Imperatív programozás

7. Előadás

Függvények, láthatóság (folytatás)

Modulok

Kivételkezelés

Beágyazott függvény

```
def lnko(x, y):
    def kivon(m, n):
        return m - n
    while not (x == y) :
        if x > y :
            x = kivon(x,y)
        else :
            y = kivon(y,x)
    return x

print(lnko(25, 100))
# print(kivon(120, 33))  --- itt nem elérhető!
```

Láthatóság

```
def lnko(x, y):
    def ciklusMag(x, y):
        if x > y :
            x = x - y
        else :
            y = y - x
        return (x, y)
    while not (x == y) :
        x,y = ciklusMag(x,y)
    return x

print(lnko(25, 100))
```

Láthatóság

```
def lnko():
    global x, y

    def ciklusMag():
        global x, y

        if x > y :
            x = x - y
        else :
            y = y - x

    while not (x == y) :
        ciklusMag()

    return x

x,y = 25, 100

print(lnko())
```

Láthatóság

```
def lnko(x, y):
    def ciklusMag():
        global x, y        #  ROSSZ MEGOLDÁS!!!!
        if x > y :
            x = x - y
        else :
            y = y - x
    while not (x == y) :
        ciklusMag()
    return x
x,y = 25, 100
print(lnko(x, y))
```

Láthatóság

```
def lnko(x, y):
    def ciklusMag():
        nonlocal x, y
        if x > y :
            x = x - y
        else :
            y = y - x
    while not (x == y) :
        ciklusMag()
    return x
x,y = 25, 100
print(lnko(x, y))
```

Modulok

fuggvenyek.py

```
def kivon(m, n):
    return m - n

def osszead(m,n):
    return m + n
```

program.py

```
import fuggvenyek

def lnko(x, y):
    while not (x == y) :
        if x > y :
            x = fuggvenyek.kivon(x,y)
        else :
            y = fuggvenyek.kivon(y,x)
    return x

print(lnko(25, 100))
```

Modulok

program.py

```
import pelda

# A másik modul kiíró művelete
# is végrehajtódik

def lnko(x, y):

    while not (x == y) :
        if x > y :
            x = pelda.kivon(x,y)
        else :
            y = pelda.kivon(y,x)

    return x


print(lnko(25, 100))
```

pelda.py

```
def kivon(m, n):
    return m - n

def osszead(m,n):
    return m + n

print(osszead(23,45))
```

Modulok

program.py

```
import pelda2

# Itt nem hajtódik végre
# a másik modul kiíró művelete

def lnko(x, y):
    while not (x == y) :
        if x > y :
            x = pelda2.kivon(x,y)
        else :
            y = pelda2.kivon(y,x)
    return x

print(lnko(25, 100))
```

pelda2.py

```
def kivon(m, n):
    return m - n

def osszead(m,n):
    return m + n

if __name__ == "__main__":
    print(osszead(23,45))
```

Modulok

pelda3.py

```
x = 20

def myFun():
    global x
    print(x)

def myFun2():
    global y
    print(y)
```

program.py

```
import pelda3

pelda3.myFun() # 20

pelda3.x = 100

pelda3.myFun() # 100

pelda3.y = 100

pelda3.myFun2() # 100
```

Modulok

program.py

```
import pelda3

z = 20

print(dir())
# ['__builtins__', '__cached__', '__doc__',
#  '__file__', '__loader__', '__name__',
#  '__package__', '__spec__', 'pelda3', 'z']

print(dir(pelda3))
# ['__builtins__', '__cached__', '__doc__',
#  '__file__', '__loader__', '__name__',
#  '__package__', '__spec__', 'myFun',
#  'myFun2', 'x']
```

pelda3.py

```
x = 20

def myFun():
    global x
    print(x)

def myFun2():
    global y
    print(y)
```

Modulok

program.py

fuggvenyek.py

```
def kivon(m, n):
    return m - n

def osszead(m,n):
    return m + n
```

```
from fuggvenyek import kivon

def lnko(x, y):
    while not (x == y) :
        if x > y :
            x = kivon(x,y)
        else :
            y = kivon(y,x)
    return x

print(lnko(25, 100))
```

Modulok

program.py

```
from fuggvenyek import kivon

# A másik modul kiíró művelete
# itt is végrehajtódik

def lnko(x, y):
    while not (x == y) :
        if x > y :
            x = kivon(x,y)
        else :
            y = kivon(y,x)
    return x

print(lnko(25, 100))
```

pelda.py

```
def kivon(m, n):
    return m - n

def osszead(m,n):
    return m + n

print(osszead(23,45))
```

Modulok

program.py

```
from fuggvenyek import *

def lnko(x, y):
    while not (x == y) :
        if x > y :
            x = kivon(x,y)
        else :
            y = kivon(y,x)
    return x

print(lnko(25, 100))
print(osszead(100,222))
```

fuggvenyek.py

```
def kivon(m, n):
    return m - n

def osszead(m,n):
    return m + n
```

Modulok

pelda3.py

```
x = 20

def myFun():
    global x
    print(x)

def myFun2():
    global y
    print(y)
```

program.py

```
from pelda3 import *

myFun() # 20

x = 100 # "lokális" változó lesz

myFun() # 20

y = 100 # "lokális" változó lesz

myFun2()
# NameError: name 'y' is not defined
```

Ciklusok kiegészítés

```
>>> x = 0
>>> while x < 10:
...     print(x)
...     if x > 3:
...         break
...     x = x + 1
...
0
1
2
3
4
```

```
>>> for x in range(12):
...     if x > 4 and x < 10:
...         continue
...     print(x)
...
0
1
2
3
4
10
11
```

Ciklusok kiegészítés

```
>>> x = 5
>>> while x < 6:
...     x = x + 1
... else:
...     print(x)
...
6
```

```
>>> x = 0
>>> while x < 10:
...     print(x)
...     if x > 2:
...         break
...     x = x + 1
... else:
...     print("Else ág!")
...
0
1
2
3
```

Ciklusok kiegészítés

„Hátultesztelő ciklus”

```
>>> x = 23
>>> while True:
...     x = x + 1
...     print(x)
...     if x > 6:
...         break
...
 24
```

Kivételkezelés

- **try**:

  belső blokk

  except kivétel$_1$: kivételkezelés$_1$

  except kivétel$_2$: kivételkezelés$_2$

  except: default kivételkezelés

  else: else ág

  finally: finally ág (mindenképpen végrehajtódik)

Kivételkezelés

Néhány beépített hiba típus:

- IndentationError - hibás indentálás (margó szabály)
- IndexError - túlindexelés
- IOError - I/O művelet hibája (pl. nem létező fájl)
- NameError - nem létező névre (pl. változó) hivatkozás
- SyntaxError - hibás szintaxis (pl. import-nál is)
- TypeError - hibás paramétertípus (pl. „hello” + 5)
- ValueError - nem megfelelő (típusú) érték
- ZeroDivisionError - nullával osztás

Kivételkezelés

```
import sys

try:
    y = int(sys.argv[1])
except ValueError: print("Nem egész szám a paraméter!")
except IndexError: print("Kell egy paraméter!")
except (NameError, TypeError) : print("Ide valószínűleg nem jutunk...")
except:
    print("Varatlan hiba!")
    raise
else:
    print("Ide jönne a program többi része.")
finally: print("Viszlat legközelebb!")
```

Kivételkezelés

- hiba kiváltása
    - raise NameError("Hiba leiras")
    - raise NameError("Hiba", 34, 562)
- hiba paraméterek kezelése
    - except NameError as errorObj: ...
        - az errorObj-ba kerül a hiba objektum
        - errorObj.args --- a kiváltáskor megadott paraméterek

Kivételkezelés

```
try:
    raise NameError("Hiba", 34, 562)
except NameError as e :   # e egy hiba objektum lesz
    print(e)              # __str__ függvény trükkje
    print(e.args)         # az objektum argumentumai
    x,y,z = e.args
    print(x, y, z)
```