Függvények, modulok

Miért kellenek függvények?

- Probléma: ismétlődő kód, átláthatatlanság.
- Megoldás: a kódot logikai egységekre bontjuk.
- A függvény csak akkor fut le, ha meghívjuk.

```
def fuggveny_nev():
    utasitas
```

Egyszerű függvény:

```
def koszones():
    print("Szia!")
def uzenet():
    print("K")
uzenet()
uzenet()
uzenet()
koszones()
```

Paraméterek Paraméter: adat, amit a függvény kap. Bármennyi lehet, akár 0 is.

```
def koszones(nev):
    print("Szia", nev)

koszones("Anna")

def osszeg(a, b):
    print(a + b)
osszeg(6,7)
```

Példa: Írj függvényt, ami kiszámolja egy téglalap területét. Add át paraméterként a szélességet és magasságot.

```
def teglalap_terulet(szelesseg, magassag):
    print(szelesseg * magassag)

teglalap_terulet(5, 3)
```

return (visszatérési érték) A return értéket ad vissza.

```
def osszeg(a, b):
    return a + b

x = osszeg(4, 6)
print(x)
```

Példa: Írj függvényt, ami visszaadja egy szám négyzetét. Használd az eredményt egy változóban.

```
def negyzet(szam):
    return szam ** 2

eredmeny = negyzet(5)
print("Négyzet:", eredmeny)
```

Összefoglaló feladatok

1. feladat: Írj függvényt, ami:

- kap egy listát paraméterként
- visszaadja az elemek összegét

```
def lista_osszeg(lista):
    osszeg = 0
    for elem in lista:
        osszeg += elem
    return osszeg

szamok = [1, 2, 3, 4]
print(lista_osszeg(szamok))
```

2. feladat: Írj programot, ami:

1. Bekér 5 számot
2. Listába teszi
3. Függvénnyel kiszámolja:
    - az összegét
    - az átlagát
4. Kiírja az eredményeket

```
def osszeg(lista):
    s = 0
    for elem in lista:
        s += elem
    return s


def atlag(lista):
    return osszeg(lista) / len(lista)

szamok = []
for i in range(5):
    szam = float(input("Adj meg egy számot: "))
    szamok.append(szam)

print("Összeg:", osszeg(szamok))
print("Átlag:", atlag(szamok))
```

3. feladat: Írj egy függvényt osszeg névvel, ami egy lista számot kap és visszaadja az összegüket. Próbáld ki a függvényt egy felhasználótól bekért számok listájával.

```
# Megoldás
def osszeg(szamok):
    total = 0
    for sz in szamok:
        total += sz
    return total

# Példa input
be = input("Adj meg számokat szóközzel elválasztva: ")
szamok_lista = [int(x) for x in be.split()]

print("Összeg:", osszeg(szamok_lista))
```

4. feladat: Írj egy függvényt legnagyobb_szo(szavak) névvel, ami egy szavak listáját kapja és visszaadja a leghosszabb szót.

```
# Megoldás
def legnagyobb_szo(szavak):
    max_szo = ""
    for szo in szavak:
        if len(szo) > len(max_szo):
            max_szo = szo
    return max_szo

szavak = ["alma", "korte", "barack", "sargadinnye"]
print("Leghosszabb szó:", legnagyobb_szo(szavak))
```

5. feladat: Készíts egy függvényt szavak_szamlalo(szoveg), ami visszaad egy dict-et: kulcs a szó, érték a darabszám.

```
# Megoldás
def szavak_szamlalo(szoveg):
    szavak = szoveg.split()
    szamolas = {}
    for szo in szavak:
        szamolas[szo] = szamolas.get(szo, 0) + 1
    return szamolas

szoveg = "alma korte alma barack alma"
print(szavak_szamlalo(szoveg))
```

6. feladat: Írj egy függvényt paros_paratlan(szamok) névvel, ami egy lista számot kap és két listát ad vissza: párosok és páratlanok.

```
# Megoldás
def paros_paratlan(szamok):
    paros = []
    paratlan = []
    for sz in szamok:
        if sz % 2 == 0:
            paros.append(sz)
        else:
            paratlan.append(sz)
    return paros, paratlan

szamok = [1,2,3,4,5,6,7,8]
paros, paratlan = paros_paratlan(szamok)
print("Párosok:", paros)
print("Páratlanok:", paratlan)
```

7.feladat: Írj egy függvényt egyedi_karakterek(szoveg), ami egy stringet kap és visszaad egy set-et, ami az összes egyedi karaktert tartalmazza.

```
# Megoldás
def egyedi_karakterek(szoveg):
    karakterek = set()
    for c in szoveg:
        karakterek.add(c)
    return karakterek

szoveg = "programozas"
print("Egyedi karakterek:", egyedi_karakterek(szoveg))
```

8. feladat: Képzeld el, hogy egy kis osztályban szeretnénk nyilvántartani a diákok nevét, tantárgyait és jegyeit. - Készíts egy programot, ami:

1. Bekéri a diákok nevét és tantárgyait.
2. Minden diákhoz több jegyet lehet felvinni listában.
3. Függvényekkel:
    - Egy függvény számolja ki a diákok átlagát tantárgyanként.
    - Egy függvény visszaadja a legjobb jegyet minden diákhoz.
    - Egy függvény megmondja, melyik diákoknak van 1-es jegye.
4. Külön set segítségével listázd az összes tantárgyat, amit az osztályban tanulnak.
5. Külön dict segítségével listázd, hogy hány diák tanul minden tantárgyat.

```
# Függvények

def atlag_diak(jegyek):
    """Lista jegyek -> átlag"""
    if len(jegyek) == 0:
        return 0
    osszeg = 0
    for szam in jegyek:
        osszeg += szam
    return osszeg / len(jegyek)

def legjobb_jegy(jegyek):
    """Lista jegyek -> legnagyobb érték"""
    if len(jegyek) == 0:
        return None
    max_jegy = jegyek[0]
    for szam in jegyek:
        if szam > max_jegy:
            max_jegy = szam
    return max_jegy

def egyes_jegy(jegyek):
    """Ellenőrzi, van-e 1-es jegy"""
    for szam in jegyek:
        if szam == 1:
            return True
    return False

# Adatok felvitele
diakok = {}

while True:
    nev = input("Add meg a diák nevét (vagy ENTER a befejezéshez): ")
    if nev == "":
        break

    tantargyak = input("Add meg a tantárgyakat szóközzel elválasztva:
```

```
").split()
    jegyek = []

    for tantargy in tantargyak:
        be = input(f"Add meg {nev} jegyeit a(z) {tantargy}-ból szóközzel
elválasztva: ")
        jegyek_tantargy = [int(x) for x in be.split()]
        jegyek.append((tantargy, jegyek_tantargy))

    diakok[nev] = jegyek

# Összes tantárgy set
osszes_tantargy = set()
for jegyek_lista in diakok.values():
    for tantargy, _ in jegyek_lista:
        osszes_tantargy.add(tantargy)

print("\nÖsszes tantárgy az osztályban:", osszes_tantargy)

# Tantárgyhoz hány diák
tantargy_darab = {}
for tantargy in osszes_tantargy:
    count = 0
    for jegyek_lista in diakok.values():
        for t, _ in jegyek_lista:
            if t == tantargy:
                count += 1
                break
    tantargy_darab[tantargy] = count

print("\nDiákok száma tantárgyanként:", tantargy_darab)

# Diákok átlagai, legjobb jegyek, 1-es jegyek
print("\nDiák statisztikák:")
for nev, jegyek_lista in diakok.items():
    print(f"\n{nev}:")
    for tantargy, jegyek_tantargy in jegyek_lista:
        print(f"  {tantargy}: átlag={atlag_diak(jegyek_tantargy):.2f},
legjobb={legjobb_jegy(jegyek_tantargy)}")
        if egyes_jegy(jegyek_tantargy):
            print("    Figyelem: van 1-es jegy!")
```

Osztályok

- Osztály (class): Sablon/tervrajz objektumokhoz.
- Objektum (instance): Konkrét példány az osztályból.

- Attribútum: Objektum tulajdonsága (self.nev, self.ev).
- Metódus: Objektumhoz tartozó függvény (def metodus(self): ...).
- Konstruktor: init – automatikusan meghívódik objektum létrehozásakor.
- Öröklődés: Egy osztály átveszi más osztály tulajdonságait és metódusait.
- Felülírás: Gyerek osztály felülírja a szülő metódusát.

Egyszerű példa:

```
class Auto:
    def __init__(self, marka, ev):
        self.marka = marka
        self.ev = ev

    def bemutatkozik(self):
        print(f"Ez egy {self.marka} típusú autó, {self.ev}-es évjárat.")

auto1 = Auto("Toyota", 2020)
auto1.bemutatkozik()
```

Öröklés:

```
class Jarmu:
    def __init__(self, marka):
        self.marka = marka

    def bemutatkozik(self):
        print(f"Jármű: {self.marka}")

class Auto(Jarmu):
    def __init__(self, marka, ev):
        super().__init__(marka)
        self.ev = ev

    def bemutatkozik(self):
        print(f"Autó: {self.marka}, évjárat: {self.ev}")

auto2 = Auto("Audi", 2021)
auto2.bemutatkozik()
```

1. feladat: Diák osztály

- Attribútumok: nev, eletkor, jegyek (lista)
- Metódusok: jegyet_hozzaad, atlag, legjobb_jegy

```
class Diak:
    def __init__(self, nev, eletkor):
        self.nev = nev
```

```
        self.eletkor = eletkor
        self.jegyek = []

    def jegyet_hozzaad(self, jegy):
        self.jegyek.append(jegy)

    def atlag(self):
        if len(self.jegyek) == 0:
            return 0
        return sum(self.jegyek)/len(self.jegyek)

    def legjobb_jegy(self):
        if len(self.jegyek) == 0:
            return None
        return max(self.jegyek)

# Példa
diak = Diak("Anna", 16)
diak.jegyet_hozzaad(5)
diak.jegyet_hozzaad(4)
diak.jegyet_hozzaad(3)

print(diak.nev, "átlaga:", diak.atlag())
print(diak.nev, "legjobb jegye:", diak.legjobb_jegy())
```

2. feladat: Banki számla

- Attribútumok: tulajdonos, egyenleg
- Metódusok: betesz, kivesz, egyenleg_lekerdez

```
class BankSzamla:
    def __init__(self, tulajdonos, egyenleg=0):
        self.tulajdonos = tulajdonos
        self.egyenleg = egyenleg

    def betesz(self, osszeg):
        self.egyenleg += osszeg

    def kivesz(self, osszeg):
        if osszeg <= self.egyenleg:
            self.egyenleg -= osszeg
        else:
            print("Nincs elég pénz!")

    def egyenleg_lekerdez(self):
        return self.egyenleg
```

```
# Példa
szamla = BankSzamla("Bence", 1000)
szamla.betesz(500)
szamla.kivesz(200)
print(szamla.tulajdonos, "egyenlege:", szamla.egyenleg_lekerdez())
```

3. feladat: Állat öröklődés:

- Szülő osztály: Allat (attribútum: nev)
- Gyerek osztályok: Kutya, Macska (metódus: hangot_ad)

```
class Allat:
    def __init__(self, nev):
        self.nev = nev

class Kutya(Allat):
    def hangot_ad(self):
        print(f"{self.nev} ugat: Vau-vau!")

class Macska(Allat):
    def hangot_ad(self):
        print(f"{self.nev} nyávog: Miau!")

kutya = Kutya("Bodri")
macska = Macska("Cirmi")
kutya.hangot_ad()
macska.hangot_ad()
```

4. feladat: Könyv osztály

- Attribútumok: cim, szerzo, oldalszam
- Metódus: bemutatkozik

```
class Konyv:
    def __init__(self, cim, szerzo, oldalszam):
        self.cim = cim
        self.szerzo = szerzo
        self.oldalszam = oldalszam

    def bemutatkozik(self):
        print(f"{self.cim} ({self.szerzo}), {self.oldalszam} oldal")

konyv = Konyv("Python alapok", "Kovács Péter", 250)
konyv.bemutatkozik()
```

5. feladat: Jármű osztály + Auto és Motor

- Szülő: Jarmu (típus, ev)
- Gyerek: Auto, Motor
- Metódus: bemutatkozik

```
class Jarmu:
    def __init__(self, tipus, ev):
        self.tipus = tipus
        self.ev = ev

class Auto(Jarmu):
    def bemutatkozik(self):
        print(f"Autó: {self.tipus}, évjárat: {self.ev}")

class Motor(Jarmu):
    def bemutatkozik(self):
        print(f"Motor: {self.tipus}, évjárat: {self.ev}")

auto = Auto("BMW", 2022)
motor = Motor("Yamaha", 2021)
auto.bemutatkozik()
motor.bemutatkozik()
```

From:
https://edu.iit.uni-miskolc.hu/ - **Institute of Information Science - University of Miskolc**

Permanent link:
https://edu.iit.uni-miskolc.hu/tanszek:oktatas:muszaki_informatika:fueggvenyek_modulok

Last update: **2026/02/19 20:28**