

# Python Programozási Útmutató

## Bevezetés

A Python egy magas szintű, általános célú programozási nyelv, amelyet Guido van Rossum fejlesztett ki 1991-ben. Jellemzője az egyszerű, olvasható szintaxis és a sokoldalúság.

## Python filozófia - Zen of Python

Szép jobb, mint csúnya.  
Explicit jobb, mint implicit.  
Egyszerű jobb, mint összetett.  
Összetett jobb, mint bonyolult.  
Az olvashatóság számít.

## Alapok

### Változók és típusok

```
# Számok
szam = 42
lebego = 3.14
komplex = 3 + 4j

# Szöveg
nev = "Python"
szoveg = 'Hello World'
tobbsoros = """Ez egy
többsoros
szöveg"""

# Boolean
igaz = True
hamis = False

# None (üres érték)
ures = None
```

### Típus konverziók

```
szam_str = str(42)          # "42"
szoveg_int = int("42")       # 42
szoveg_float = float("3.14") # 3.14
lista_str = str([1, 2, 3])   # "[1, 2, 3]"
```

## Operátorok

### Aritmetikai:

```
osszead = 5 + 3      # 8
kivon = 5 - 3       # 2
szoroz = 5 * 3       # 15
oszt = 5 / 3        # 1.666...
egesz_oszt = 5 // 3 # 1
maradek = 5 % 3      # 2
hatvany = 5 ** 3     # 125
```

### Összehasonlító:

```
egyenlo = (5 == 5)      # True
nem_egyenlo = (5 != 3)   # False
nagyobb = (5 > 3)       # True
kisebb = (5 < 3)        # False
nagyobb_egyenlo = (5 >= 5) # True
```

### Logikai:

```
es = True and False    # False
vagy = True or False   # True
nem = not True         # False
```

## Adatszerkezetek

### Listák (list)

Rendezett, módosítható gyűjtemény.

```
# Létrehozás
gyumolcsok = ["alma", "banán", "cseresznye"]
szamok = [1, 2, 3, 4, 5]
vegyes = [1, "két", 3.0, True]

# Hozzáférés
elso = gyumolcsok[0]          # "alma"
utolso = gyumolcsok[-1]        # "cseresznye"
reszlet = szamok[1:4]          # [2, 3, 4]

# Módosítás
gyumolcsok.append("dinnye")    # Hozzáad a végére
gyumolcsok.insert(1, "barack")  # Beszűr adott pozícióra
gyumolcsok.remove("alma")       # Eltávolít érték alapján
torolt = gyumolcsok.pop()       # Eltávolítja az utolsót
gyumolcsok.sort()              # Rendez
gyumolcsok.reverse()            # Megfordít
```

```
# List comprehension
negyzetek = [x**2 for x in range(10)]
parosak = [x for x in range(10) if x % 2 == 0]
```

### Tuple (tuple)

Rendezett, NEM módosítható gyűjtemény.

```
koordinatak = (10, 20)
szinek = ("piros", "kék", "zöld")

# Tuple unpacking
x, y = koordinatak
print(x) # 10
```

### Halmazok (set)

Rendezetlen, egyedi elemek gyűjteménye.

```
szamok = {1, 2, 3, 4, 5}
gyumolcsok = {"alma", "banán", "cseresznye"}

# Műveletek
szamok.add(6)           # Hozzáad
szamok.remove(1)         # Eltávolít
szamok.discard(10)      # Eltávolít (nincs hiba, ha nem létezik)

# Halmazműveletek
a = {1, 2, 3}
b = {3, 4, 5}

unio = a | b            # {1, 2, 3, 4, 5}
metszet = a & b          # {3}
kulonbseg = a - b        # {1, 2}
szimmetrikus = a ^ b      # {1, 2, 4, 5}
```

### Szótárak (dictionary)

Kulcs-érték párok gyűjteménye.

```
# Létrehozás
szemely = {
    "nev": "János",
    "kor": 30,
    "varos": "Budapest"
}

# Hozzáférés
```

```

nev = szemely["nev"]           # "János"
kor = szemely.get("kor", 0)     # 30 (vagy 0, ha nem létezik)

# Módosítás
szemely["kor"] = 31
szemely["email"] = "janos@example.com"
del szemely["varos"]

# Iteráció
for kulcs in szemely:
    print(kulcs, szemely[kulcs])

for kulcs, ertek in szemely.items():
    print(f"{kulcs}: {ertek}")

# Dictionary comprehension
negyzetek = {x: x**2 for x in range(5)}
# {0: 0, 1: 1, 2: 4, 3: 9, 4: 16}

```

## Vezérlési szerkezetek

### If-elif-else

```

kor = 18

if kor < 13:
    print("Gyerek")
elif kor < 18:
    print("Tizenéves")
elif kor < 65:
    print("Felnőtt")
else:
    print("Idős")

# Ternary operator
allapot = "Felnőtt" if kor >= 18 else "Kiskorú"

```

## Ciklusok

### For ciklus:

```

# Lista bejárása
gyumolcsok = ["alma", "banán", "cseresznye"]
for gyumolcs in gyumolcsok:
    print(gyumolcs)

# Range használata

```

```

for i in range(5):           # 0-tól 4-ig
    print(i)

for i in range(2, 10, 2): # 2-től 10-ig, 2-esével
    print(i)

# Enumerate (index + érték)
for index, gyumolcs in enumerate(gyumolcsok):
    print(f"{index}: {gyumolcs}")

# Zip (párhuzamos bezárás)
nevek = ["Anna", "Béla", "Cecil"]
korok = [25, 30, 35]
for nev, kor in zip(nevek, korok):
    print(f"{nev} {kor} éves")

```

While ciklus:

```

szamlalo = 0
while szamlalo < 5:
    print(szamlalo)
    szamlalo += 1

# Break és continue
for i in range(10):
    if i == 3:
        continue # Ugrik a következő iterációra
    if i == 7:
        break   # Kilép a ciklusból
    print(i)

```

## Függvények

### Alapok

```

# Egyszerű függvény
def udvozles():
    print("Hello, Világ!")

# Paraméterekkel
def udvozol(nev):
    print(f"Hello, {nev}!")

# Visszatérési értékkal
def osszeadas(a, b):
    return a + b

```

```

# Alapértelmezett paraméterek
def hatvany(szam, kitevo=2):
    return szam ** kitevo

# Több visszatérési érték
def osztas_maradekkal(a, b):
    return a // b, a % b

eredmeny, maradek = osztas_maradekkal(17, 5)

```

### Speciális paraméterek

```

# *args - változó számú pozicionális paraméter
def osszeg(*szamok):
    return sum(szamok)

print(osszeg(1, 2, 3, 4, 5)) # 15

# **kwargs - változó számú kulcsszavas paraméter
def adatok(**kwargs):
    for kulcs, ertek in kwargs.items():
        print(f"{kulcs}: {ertek}")

adatok(nev="János", kor=30, varos="Budapest")

```

### Lambda függvények

```

# Névtelen, egyszerű függvények
negyzet = lambda x: x ** 2
osszead = lambda a, b: a + b

# Gyakran használják beépített függvényekkel
szamok = [1, 2, 3, 4, 5]
negyzetek = list(map(lambda x: x**2, szamok))
parosak = list(filter(lambda x: x % 2 == 0, szamok))

```

### Dekorátorok

```

def ido_meres(func):
    import time
    def wrapper(*args, **kwargs):
        start = time.time()
        eredmeny = func(*args, **kwargs)
        end = time.time()
        print(f"Futási idő: {end - start:.4f} másodperc")
        return eredmeny
    return wrapper

```

```

@ido_meres
def lassú_fuggveny():
    import time
    time.sleep(1)
    return "Kész!"

lassú_fuggveny()

```

## Objektumorientált programozás

### Osztályok

```

class Auto:
    # Osztály változó ( minden példány közös )
    kerekkek_szama = 4

    # Konstruktor
    def __init__(self, marka, modell, ev):
        # Példány változók
        self.marka = marka
        self.modell = modell
        self.ev = ev
        self.km_ora = 0

    # Metódus
    def vezet(self, tavolsag):
        self.km_ora += tavolsag
        print(f"{tavolsag} km-t vezetett. Összes: {self.km_ora} km")

    # String reprezentáció
    def __str__(self):
        return f"{self.ev} {self.marka} {self.modell}"

    # Osztály metódus
    @classmethod
    def modell_ev_alapjan(cls, evjarat):
        return cls("Ismeretlen", "Modell", evjarat)

    # Statikus metódus
    @staticmethod
    def kerekkek():
        return 4

    # Használat
autom = Auto("Toyota", "Corolla", 2020)

```

```
print(autom)           # 2020 Toyota Corolla
autom.vezet(100)       # 100 km-t vezetett
```

## Öröklődés

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

    def inditas(self):
        print("A jármű elindul")

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

    def inditas(self): # Override
        print("Az autó motorja elindul")
        super().inditas()

class Motor(Jarmu):
    def __init__(self, marka, modell, tipus):
        super().__init__(marka, modell)
        self.tipus = tipus

# Használat
auto = Auto("Ford", "Focus", 5)
auto.inditas()
```

## Private változók és property

```
class Szamla:
    def __init__(self, egyenleg=0):
        self.__egyenleg = egyenleg # Private változó

    @property
    def egyenleg(self):
        return self.__egyenleg

    @egyenleg.setter
    def egyenleg(self, ertek):
        if ertek < 0:
            raise ValueError("Az egyenleg nem lehet negatív!")
        self.__egyenleg = ertek
```

```

def befizetés(self, osszeg):
    if osszeg > 0:
        self.__egyenleg += osszeg

def kivétel(self, osszeg):
    if osszeg > self.__egyenleg:
        raise ValueError("Nincs fedezet!")
    self.__egyenleg -= osszeg

# Használat
szamla = Szamla(1000)
print(szamla.egyenleg) # 1000
szamla.befizetés(500) # 1500

```

## Fájlkezelés

### Olvasás és írás

```

# Írás
with open("fajl.txt", "w", encoding="utf-8") as f:
    f.write("Hello, Világ!\n")
    f.write("Python programozás\n")

# Olvasás
with open("fajl.txt", "r", encoding="utf-8") as f:
    tartalom = f.read()
    print(tartalom)

# Soronkénti olvasás
with open("fajl.txt", "r", encoding="utf-8") as f:
    for sor in f:
        print(sor.strip())

# Hozzáfűzés
with open("fajl.txt", "a", encoding="utf-8") as f:
    f.write("Új sor\n")

```

### JSON kezelés

```

import json

# Adatok
adatok = {
    "nev": "János",
    "kor": 30,
    "kedvenc_szamok": [7, 13, 42]
}

```

```
}
```

```
# JSON fájlba írás
with open("adatok.json", "w", encoding="utf-8") as f:
    json.dump(adatok, f, indent=4, ensure_ascii=False)
```

```
# JSON fájl olvasása
with open("adatok.json", "r", encoding="utf-8") as f:
    betoltott = json.load(f)
    print(betoltott["nev"])
```

## CSV kezelés

```
import csv

# CSV írás
with open("adatok.csv", "w", newline="", encoding="utf-8") as f:
    writer = csv.writer(f)
    writer.writerow(["Név", "Kor", "Város"])
    writer.writerow(["János", 30, "Budapest"])
    writer.writerow(["Anna", 25, "Debrecen"])

# CSV olvasás
with open("adatok.csv", "r", encoding="utf-8") as f:
    reader = csv.reader(f)
    for sor in reader:
        print(sor)

# DictReader/DictWriter
with open("adatok.csv", "r", encoding="utf-8") as f:
    reader = csv.DictReader(f)
    for sor in reader:
        print(f"{sor['Név']} - {sor['Kor']} éves")
```

## Hibakezelés

```
# Try-except
try:
    szam = int(input("Adj meg egy számot: "))
    eredmény = 10 / szam
    print(f"Eredmény: {eredmény}")
except ValueError:
    print("Nem számot adtál meg!")
except ZeroDivisionError:
    print("Nullával nem lehet osztani!")
except Exception as e:
```

```

        print(f"Váratlan hiba: {e}")
else:
    print("Minden rendben ment!")
finally:
    print("Ez mindig lefut")

# Saját kivétel
class HibásÉrtékHiba(Exception):
    pass

def ellenőriz(szam):
    if szam < 0:
        raise HibásÉrtékHiba("A szám nem lehet negatív!")
    return szam * 2

```

## Modulok és csomagok

```

# Importálás
import math
print(math.pi)

# Részleges import
from math import sqrt, pi
print(sqrt(16))

# Álnév használata
import numpy as np

# Saját modul
# fajl: segito.py
def udvozles(nev):
    return f"Hello, {nev}!"

# masik_fajl.py
import segito
print(segitto.udvozles("Világ"))

```

## Hasznos beépített függvények

```

# Tipusok
type(42)                      # <class 'int'>
isinstance(42, int)             # True

# Konverziók
str(), int(), float(), list(), tuple(), set(), dict()

```

```

# Matematikai
abs(-5)                      # 5
round(3.7)                     # 4
min(1, 2, 3)                   # 1
max(1, 2, 3)                   # 3
sum([1, 2, 3])                 # 6
pow(2, 3)                      # 8

# Iterálható
len([1, 2, 3])                 # 3
sorted([3, 1, 2])               # [1, 2, 3]
reversed([1, 2, 3])             # [3, 2, 1]
enumerate([('a', 'b')])         # [(0, 'a'), (1, 'b')]
zip([1, 2], ['a', 'b'])          # [(1, 'a'), (2, 'b')]

# Szűrés és átalakítás
map(lambda x: x*2, [1, 2, 3])    # [2, 4, 6]
filter(lambda x: x>2, [1, 2, 3])  # [3]

# Input/Output
input("Kérdés: ")
print("Kimenet")

```

## Összefoglalás

A Python egy rendkívül sokoldalú nyelv, amely alkalmas: - Webfejlesztésre (Django, Flask) - Adatelemzésre (Pandas, NumPy) - Gépi tanulásra (TensorFlow, PyTorch) - Automatizálásra - Játékfejlesztésre - És még sok másra!

A gyakorlás a kulcs a Python elsajátításához. Kezdj egyszerű projektekkel és fokozatosan haladj a bonyolultabb feladatok felé!