Python programozás

Jegyzet

Tartalomjegyzék

1 Helló, világ!

1.1 Írás konzolra

```
1 print("Helló, világ!")
```

```
>>> "Helló, világ!"
```

1.2 Kommentek

```
1 # Ez egy komment
2
3 # Több soros komment
4 """
5 Több
6 soros
7 komment
8 """
```

2 Primitív adattípusok

Másnéven, előre meghatározott (alapvető) adattípusok.

Értékadás szintaxisa: `<változó> = <érték>`

- Szöveg (Text): str (string)
- Szám (Number): int, float, complex
- Logikai (Boolean): bool
- Értékhiány (None): NoneType

2.1 Szöveg (str)

```
1    greeting = "Helló"
2    name = "XYZ"
```

```
>>> greeting = "Helló"
>>> name = "XYZ"
```

2.1.1 Karakter

```
1    character = "X"
```

```
>>> character = "X"
```

2.1.2 String-ek összefűzése

```
1    greeting_user = greeting + ", " + name
```

```
>>> greeting_user = "Helló, XYZ"
```

2.1.3 Ugyanazon érték megadása több változónak egyszerre

```
    friend_name = "XYZ
    friend1 = friend2 = friend3 = friend_name
```

```
>>> friend_name = "XYZ"
>>> friend1 = friend2 = friend3 = "XYZ"
```

2.1.4 Többsoros string

```
    multi_line_string = """Ez egy
    többsoros
    string"""
```

```
>>> multi_line_string = """Ez egy
... többsoros
... string"""
```

2.1.5 Alapvető string műveletek

```
message = "Ez egy üzenet!"
upper_message = message.upper() # Nagybetűssé alakítás
lower_message = upper_message.lower() # Kisbetűssé alakítás
remove_whitespace = message.strip() # VEZETŐ és VÉG szóközök eltávolítása
replace_message = message.replace("üzenet", "szöveg") # Csere
```

```
>>> message = "Ez egy üzenet!"
>>> upper_message = "EZ EGY ÜZENET!"
>>> lower_message = "ez egy üzenet!"
>>> remove_whitespace = "Ez egy üzenet!"
>>> replace_message = "Ez egy szöveg!"
```

2.1.6 Bemenet

`input()` függvény: A felhasználótól vár bemenetet, és azt stringként adja vissza.

```
1    user_input = input("Kérem, adja meg a nevét: ")
2    greeting_message = greeting + ", " + user_input
```

```
>>> user_input = input("Kérem, adja meg a nevét: ")
Kérem, adja meg a nevét: XYZ
>>> greeting_message = "Helló, XYZ"
```

2.2 Szám (Number)

2.2.1 Egész szám (int)

```
1    integer_number = 10
```

```
>>> integer_number = 10
```

2.2.2 Lebegőpontos szám (float)

```
1    float_number = 10.5
```

```
>>> float_number = 10.5
```

2.2.3 Komplex szám (complex)

`a + bj`, ahol `a` és `b` valós számok, `j` pedig az imaginárius egység.

```
1    complex_number = 1 + 2j
```

```
>>> complex_number = 1 + 2j
```

2.2.4 Aritmetikai műveletek

```
addition = 10 + 5 # Összeadás
subtraction = 10 - 5 # Kivonás
multiplication = 10 * 5 # Szorzás
division = 10 / 5 # Osztás
modulo = 10 % 5 # Maradékos osztás
```

```
>>> addition = 15
>>> subtraction = 5
>>> multiplication = 50
>>> division = 2.0
>>> modulo = 0
```

2.2.5 Értékadó operátorok

```
number = 10
number += 5 # number = number + 5
number -= 5 # number = number - 5
```

```
>>> number = 10
>>> number += 5
>>> number -= 5
```

2.3 Logikai (Boolean)

Két értéke lehet: `True` (1) vagy `False` (0).

2.3.1 Igazságtábla

A	B	A ÉS B
1	1	1
1	0	0
0	1	0
0	0	0

`A ÉS B` csak akkor igaz, ha `A` és `B` is igaz.

A	B	A VAGY B
1	1	1
1	0	1
0	1	1
0	0	0

`A VAGY B` akkor igaz, ha legalább az egyik igaz.

2.3.2 Összehasonlítás (érték alapján)

```
1     is_equal = 10 == 10
2     is_not_equal = 10 != 5
3     is_greater = 10 > 5
4     is_less = 10 < 5
5     is_greater_or_equal = 10 >= 10
6     is_less_or_equal = 10 <= 5
```

```
>>> is_equal = True
>>> is_not_equal = True
>>> is_greater = True
>>> is_less = False
>>> is_greater_or_equal = True
>>> is_less_or_equal = False
```

2.3.3 Logikai műveletek

Lásd: Igazságtábla.

```
number = 5
and_result = number == 5 and number != 2  # True, ha mindkettő igaz
or_result = number == 10 or number == 5  # True, ha legalább az egyik igaz
not_result = not True  # True, ha az érték hamis
```

```
>>> and_result = True
>>> or_result = True
>>> not_result = False
```

2.3.4 Objektumok összehasonlítása

```
x = 9
y = x
is_result = x is y
is_not_result = x is not y
```

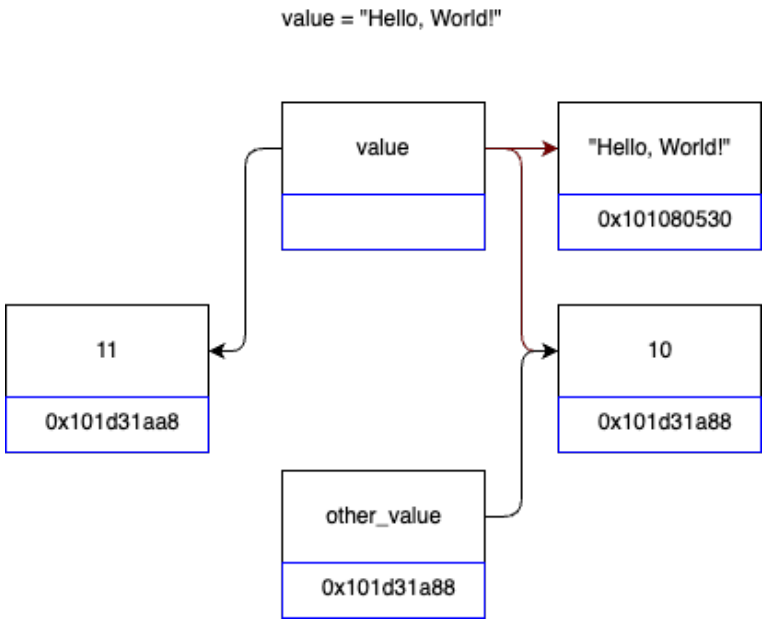
```
>>> is_result = True
>>> is_not_result = False
```

2.3.5 Tartalmazás vizsgálata

```
    text = "Hello, World!"
    is_in = "Hello" in text
    is_not_in = "Python" not in text
```

2.4 Hogyan tárolódnak a memóriában a változók?

```
1  value = "Hello, World!"
2  print(hex(id(value)))
3  print(type(value))
4
5  print("\n")
6
7  value = 10
8  print(hex(id(value)))
9  print(type(value))
10
11 other_value = value
12 print(hex(id(other_value)))
13 print(type(other_value))
14
15 value = 11
16 print(hex(id(value)))
17 print(type(value))
18
19 print("\n")
20 del value  # Névtérből törlődik csak
21 # print(value)
```



3 Indexelés

Az indexelés a szövegekben és listákban való elemek elérésére szolgál. Az indexelés 0-tól kezdődik, azaz az első elem indexe 0, a másodiké 1, és így tovább. Az indexelés negatív számokkal is lehetséges, ahol az utolsó elem indexe -1, az utolsó előttié -2, és így tovább.

3.1 Pozitív indexelés

```
text = "Hello, World!"
first_character = text[0]
second_character = text[1]
third_character = text[2]
last_character = text[-1]
second_last_character = text[-2]
```

3.2 Negatív indexelés

```
text = "Hello, World!"
last_character = text[-1]
second_last_character = text[-2]
```

3.3 Szeletelés

```
text = "Hello, World!"
first_word = text[:5]
second_word = text[7:]
middle_word = text[5:12]
```

4 Feltételek

A feltételek segítségével tudjuk irányítani a program futását. A feltételek egy logikai kifejezésből és egy vagy több utasításból állnak. Ha a feltétel igaz, akkor az utasítások lefutnak, ha hamis, akkor nem.

4.1 Egyágú elágazás

```
number = 5
if number > 0:
    is_positive = True
else:
    is_positive = False
```

4.2 Többágú elágazás

```
number = 5
if number > 0:
    is_positive = True
elif number == 0:
    is_zero = True
else:
    is_negative = True
```

4.3 Beágyazott feltételek

```
number = 5
if number > 0:
    if number % 2 == 0:
        is_positive_even = True
    else:
        is_positive_odd = True
else:
    is_negative = True
```

4.4 Short-hand feltételek

```
number = 5
is_positive = True if number > 0 else False
```

4.5 Match (switch-case)

```
number = 5
match number:
    case 0:
        is_zero = True
    case 1:
        is_one = True
    case 2:
        is_two = True
    case _:
        is_other = True
```

5 Függvények

A függvények olyan kód blokkok, melyeket újrahasználhatunk. A függvények segítségével csoportosíthatjuk a kódot, és egyszerűbben olvashatóvá tehetjük azt.

5.1 WET (Write Everything Twice) vs. DRY (Don't Repeat Yourself)

A WET (Write Everything Twice) és a DRY (Don't Repeat Yourself) két különböző programozási elv. A WET azt jelenti, hogy minden kódot kétszer vagy többször is megírunk, míg a DRY azt jelenti, hogy a kódot egyszer írjuk meg, majd újrahasználjuk.

5.2 Függvény létrehozása

```
1    def greet():
2        print("Hello, World!")
```

5.3 Függvény meghívása

```
1    greeting = greet()
```

5.4 Paraméterek

```
1    def greet(name):
2        return f"Hello, {name}!"
```

5.5 Paraméterek alapértelmezett értékkel

```
    def greet(name="World"):
        return f"Hello, {name}!"

    greeting = greet()
    greeting_with_name = greet("Alice")
```

5.6 Visszatérési érték

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

    result = add(5, 3)
```

5.7 Visszatérési érték nélkül

```
    def greet(name):
        print(f"Hello, {name}!")

    greet("Alice")
```