UNIVERSITAS SCIENTIARUM SZEGEDIENSIS
SZEGEDI TUDOMÁNYEGYETEM
Department of Software Engineering

Szkriptnyelvek - 3. gyakorlat

Cservenák Bence

Függvények

- Pythonban is lehetőségünk van függvényeket írni az alábbi szintaxissal:

```
def fuggveny_neve([param1, param2, …]):
    utasitasok
    [return visszateresi_ertek]
```

  - a függvény törzsét beljebb kell indentálni
  - a függvényeknek lehet visszatérési értéke, ezt **`return`** után adjuk meg

- A megírt függvényeinket meghívhatjuk az alábbi szintaxissal:

```
fuggveny_neve([param1, param2, …])
```

Függvények

- Pythonban **nincs** function overload (még eltérő paraméterezés esetén sem!)
  - névütközés esetén a kódban a legkésőbb definiált függvény lesz hívható

```
def hello():
    print("Hello!")

def hello(nev):
    print("Hello " + nev + "!")

# hello()                    # hiba!
hello("Béla")
```

```
Hello Béla!
```

Függvények

- A függvény paramétereinek adhatók default értékek is
  - ha a függvényhívás során nem adjuk meg az adott paraméter értékét, akkor az a default értéket veszi fel
  - ha nem minden paraméternek adunk default értéket, akkor mindig az utolsó néhány paraméternek kell default értéket adnunk

```
def osszead(a, b = 0, c = 0):
    return a + b + c

print(osszead(5))                # 5
print(osszead(5, 2))             # 7
print(osszead(5, 2, 3))          # 10
```


Függvények

- Default paraméterek esetén lehetőségünk van nevesített paraméterátadásra
  - az opcionális paraméterek közül expliciten megmondjuk, hogy melyiknek az értékét akarjuk megadni (`név=érték` formában)

```
def user_info(felhasznalonev, jelszo=None, szak=None):
    if jelszo is not None:
        print(felhasznalonev, "jelszava:", jelszo)
    if szak is not None:
        print(felhasznalonev, "szakja:", szak)

user_info("cservZ", szak="proginfó")
```

```
cservZ szakja: proginfó
```

    - a példában szereplő `None` a más nyelvekből ismert `null` Pythonos megfelelője

Stringek

UNIVERSITAS SCIENTIARUM SZEGEDIENSIS
SZEGEDI TUDOMÁNYEGYETEM
Department of Software Engineering

- Stringek megadása: aposztrófok (`'…'`) vagy idézőjelek (`"…"`) között
- Összefűzés: `+` operátorral

```
s1 = "szkript"
s2 = "nyelvek"

szoveg = s1 + s2
print(szoveg)
```

```
szkriptnyelvek
```

- **`len`**`(s)`: visszaadja az `s` string hosszát

```
szoveg = "Python"
print(len(szoveg))
```

```
6
```

Stringek

- String karaktereinek indexelése: `string_neve[index]`
  - az indexelés minden esetben 0-tól kezdődik
  - negatív index esetén a string végétől kezd el számolni
  - az `index` megadható intervallumosan is: `mettől:meddig:lépésköz`
    - nem muszáj mindegyik értéket megadni, alapból a 0. karaktertől megy az utolsó karakterig, 1-es lépésközzel

```
szoveg = "macskajancsi"

print(szoveg[0])          # 0. karakter
print(szoveg[-1])         # utolsó karakter
print(szoveg[1:5])        # 1-5. karakter
print(szoveg[:])          # a string elejétől a végéig
print(szoveg[::2])        # minden második karakter
print(szoveg[::-1])       # string megfordítása
```

```
m
i
acsk
macskajancsi
mckjns
iscnajakscam
```

Stringek

- Pythonban minden string immutable!
  - a kezdeti értékük nem változhat meg

```
szoveg = "Jaj, erre figyelni kell!"
szoveg[1] = 'u'                    # ez nem fog működni!
```

```
Traceback (most recent call last):
  File "strings.py", line 2, in <module>
    szoveg[1] = 'u'
TypeError: 'str' object does not support item assignment
```


Stringek

- Rengeteg beépített stringkezelő függvény
  - `s.lower()`: csupa kisbetűssé alakítja az `s` stringet
  - `s.upper()`: csupa nagybetűssé alakítja az `s` stringet
  - `s.startswith(v)`: igazat ad vissza, ha az `s` string a `v` értékkel kezdődik
  - `s.endswith(v)`: igazat ad vissza, ha az `s` string a `v` értékre végződik
  - `s.strip()`: eltávolítja az `s` string elején és végén lévő whitespace karaktereket
  - `s.count(v)`: visszaadja, hogy hányszor szerepel az `s` stringben a `v` érték
  - `s.replace(old, new)`: lecseréli az `s` stringben az összes `old` részstringet `new`-ra
  - `s.split(delim)`: feldarabolja az `s` stringet, `delim` karakterek mentén (listát ad vissza)
  - `s.isalpha()`: igazat ad vissza, ha az `s` stringben csak betűk vannak
  - `s.isdigit()`: igazat ad vissza, ha az `s` stringben csak számjegyek vannak
  - …

Stringek

- Rengeteg beépített stringkezelő függvény

```
szoveg = "  A füstölt szalonna a legjobb szalonna Gábor szerint                  "
szoveg = szoveg.strip()  # whitespace eltávolítása a szöveg elejéről és a végéről

print(szoveg.lower())                  # kisbetűsítés
print(szoveg.upper())                  # nagybetűsítés
print(szoveg.endswith("szerint"))      # a "szerint" stringre végződik-e a szöveg
print(szoveg.count("a"))               # hány darab 'a' betű van a szövegben
print(szoveg.replace("szalonna", "kolbász"))    # lecserélés
print(szoveg.split(" "))               # feldarabolás szóközök mentén
```

```
a füstölt szalonna a legjobb szalonna gábor szerint
A FÜSTÖLT SZALONNA A LEGJOBB SZALONNA GÁBOR SZERINT
True
5
A füstölt kolbász a legjobb kolbász Gábor szerint
['A', 'füstölt', 'szalonna', 'a', 'legjobb', 'szalonna', 'Gábor', 'szerint']
```

Stringek

- Stringek karaktereinek bejárása:

```
szoveg = "Python"


for betu in szoveg:
    print(betu)
```

```
P
y
t
h
o
n
```


Stringek

- Kiegészítés: stringek egyéb megadási módjai

```
print("A pí értéke: {pi:.2f}".format(pi = 3.14159265359))    # format() metódus

nev = "Béla"
eletkor = 20
print("Hello, %s vagyok, %d éves." % (nev, eletkor))         # %-formatting

elet_ertelme = 42
print(f"Az élet értelme: {elet_ertelme}")                    # f-stringek
```

```
A pí értéke: 3.14
Hello, Béla vagyok, 20 éves.
Az élet értelme: 42
```

UNIVERSITAS SCIENTIARUM SZEGEDIENSIS
SZEGEDI TUDOMÁNYEGYETEM
Department of Software Engineering

- Pythonban a lista gyakorlatilag dinamikus méretű tömb
- Tetszőleges számú, nem feltétlenül azonos típusú elemek tárolására alkalmas
- Létrehozás (példa):

```
ures = list()              # üres listát hoz létre
kutyak = ["Ubul", "Snoopy", "Scooby Doo"]
```

- **len**(lista): visszaadja a lista hosszát

```
kutyak = ["Ubul", "Snoopy", "Scooby Doo"]
print(len(kutyak))
```

```
3
```

- Listaelemek indexelése: `lista_neve[index]`
  - az indexelés minden esetben 0-tól kezdődik
  - negatív index és intervallumos indexelés ugyanúgy, mint a stringeknél

```
gyumolcsok = ["alma", "citrom", "barack", "pomeló"]


print(gyumolcsok[0])     # 0. elem
print(gyumolcsok[-1])    # utolsó elem
print(gyumolcsok[1:3])   # 1-3. elem
print(gyumolcsok[::])    # a lista elejétől a végéig
print(gyumolcsok[::2])   # minden második elem
print(gyumolcsok[::-1]) # a lista megfordítása
```

```
alma
pomeló
['citrom', 'barack']
['alma', 'citrom', 'barack', 'pomeló']
['alma', 'barack']
['pomeló', 'barack', 'citrom', 'alma']
```

UNIVERSITAS SCIENTIARUM SZEGEDIENSIS
SZEGEDI TUDOMÁNYEGYETEM
Department of Software Engineering

- Rengeteg beépített függvény listák kezelésére
  - `lista.append(e)`: beszúrja az `e` elemet a `lista` végére
  - `lista.insert(i, e)`: beszúrja az `e` elemet a `lista i`. indexére
  - `lista.remove(e)`: törli a `lista`-ból a **legelső** `e` elemet
  - `lista.sort()`: rendezi a `lista` elemeit
  - `lista.clear()`: kiüríti a `lista`-t
  - …
- `e in lista`: igazat ad vissza, ha `e` szerepel a `lista`-ban
- `del lista[i]`: törli a `lista i`. indexén lévő elemet

- Rengeteg beépített függvény listák kezelésére

```
hallgatok = ["Józsi", "Béla", "Sanyi", "Béla"]
hallgatok[0] = "András"                  # 0. elem módosítása

hallgatok.append("Ervin")                # hozzáfűzés a lista végéhez
print(hallgatok)
hallgatok.insert(0, "Gábor")             # hozzáfűzés a lista elejéhez
print(hallgatok)

hallgatok.remove("Béla")                 # az első "Béla"-t törli

if "Sanyi" in hallgatok:                 # "Sanyi" benne van-e a listában
   print("Sanyi hallgató")

hallgatok.sort()                         # rendezés (itt: ábécé sorrend)
print(hallgatok)

hallgatok.clear()                        # lista kiürítése
print(hallgatok)
```

- Az előző dián szereplő kód outputja:

```
['András', 'Béla', 'Sanyi', 'Béla', 'Ervin']
['Gábor', 'András', 'Béla', 'Sanyi', 'Béla', 'Ervin']
Sanyi hallgató
['András', 'Béla', 'Ervin', 'Gábor', 'Sanyi']
[]
```

UNIVERSITAS SCIENTIARUM SZEGEDIENSIS
SZEGEDI TUDOMÁNYEGYETEM
Department of Software Engineering

- Lista elemeinek bejárása:

```
hallgatok = ["Józsi", "Béla", "Sanyi"]

for hallgato in hallgatok:
   print(hallgato)
```

```
Józsi
Béla
Sanyi
```

- A tuple egy rendezett elem n-es, tulajdonképpen egy immutable lista
  - létrehozás után nem módosítható
- Az `append` és `sort` metódusokon kívül minden ugyanúgy működik, mint a listáknál

```
lottoszamok = (42, 21, 8, 67, 15)                # tuple létrehozása

print(lottoszamok[1:4])                          # elemek indexelése (mint a listáknál)
# lottoszamok[1] = 12                            # hiba! (immutable)
print(42 in lottoszamok)                         # 42 benne van-e a tuple-ben
# lottoszamok.append(20)                         # hiba! (append és sort nem működik)
```

```
(21, 8, 67)
True
```

- A halmaz (set) olyan adatszerkezet, amelyben nem lehetnek duplikátumok, és az elemek között nincs sorrendiség

```
prognyelvek = set()                                 # üres halmaz létrehozása
prognyelvek.add("Python")                           # egy elem hozzáadása
prognyelvek.add("JavaScript")
prognyelvek.add("JavaScript")
prognyelvek.update(["Java", "C++", "PHP"])          # több elem együttes hozzáadása
print(prognyelvek)

prognyelvek.remove("PHP")                           # elem törlése
print(prognyelvek)
print("Matlab" in prognyelvek)                      # elem előfordulásának lekérése
```

```
{'Java', 'PHP', 'C++', 'Python', 'JavaScript'}
{'Java', 'C++', 'Python', 'JavaScript'}
False
```