Hanoi tornide ülesanne
 - Puu joonistamine
 - Sierpinski kolmnurk

Sabarekursioon

- Kui kõik tegevused toimuvad enne rekursiivset väljakutset, nimetatakse rekursiooni **sabarekursiooniks**.
- Sabarekursiooni on väga lihtne teisendada tavaliseks tsüklikks (ja ka vastupidi).

Sabarekursioon

- Kui kõik tegevused toimuvad enne rekursiivset väljakutset, nimetatakse rekursiooni **sabarekursiooniks**.
- Sabarekursiooni on väga lihtne teisendada tavaliseks tsüklikks (ja ka vastupidi).

Rekursiivne faktoriaal

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

Faktoriaal

`factorial(5)`

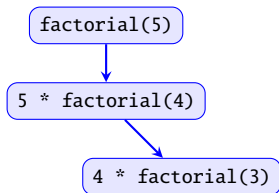
Faktoriaal

`factorial(5)`

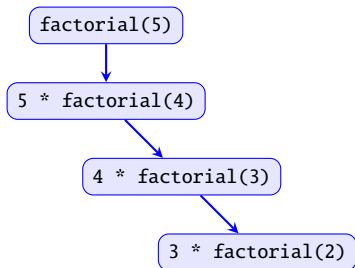


`5 * factorial(4)`

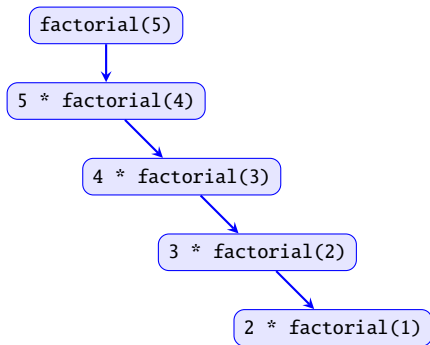
Faktoriaal



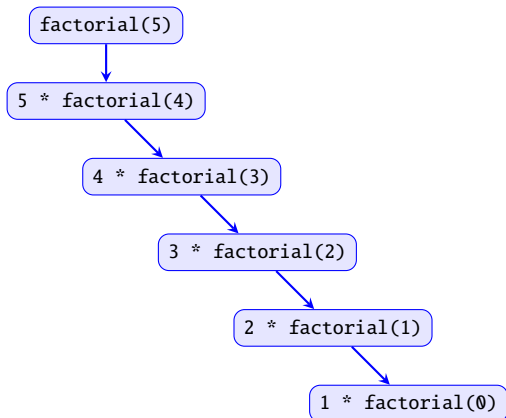
Faktoriaal



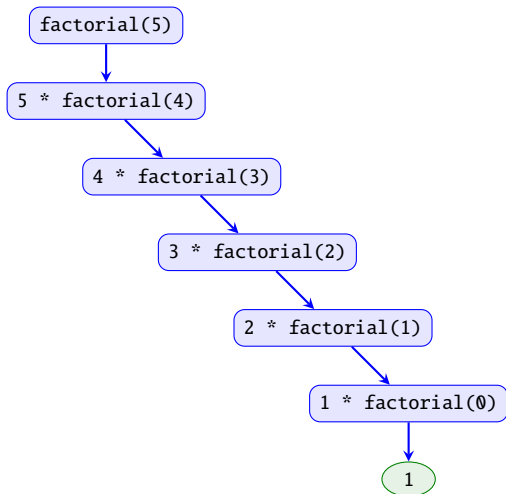
Faktoriaal



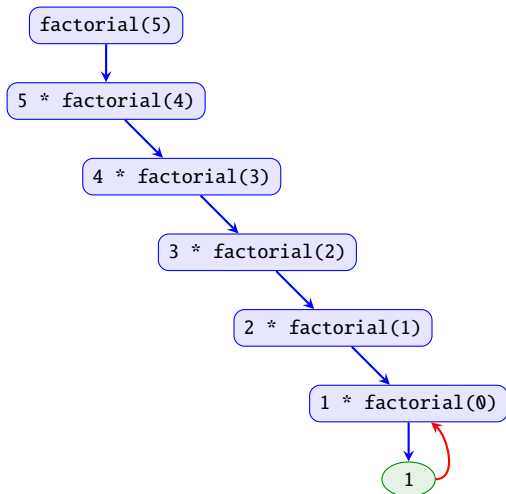
Faktoriaal



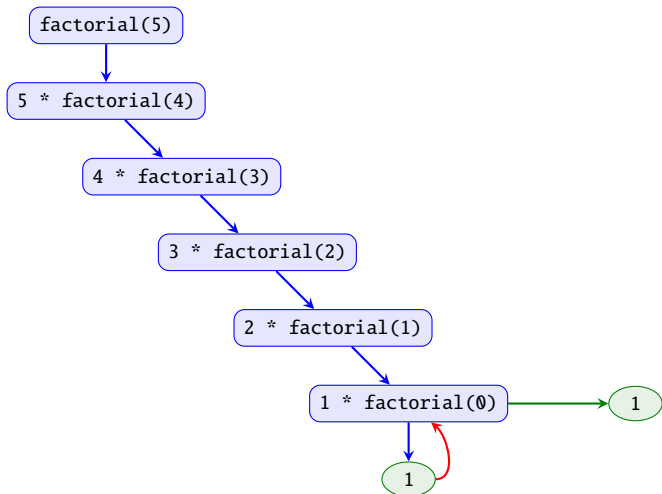
Faktoriaal



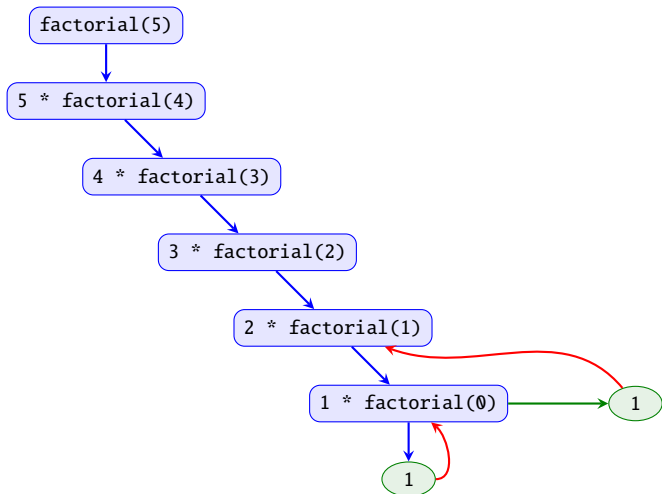
Faktoriaal



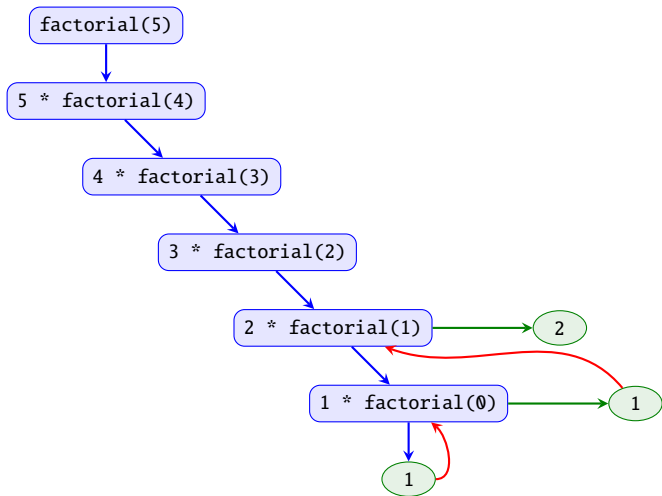
Faktoriaal



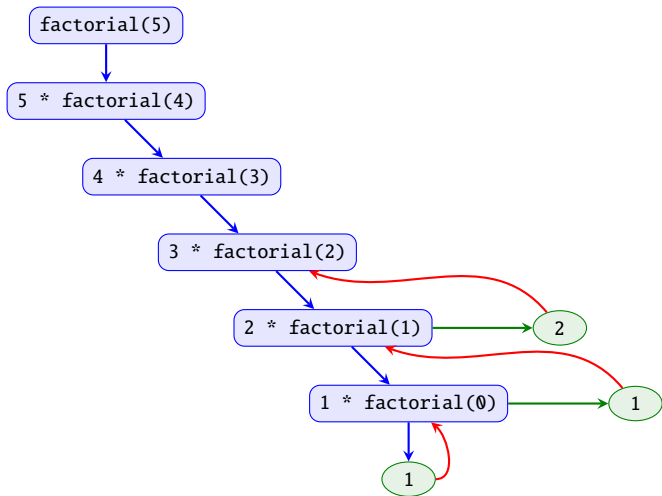
Faktoriaal



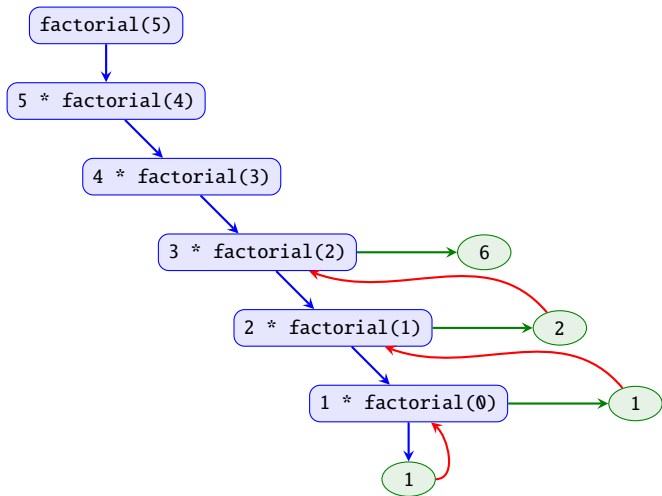
Faktoriaal



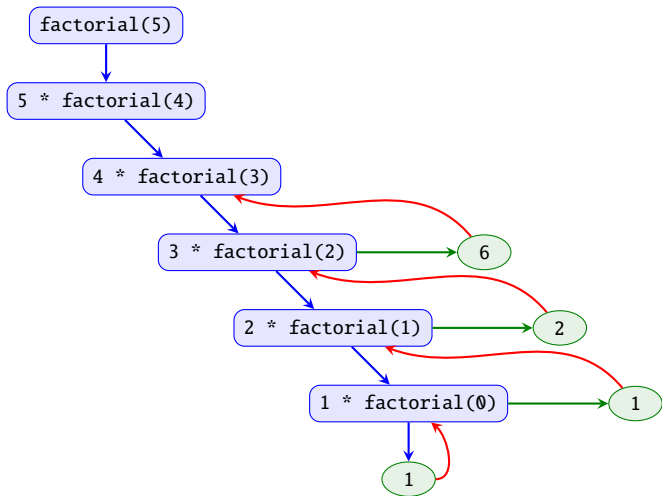
Faktoriaal



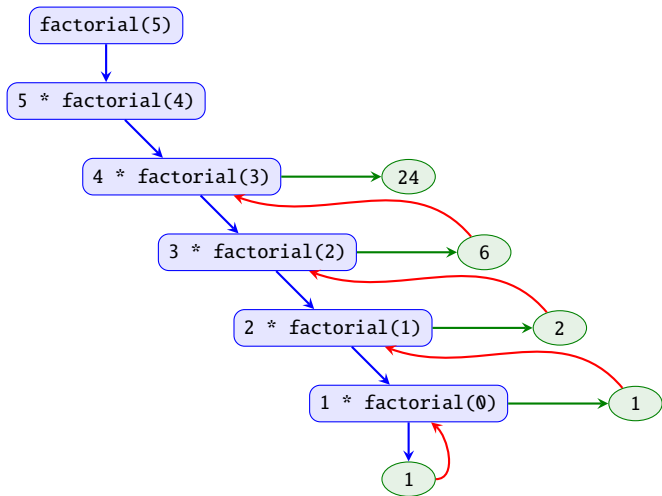
Faktoriaal



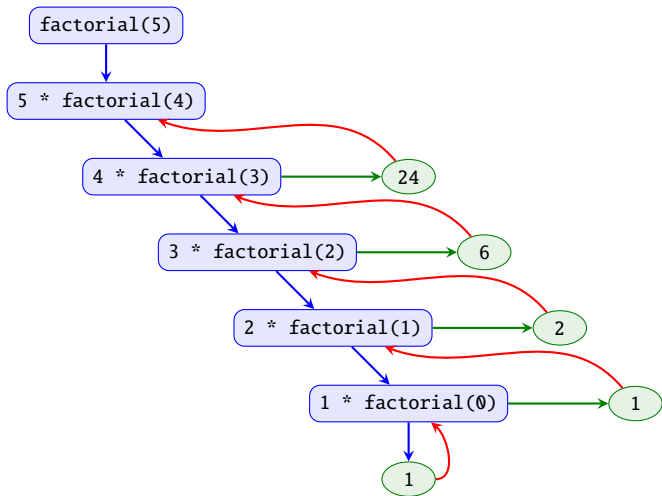
Faktoriaal



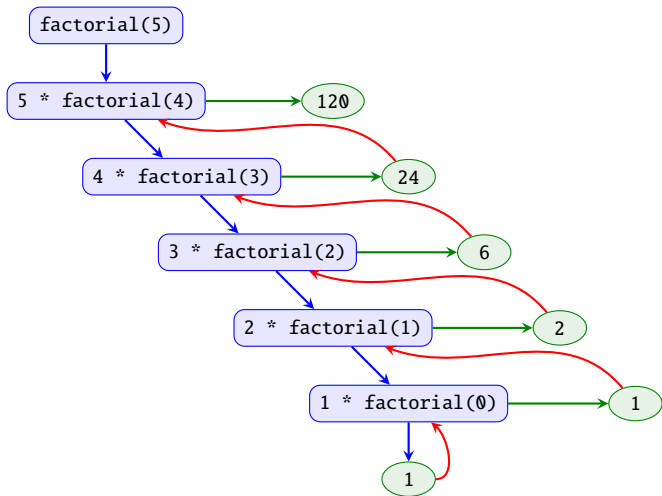
Faktoriaal



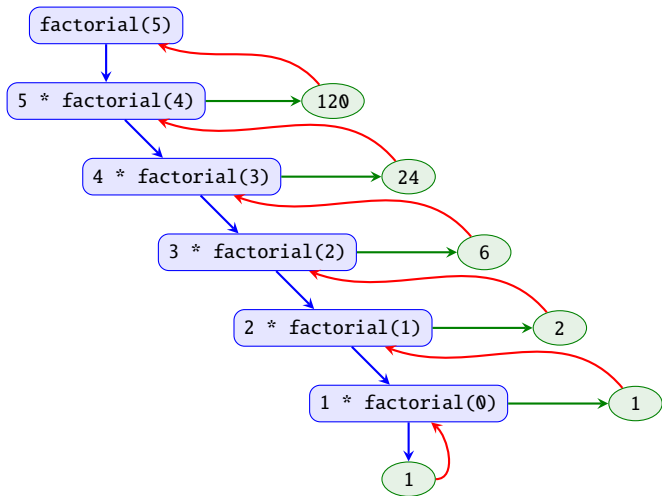
Faktoriaal



Faktoriaal



Faktoriaal



Sabarekursioon

- Kui kõik tegevused toimuvad enne rekursiivset väljakutset, nimetatakse rekursiooni **sabarekursiooniks**.
- Sabarekursiooni on väga lihtne teisendada tavaliseks tsüklikuks (ja ka vastupidi).

Sabarekursiivne faktoriaal

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

Faktoriaal

`factorial(5)`

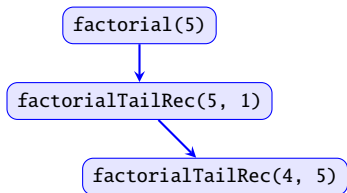
Faktoriaal

`factorial(5)`

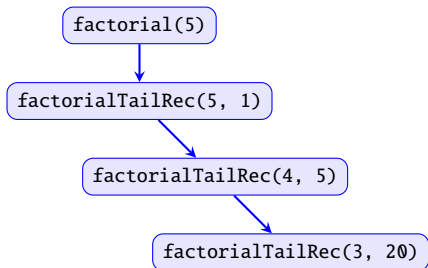


`factorialTailRec(5, 1)`

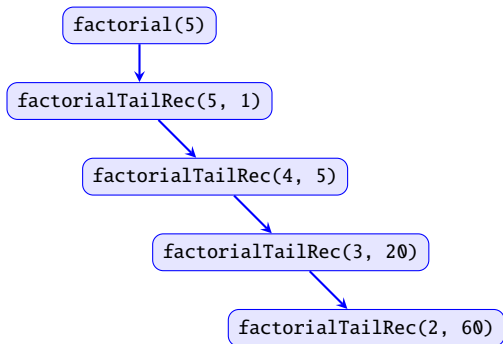
Faktoriaal



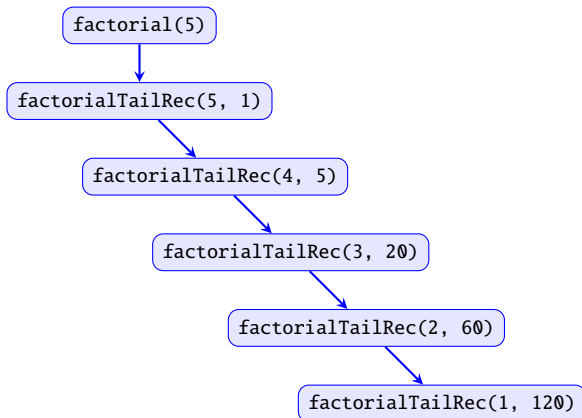
Faktoriaal



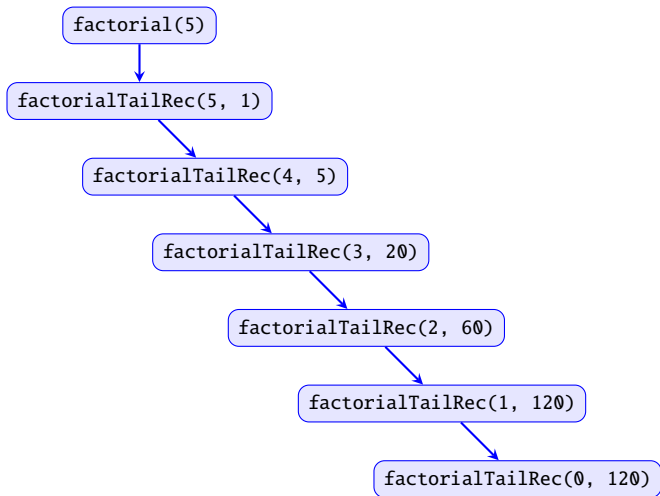
Faktoriaal



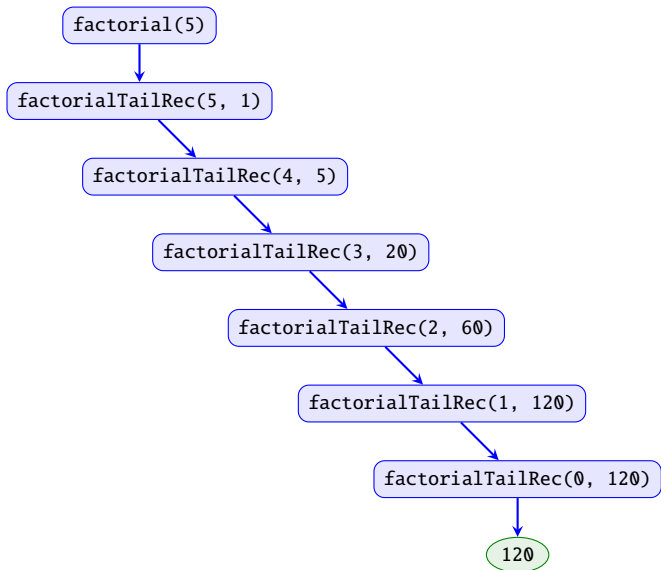
Faktoriaal



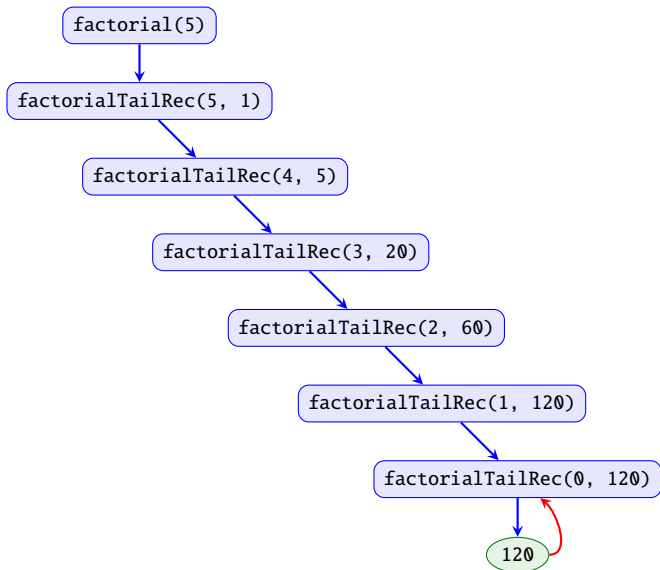
Faktoriaal



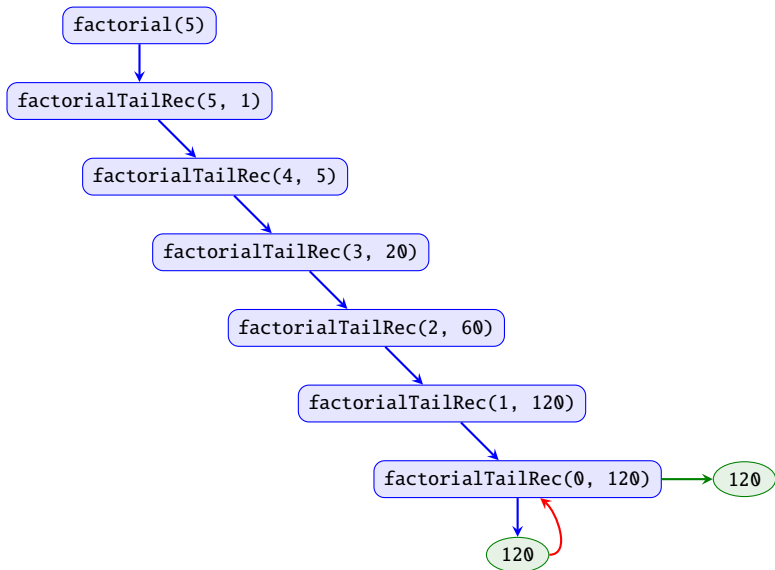
Faktoriaal



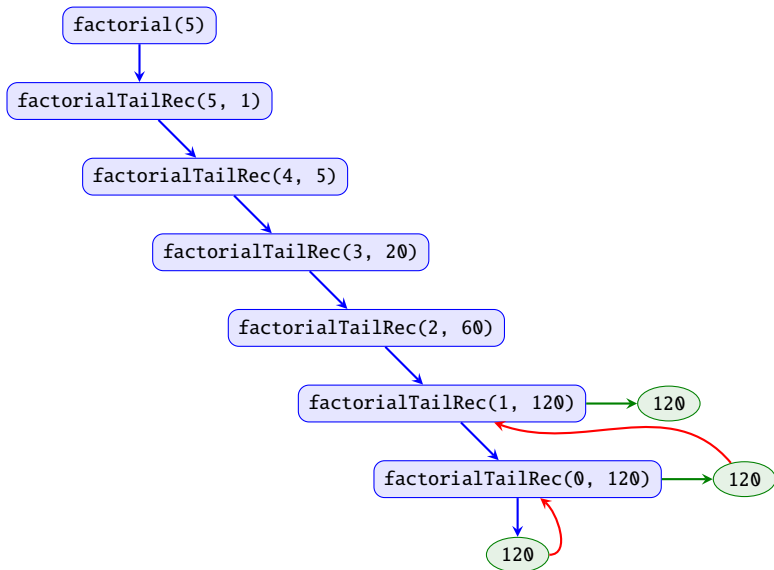
Faktoriaal



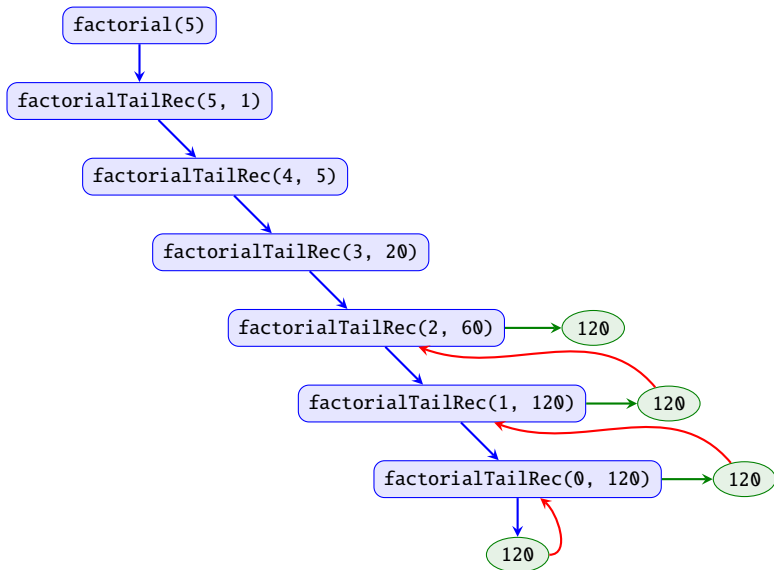
Faktoriaal



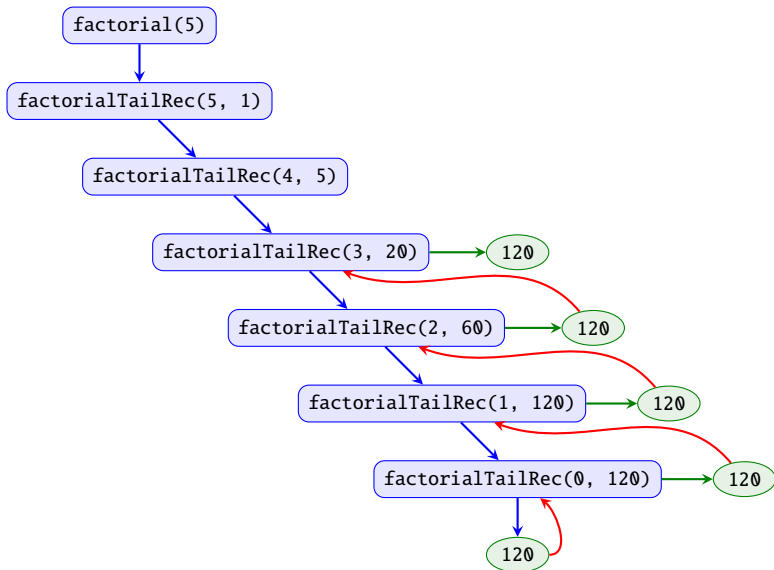
Faktoriaal



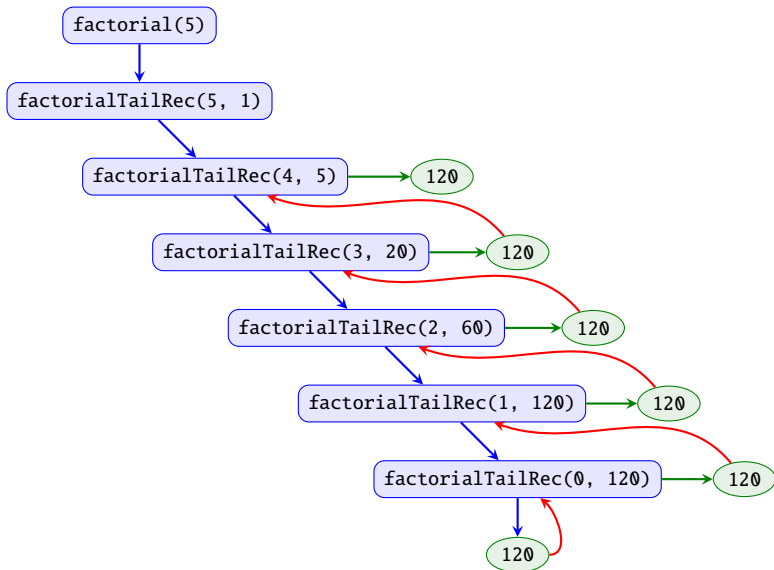
Faktoriaal



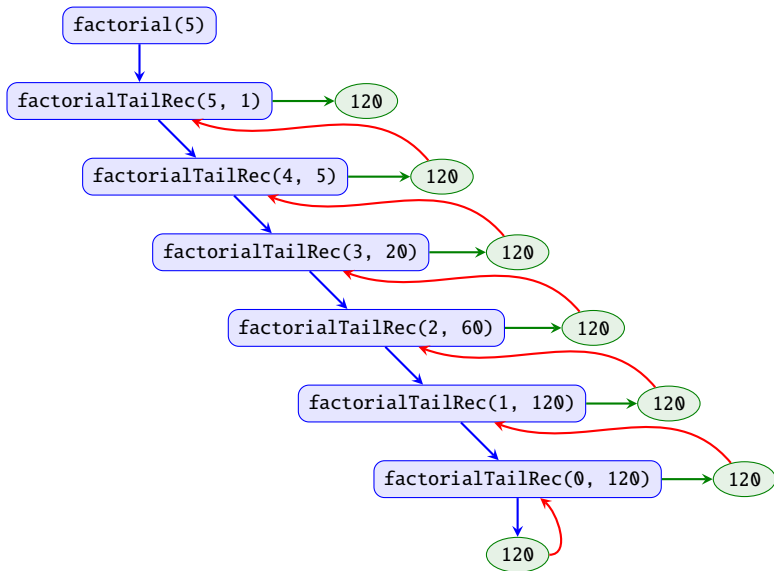
Faktoriaal



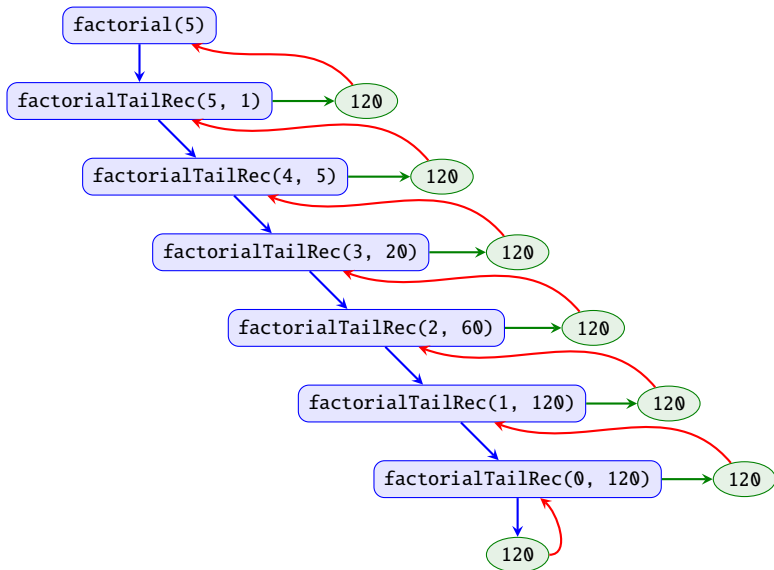
Faktoriaal



Faktoriaal



Faktoriaal



Sabarekursioon

- Kui kõik tegevused toimuvad enne rekursiivset väljakutset, nimetatakse rekursiooni **sabarekursiooniks**.
- Sabarekursiooni on väga lihtne teisendada tavaliseks tsüklikuks (ja ka vastupidi).

Sabarekursiivne faktoriaal

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


Sabarekursioon

- Kui kõik tegevused toimuvad enne rekursiivset väljakutset, nimetatakse rekursiooni **sabarekursiooniks**.
- Sabarekursiooni on väga lihtne teisendada tavaliseks tsüklikks (ja ka vastupidi).

Sabarekursiivne faktoriaal (ver. 2)

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

Sabarekursioon

- Kui kõik tegevused toimuvad enne rekursiivset väljakutset, nimetatakse rekursiooni **sabarekursiooniks**.
- Sabarekursiooni on väga lihtne teisendada tavaliseks tsüklikks (ja ka vastupidi).

Sabarekursiivne faktoriaal (ver. 3)

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

Sabarekursioon

- Kui kõik tegevused toimuvad enne rekursiivset väljakutset, nimetatakse rekursiooni **sabarekursiooniks**.
- Sabarekursiooni on väga lihtne teisendada tavaliseks tsüklikks (ja ka vastupidi).

Ekvivalentne iteratiivne faktoriaal

```
def factorial (n):  
    f = 1  
    while n != 0:  
        f = f*n  
        n = n-1  
    return f
```

Rekursiooniskeemid

- Kõigis seni vaadeldud näidetes kutsuti funktsiooni kehas funktsiooni ennast välja ülimalt üks kord.
- Sellist rekursiooni nimetatakse **lineaarseks**, kuna rekursiivsete väljakutsete kontrollvoog on lineaarne ahel.
- Keerukamad **rekursiooniskeemid** kasutavad erinevaid rekursiivse väljakutse mustreid:
 - **Puurekursiooni** korral kutsutakse funktsiooni rekursiivselt "samal tasemel" välja kaks või enam korda.
 - **Vastastikuse rekursiooni** korral kutsub üks funktsioon teist ja teine esimest.
 - ...

Puurekursioon

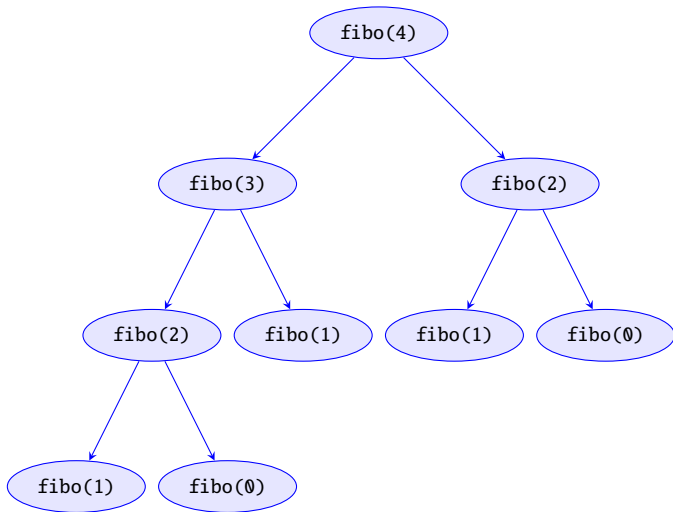
Fibonacci arvud

```
def fibonacci(n):  
    if n == 0:  
        return 0  
    elif n == 1:  
        return 1  
    else:  
        return fibonacci(n-1) + fibonacci(n-2)
```

Näide

```
print(fibonacci(2))      # Trükib: 1  
print(fibonacci(3))      # Trükib: 2  
print(fibonacci(4))      # Trükib: 3  
print(fibonacci(5))      # Trükib 5  
print(fibonacci(6))      # Trükib 8
```

Puurekursioon



Vastastikune rekursioon

Arvu paarsuse leidmine

```
def even(n):  
    if n == 0: return True  
    else:     return odd(n-1)  
  
def odd(n):  
    if n == 0: return False  
    else:     return even(n-1)
```

Näide

```
print(even(2))      # Trükitab: True  
print(even(3))     # Trükitab: False  
print(odd(2))      # Trükitab: False  
print(odd(3))     # Trükitab: True
```

Sisestatud rekursioon

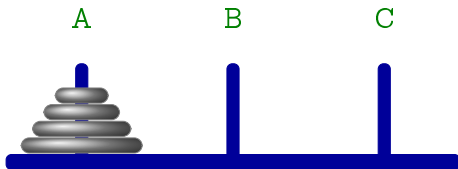
Ackermanni funktsioon

```
def ackermann(m, n):  
    if m == 0:  
        return n+1  
    elif n == 0:  
        return ackermann(m-1, 1)  
    else:  
        return ackermann(m-1, ackermann(m, n-1))
```

Näide

```
print(ackermann(3,4))    # Trükib: 125  
print(ackermann(3,5))    # Trükib: 253  
print(ackermann(3,6))    # Trükib: 509  
print(ackermann(3,7))    # Trükib: ...  
print(ackermann(4,1))    # Trükib: ...
```


Näide: Hanoi tornid



Algseis:

Tornis **A** on n erineva suurusega ketast.
Kettad on suuruse järjekorras (väiksem kõige peal).

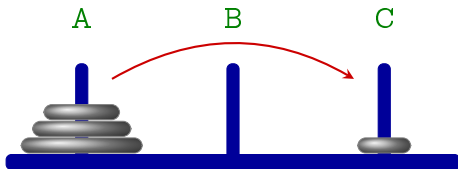
Eesmärk:

Viia kõik kettad tornist **A** torni **B**.

Käigud:

Mingi torni ülemise ketta võib tõsta teise torni.
Suuremat ketast ei tohi väiksema peale panna.

Näide: Hanoi tornid



Algseis:

Tornis **A** on n erineva suurusega ketast.
Kettad on suuruse järjekorras (väiksem kõige peal).

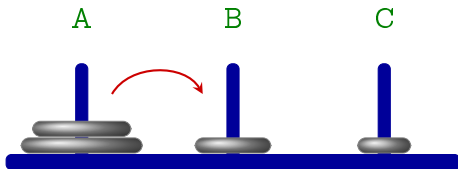
Eesmärk:

Viia kõik kettad tornist **A** torni **B**.

Käigud:

Mingi torni ülemise ketta võib tõsta teise torni.
Suuremat ketast ei tohi väiksema peale panna.

Näide: Hanoi tornid



Algseis:

Tornis **A** on n erineva suurusega ketast.

Kettad on suuruse järjekorras (väiksem kõige peal).

Eesmärk:

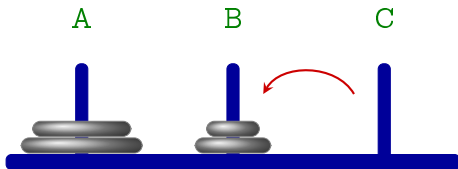
Viia kõik kettad tornist **A** torni **B**.

Käigud:

Mingi torni ülemise ketta võib tõsta teise torni.

Suuremat ketast ei tohi väiksema peale panna.

Näide: Hanoi tornid



Algseis:

Tornis **A** on n erineva suurusega ketast.

Kettad on suuruse järjekorras (väiksem kõige peal).

Eesmärk:

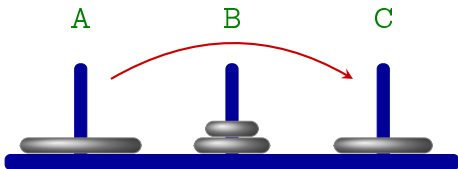
Viia kõik kettad tornist **A** torni **B**.

Käigud:

Mingi torni ülemise ketta võib tõsta teise torni.

Suuremat ketast ei tohi väiksema peale panna.

Näide: Hanoi tornid



Algseis:

Tornis **A** on n erineva suurusega ketast.

Kettad on suuruse järjekorras (väiksem kõige peal).

Eesmärk:

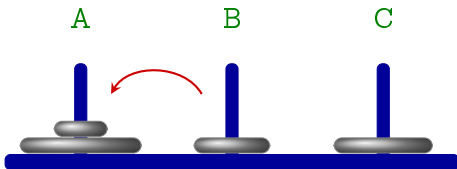
Viia kõik kettad tornist **A** torni **B**.

Käigud:

Mingi torni ülemise ketta võib tõsta teise torni.

Suuremat ketast ei tohi väiksema peale panna.

Näide: Hanoi tornid



Algseis:

Tornis **A** on n erineva suurusega ketast.
Kettad on suuruse järjekorras (väiksem kõige peal).

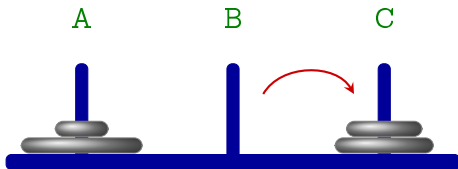
Eesmärk:

Viia kõik kettad tornist **A** torni **B**.

Käigud:

Mingi torni ülemise ketta võib tõsta teise torni.
Suuremat ketast ei tohi väiksema peale panna.

Näide: Hanoi tornid



Algseis:

Tornis **A** on n erineva suurusega ketast.
Kettad on suuruse järjekorras (väiksem kõige peal).

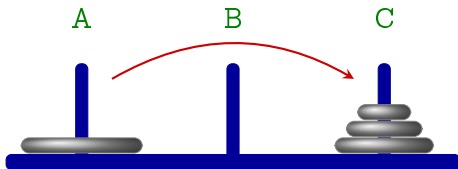
Eesmärk:

Viia kõik kettad tornist **A** torni **B**.

Käigud:

Mingi torni ülemise ketta võib tõsta teise torni.
Suuremat ketast ei tohi väiksema peale panna.

Näide: Hanoi tornid



Algseis:

Tornis **A** on n erineva suurusega ketast.
Kettad on suuruse järjekorras (väiksem kõige peal).

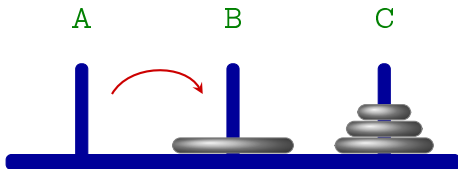
Eesmärk:

Viia kõik kettad tornist **A** torni **B**.

Käigud:

Mingi torni ülemise ketta võib tõsta teise torni.
Suuremat ketast ei tohi väiksema peale panna.

Näide: Hanoi tornid



Algseis:

Tornis **A** on n erineva suurusega ketast.
Kettad on suuruse järjekorras (väiksem kõige peal).

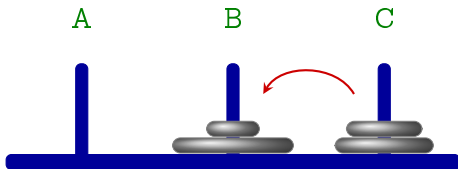
Eesmärk:

Viia kõik kettad tornist **A** torni **B**.

Käigud:

Mingi torni ülemise ketta võib tõsta teise torni.
Suuremat ketast ei tohi väiksema peale panna.

Näide: Hanoi tornid



Algseis:

Tornis **A** on n erineva suurusega ketast.
Kettad on suuruse järjekorras (väiksem kõige peal).

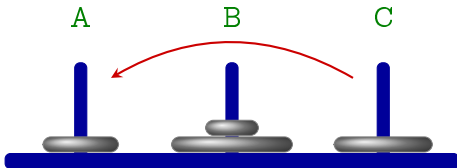
Eesmärk:

Viia kõik kettad tornist **A** torni **B**.

Käigud:

Mingi torni ülemise ketta võib tõsta teise torni.
Suuremat ketast ei tohi väiksema peale panna.

Näide: Hanoi tornid



Algseis:

Tornis **A** on n erineva suurusega ketast.
Kettad on suuruse järjekorras (väiksem kõige peal).

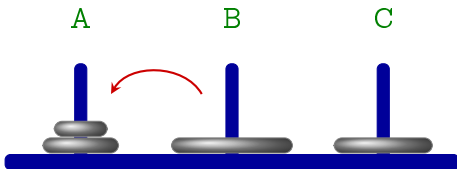
Eesmärk:

Viia kõik kettad tornist **A** torni **B**.

Käigud:

Mingi torni ülemise ketta võib tõsta teise torni.
Suuremat ketast ei tohi väiksema peale panna.

Näide: Hanoi tornid



Algseis:

Tornis **A** on n erineva suurusega ketast.
Kettad on suuruse järjekorras (väiksem kõige peal).

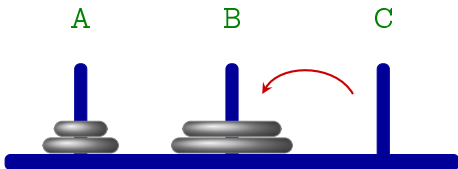
Eesmärk:

Viia kõik kettad tornist **A** torni **B**.

Käigud:

Mingi torni ülemise ketta võib tõsta teise torni.
Suuremat ketast ei tohi väiksema peale panna.

Näide: Hanoi tornid



Algseis:

Tornis **A** on n erineva suurusega ketast.
Kettad on suuruse järjekorras (väiksem kõige peal).

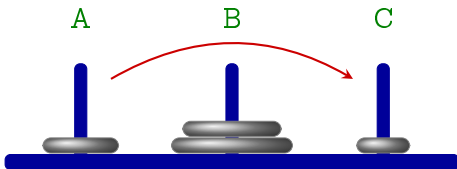
Eesmärk:

Viia kõik kettad tornist **A** torni **B**.

Käigud:

Mingi torni ülemise ketta võib tõsta teise torni.
Suuremat ketast ei tohi väiksema peale panna.

Näide: Hanoi tornid



Algseis:

Tornis **A** on n erineva suurusega ketast.
Kettad on suuruse järjekorras (väiksem kõige peal).

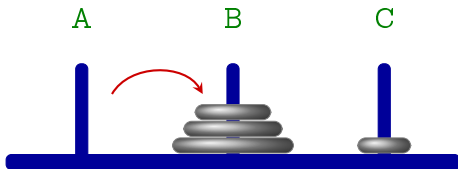
Eesmärk:

Viia kõik kettad tornist **A** torni **B**.

Käigud:

Mingi torni ülemise ketta võib tõsta teise torni.
Suuremat ketast ei tohi väiksema peale panna.

Näide: Hanoi tornid



Algseis:

Tornis **A** on n erineva suurusega ketast.
Kettad on suuruse järjekorras (väiksem kõige peal).

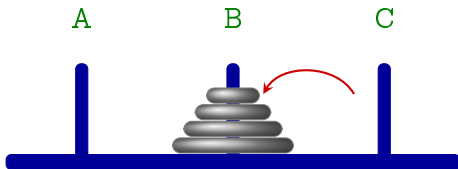
Eesmärk:

Viia kõik kettad tornist **A** torni **B**.

Käigud:

Mingi torni ülemise ketta võib tõsta teise torni.
Suuremat ketast ei tohi väiksema peale panna.

Näide: Hanoi tornid



Algseis:

Tornis **A** on n erineva suurusega ketast.
Kettad on suuruse järjekorras (väiksem kõige peal).

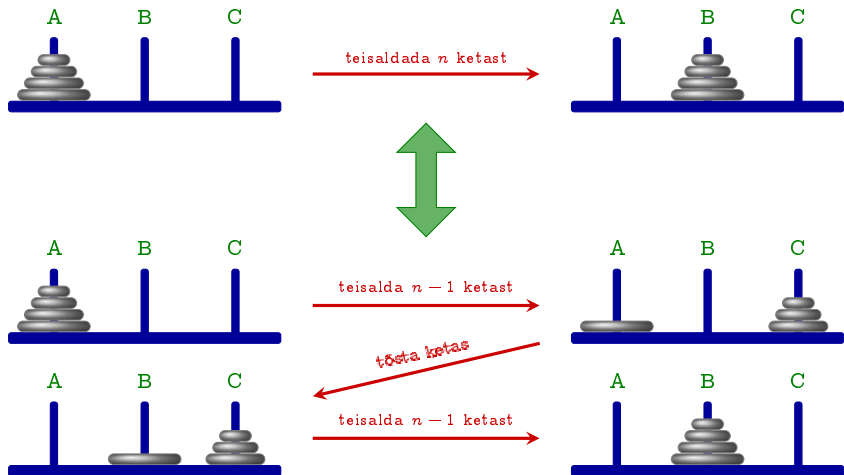
Eesmärk:

Viia kõik kettad tornist **A** torni **B**.

Käigud:

Mingi torni ülemise ketta võib tõsta teise torni.
Suuremat ketast ei tohi väiksema peale panna.

Näide: Hanoi tornid



Näide: Hanoi tornid

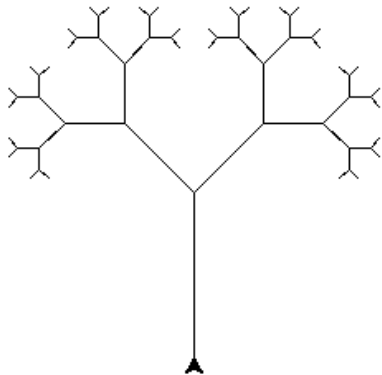
Lahendprogramm Pythonis

```
def moveTower(n,src,tgt,aux):
    if n == 1:
        moveDisk(src,tgt)
    else:
        moveTower(n-1,src,aux,tgt)
        moveDisk(src,tgt)
        moveTower(n-1,aux,tgt,src)

def moveDisk(src,tgt):
    print(src, "->", tgt)

n = int(input('Sisesta ketaste arv: '))
moveTower(n,'A','B','C')
```

Näide: puu joonistamine



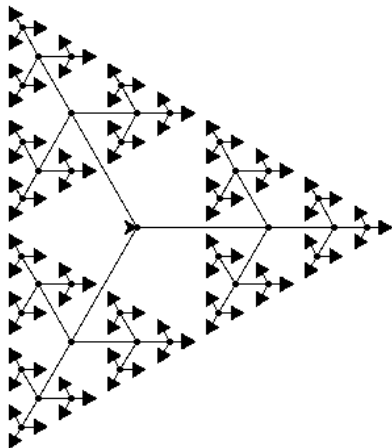
Näide: puu joonistamine

```
import turtle

def tree(length, level):
    if level <= 0:
        return
    turtle.forward(length)
    turtle.left(45)
    tree(0.6*length, level-1)
    turtle.right(90)
    tree(0.6*length, level-1)
    turtle.left(45)
    turtle.backward(length)
    return

turtle.left(90)
tree(100,6)
```

Näide: Sierpinski kolmnurk



Näide: Sierpinski kolmnurk

```
def sierpinski(length, level):  
    if level > 1:  
        turtle.dot()  
    if level == 0:  
        turtle.stamp()  
    else:  
        turtle.forward(length)  
        sierpinski(length/2, level-1)  
        turtle.backward(length)  
        turtle.left(120)  
        turtle.forward(length)  
        sierpinski(length/2, level-1)  
        turtle.backward(length)  
        turtle.left(120)  
        turtle.forward(length)  
        sierpinski(length/2, level-1)  
        turtle.backward(length)  
        turtle.left(120)
```

Suur tänu osalemast

ja

kohtumiseni!