Tema 2.
Variabile, tipuri de date și operatori în Python

Ce ne așteaptă?

1. Variabile în Python

2. Tipuri de date în Python

3. Operatori în Python

1. Ce este o variabilă în Python?
2. Cum se declară o variabilă în Python?
3. Ce este un tip de date?
4. Ce tipuri de date sunt în Python?
5. Care sunt tipurile de date numerice în Python?
6. Care sunt tipurile de date secvențiale în Python?
7. Ce este un operator?
8. Care sunt operatorii aritmetici din Python?
9. Care sunt operatorii relaționali din Python?
10. Care sunt operatorii logici din Python?
11. Care sunt operatorii de atribuire din Python?
12. Care sunt operatorii de apartenență din Python?
13. Care sunt operatorii de identificare din Python?

1. Variabilele

Noțiune de variabilă și proprietățile ei

- **Variabila** – numele locației de memorie în care se păstrează o valoare

- **Proprietățile variabilei:**

- Tipul variabilei

- Vizibilitatea variabilei

- Global
- Locale

- Valoarea variabilei

- Locația variabilei

- Timpul de viață a variabilei

Declararea variabilei

- **Python – limbaj dynamically typed = nu necesită specificarea tipului**

- **Declararea variabilei necesită:**

  - numele variabilei

  - valoarea atribuită variabilei

- **Sintaza declarării:**

  nume_variabilă = valoare

- **Exemple:**

```
vârstă = 30

nume = "Pavel"
```

Particularități în declararea variabilelor

- Numele în stânga, valoarea în dreapta expresiei

```
vârstă = 30 # corect
30 = vârstă # greșit SyntaxError
```

- Declararea variabilelor multiple într-o singură linie

```
a,b,c = 10,20,30 # corect
a,b,c = 10,20 # greșit ValueError
a,b = 10,20,30 # greșit ValueError
```

- Declararea variabilelor multiple de o singură valoare

```
a = b = c = 10
```

- Reinițializarea variabilei

```
a = 30
a = "Pavel"
```

2. Tipuri de date

Tipuri de tipuri de date

- Tipuri de date încorporate
- Tipuri de date definite de utilizator

- Funcția type()

```
a = 10
print(a)
print(type(a))
```

- Metoda sys.getsizeof(nume_variabilă | varaibilă)

```
import sys
a = 10
print(a)
print(type(a))
print(sys.getsizeof(a))
```

Tipuri de date numerice

- Date de tip *int*

```
a = 10
print(a)
print(type(a))
```

- Date de tip *float*

```
a = 10.6
b=2E2
print(a)
print(b)
print(type(a))
print(type(b))
```

- Date de tip *complex*

```
a = 5+7j
print(a)
print(type(a))
```

Sisteme de reprezentare a numerelor întregi

- **Reprezentare zecimală (implicită)**

```
a = 10
print(a)
```

- **Reprezentare binară (0b... sau 0B...)**

```
a = 0b10
print(a)
```

- **Reprezentare octală (0o... sau 0O...)**

```
a = 0o10
print(a)
```

- **Reprezentare hexazecimala (0x... sau 0X...)**

```
a = 0x10
print(a)
```

Tipuri de date bool și None

- **Date de tip *bool***

```
a = True
b = False
print(a)
print(b)
print(type(a))
print(type(b))
```

https://docs.python.org/2.4/lib/truth.html

https://www.freecodecamp.org/news/truthy-and-falsy-values-in-python/

- **Date de tip None**

```
a = None
print(a)
print(type(a))
```

Tipuri de date secvențiale

- Date de tip *str* `a = 'Pavel'`
- Date de tip *list* `a = [10, 'Pavel', 3.2, True]`
- Date de tip *tuple* `a = (10, 'Pavel', 3.2, True)`
- Date de tip *bytes* `a = bytes([10, 200, 150, 37])`
- Date de tip *bytearray* `a = bytearray([10, 200, 150, 37])`
- Date de tip *range* `a = range(10)`

Alte tipului de date

- **Date de tip *range***

  - Funcția range() – creează o secvență de numere întregi consecutive într-o gamă specificată

  range(*start*, *stop*, *pas*)
  - start – numărul de început al gamei (valoare implicită = 0)
  - stop – primul număr ce nu se cuprinde în gama de valori (argument obligatoriu)
  - pas – pasul de selecție a numerelor din gama de valori (valoare implicită = 1)

```
a = range(10)
print(a)
for i in a:
    print(i)
print(type(a))
```

- **Date de tip *set*** `a = {10, 'Pavel', 3.2, True}`
- **Date de tip *dict*** `a = {'Prenume':'Pavel', 'Nume':'Nicolaev', 'vârstă':32}`

Conversia tipului de date

- Tipuri fundamentale de date
  1. int
  2. float
  3. complex
  4. bool
  5. str

- Funcții de conversie

int()	`a = int(3.2)`
float()	`a = float(5)`
complex()	`a = complex(3.2)`
bool()	`a = bool(7)`
str()	`a = str(22)`

6. Operatori

Clasificarea operatorilor

- **Operatori unari**

```
a = -5
```

- **Operatori *binari***

```
a = 3 * 'Pavel'
```

- **Operatori ternari**

x = valoare_adevarat* if *condiție* else *valoare_fals

```
a = 10 if b==3 else 11
```

Tipuri de operatori

- **Operatori aritmetici:**
  +, -, *, /, %, **, //
- **Operatori relaționali (de comparație):**
  <, >, <=, >=, ==, !=
- **Operatori logici:**
  ‘and’, ‘or’, ‘not’
- **Operatori logici bit cu bit:**
  ‘&’, ‘|’, ‘^’
- **Operatori de atribuire:**
  ‘=’, ‘+=’, -=
- **Operatori de apartenență:**
  ‘in’, ‘not in’
- **Operatori de identificare:**
  ‘is’, ‘is not’

Operatori aritmetici

- Adunare: +
- Scăderea: -
- Înmulțirea: *
- Împărțirea: /
- Împărțirea întreg: //
- Ridicarea la putere: **
- Restul divizării: %

```
a = 20
b = 12
print(a+b)
print(a-b)
print(a*b)
print(a/b)
print(a//b)
print(a**b)
print(a%b)
```

Valori ale operatorilor aritmetici

- **Reguli generale:**
  - Ambele valori sunt de tip *int* – rezultatul este *int*
  - Cel puțin o valoare *float* – rezultatul *float*

```
a = 20.3
b = 12
c = a // b
print(c)
print(type(c))
```

- **Reguli operatorul / :**
  - Rezultatul va fi *float* indiferent de tipul valorilor

```
a = 20
b = 12
c = a / b
print(c)
print(type(c))
```

Operatori relaționali (de comparație)

- Egal: ==
- Diferit de: !=
- Mai mare: >
- Mai mic: <
- Mai mare și egal: >=
- Mai mic și egal: <=

```
a = 13
b = 5
print(a>b)
print(a>=b)
print(a<b)
print(a<=b)
print(a==b)
print(a!=b)
```

Operatori logici

- **Cu valori logice:**
  - **and** : returnează False daca cel puțin o valoare este False
  - **or** : returnează True dacă cel puțin o valoare este True
  - **not** : returnează valoarea opusă

```
a, b = True, False
print(a and b)
print(a or b)
print(not a)
```

- **and cu valori non-boolean :**

  '**a and b**' returnează 0 dacă a este 0
  '**a and b**' returnează b dacă a nu este 0

- **or cu valori non-boolean :**

  '**a or b**' returnează a dacă a nu este 0
  '**a or b**' returnează b dacă a este 0

```
a = 0
b = 7
print(a and b)
print(a or b)
```

Operatori logici bit cu bit

- Numerele zecimale se transforma binare și se realizează operațiile logice între biți

- **Operația și - '&':**

```
a = 3 & 5
print(a)
```

$3_{10} = 011_2$ &
$5_{10} = 101_2$
$001_2 = 1_{10}$

Arg A	Arg B	&
0	0	0
0	1	0
1	0	0
1	1	1

- **Operația sau - '|' :**

```
a = 3 | 5
print(a)
```

$3_{10} = 011_2$ |
$5_{10} = 101_2$
$111_2 = 7_{10}$

Arg A	Arg B	|
0	0	0
0	1	1
1	0	1
1	1	1

- **Operația sau cu excludere - '^' :**

```
a = 3 ^ 5
print(a)
```

$3_{10} = 011_2$
$5_{10} = 101_2$ ^
$110_2 = 6_{10}$

Arg A	Arg B	^
0	0	0
0	1	1
1	0	1
1	1	0

Operatori de atribuire

- **Atribuire simplă: =**

```
a = 13
```

- **Atribuire combinată: +=, -=, *=, /=, %=, **=, //=**

a += b sau a=a+b

```
a=13
print(a)
a+=5
print(a)
```

Operatori de apartenență și de identificare

- **Operatori de apartenență:**
  - **in** : returnează True daca valoarea se găsește în secvență și False dacă nu
  - **not in** : returnează False daca valoarea se găsește în secvență și True dacă nu

```
text = "Bun venit la UTM"
print("UTM" in text)
print("venit" not in text)
print("masterat" in text)
print("masterat" not in text)
```

- **Operatori de identificare:**
  - is : întoarce True daca valorile se găsesc la aceeași adresă de memorie și False dacă nu
  - is not: întoarce False daca valorile se găsesc la aceeași adresă de memorie și True dacă nu

funția id()

```
a = 25
b = 25
print(a is b)
print(id(a))
print(id(b))
```

Prioritățile operatorilor

Prioritate	Operator	Nota
1	()	
2	+, -	unary
3	**	
4	*, /, //, %	
5	+, -	binary
6	<, <=, >, >=	
7	==, !=	
8	&	
9	|	
10	=, +=, -=, *=, /=, %=, &=, ^=, |=, >>=, <<=	