TALENTO TECH

TIC

Universidad Tecnológica de Bolívar

Estructuras 2 en Python

Ejecutor técnico: Jorge Luis Villalba Acevedo

www.utb.edu.co/talento-tech

Scalar Types

Tipo de Dato	Descripción	Ejemplos
`int`	Números enteros (positivos, negativos, o cero).	`1, -5, 0, 42`
`float`	Números de punto flotante (decimales).	`3.14, -0.001, 2.0`
`complex`	Números complejos (con parte real e imaginaria).	`3 + 5j, 1.5 - 2j`
`bool`	Tipo booleano, representa valores de verdad.	`True, False`

Tipo de Dato	Descripción	Ejemplos
`NoneType`	Representa la ausencia de valor (tipo especial).	`None`
`str`	Cadena de caracteres (texto).	`'hello'`, `"Python"`, `'''multilínea'''`
`bytes`	Secuencia inmutable de bytes (datos binarios).	`b'abc'`, `b'\x89PNG\r\n\x1a\n'`

Estos son los tipos de datos escalares más comunes en Python, que representan valores individuales en lugar de estructuras de datos complejas.

Ejemplos

```
ival = 17239871 # int

fval = 7.243    # float

a = 'one way of writing a string' # Strings

si = True  #Booleans
```

Type casting

```
s = "3.14159"
fval = float(s)
type(fval)
int(fval)
bool(fval)
bool(0)
```

None

```
a = None

a is None

b = 5

b is not None
```

function arguments

```
def add_and_maybe_multiply(a, b, c=None):
    result = a + b

    if c is not None:
        result = result * c

    return result
```

function arguments

```
add_and_maybe_multiply(5, 6)

add_and_maybe_multiply(5, 6, 2)
```

Control Flow

if, elif, and else

```
x = -5
if x < 0:
    print("It's negative")

if x < 0:
    print("It's negative")
elif x == 0:
    print("Equal to zero")
elif 0 < x < 5:
    print("Positive but smaller than 5")
else:
    print("Positive and larger than or equal to 5")
```

if, elif, and else

```
a = 5; b = 7

c = 8; d = 4

if a < b or c > d:
    print("Hecho")
```

if, elif, and else

```
4 > 3 > 2 > 1
```

for loops

```
for value in collection:
    # do something with value
```

for loops

```
sequence = [1, 2, None, 4, None, 5]
total = 0
for value in sequence:
    if value is None:
        continue
    total += value
    print(total)
```

Data Structures and Sequences

Tuple

```
tup = (4, 5, 6)

tup
```

Tuple

```
tup = 4, 5, 6

tup


tup[0]
```

- You can convert any sequence or iterator to a tuple

```
tuple([4, 0, 2])
```

```
tup = tuple('string')
```

List

```
a_list = [2, 3, 7, None] # Creando una lista
tup = ("foo", "bar", "baz")
b_list = list(tup)
b_list
b_list[1] = "peekaboo"
b_list
```

Dictionary

```
empty_dict = {}
d1 = {"a": "some value", "b": [1, 2, 3, 4]}
d1
d1[7] = "an integer"
d1
d1["b"]
```

Functions

```
def my_function(x, y):
    return x + y

my_function(10,25)

result = my_function(10,25)

result
```

Functions

```
def my_function2(x, y, z=1.5):
    if z > 1:
        return z * (x + y)
    else:
        return z / (x + y)
```

Functions

```
my_function2(5, 6, z=0.7)

my_function2(3.14, 7, 3.5)

my_function2(10, 20)
```

Referencias

1. **McKinney, W. (2017).** *Python for data analysis: Data wrangling with pandas, numpy, and ipython* (2nd ed.). O'Reilly Media.
2. **Sweigart, A. (2019).** *Automate the boring stuff with Python: Practical programming for total beginners* (2nd ed.).
3. **González Duque, R. (2015).** *Python para todos*. Independently published.
4. **Bosch, J. (2020).** *Introducción a la programación en Python: Algoritmos y lógica para principiantes*. Independently published.