Python 3

WPROWADZENIE

Fundusze Europejskie
Wiedza Edukacja Rozwój

Unia Europejska
Europejski Fundusz Społeczny

Python 3

Język skryptowy

Interpretowany, nie kompilowany

Brak silnego typowania, ale typy zmiennych mają znaczenie

Brak narzuconej struktury kodu

Python 3 - biblioteki

Dostępnych jest bardzo dużo bibliotek o różnym przeznaczeniu:

NAUKOWE

SZTUCZNA INTELIGENCJA

TWORZENIE GIER

TWORZENIE APLIKACJI MOBILNYCH

I WIELE INNYCH

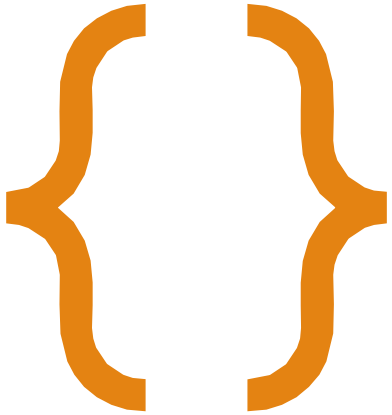
Python 3 – zarządzanie zależnościami

Do zarządzania zależnościami i automatycznego instalowania i aktualizowania bibliotek służy polecenie **pip**

Za pomocą jednego polecenia możemy zainstalować potrzebną bibliotekę:

pip install [nazwa biblioteki]

Wymagane biblioteki dla projektu zazwyczaj umieszczamy w pliku **requirements.txt**



Python 3

KONSTRUKCJE JĘZYKA

Komentarze

 # komentarz zwykły

"""
komentarz
dokumentacyjny
"""

Zmienne

liczba = 34

tekst = **"Hello World"**

liczba_rzeczywista = 3.4

Operatory arytmetyczne

```
a = 10
b = 3

suma = a + b              # 13
iloczyn = a * b           # 30
iloraz = a / b            # 3.3333...
modulo = a % b            # 1
potega = a ** b           # 1000
```

Operatory przypisania

```
a = 10    # 10
a += 1    # 11
a -= 1    # 10
a *= 2    # 20
a /= 2    # 10.0
a %= 4    # 2.0
a **= 3   # 8.0
```

Operatory porównania

```
a = 10
b = 5

a == b    # False
a != b    # True
a < b     # False
a <= b    # False
a > b     # True
a >= b    # True
```

Wejście

```
tekst = input("Podaj tekst")
```

```
liczba = int(input("Podaj liczbę"))
```

Wyjście

```
print("Hello World!")

a = 10
print("a = " + str(a))
print(f"a = {a}")
```

Instrukcja warunkowa

```
temperatura = 25

if temperatura < 10:
    print("Zimno!")
elif temperatura < 20:
    print("Cieplo!")
else:
    print("Goraco!")
```

Pętla warunkowa while

```
x = 0
while x < 10:
    print(f"x = {x}")
    x += 1
```

Pętla licząca for

```
for i in range(0, 10):
    print(f"i = {i}")
```

Pętla licząca for z ujemnym krokiem

```
for i in range(10, 0, -1):
    print(f"i = {i}")
```

Listy

```
pusta_lista = []
print(len(pusta_lista))  # 0

wypelniona_lista = [1, 2, 3, 4, 5]
print(len(wypelniona_lista))  # 5
```

Listy c.d.

```
lista = []
print(len(lista))   # 0

lista.append(10)
print(len(lista))   # 1

lista.append(20)
lista.append(30)
print(len(lista))   # 3
print(lista)        # [10, 20, 30]

print(lista[0])     # 10
print(lista[1])     # 20
print(lista[2])     # 30

print(lista[-1])    # 30
print(lista[-2])    # 20
```

Funkcje

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


print(suma(2, 5))   # 7
```

Funkcje c.d.

```
def punkt(a):
    x = a
    y = a * 2
    return x, y


print(punkt(5))   # (5, 10)
```

Funkcje - typowanie

```
def iloczyn(a: int, b: int) -> int:
    return a * b
```

Klasy

```
class Prostokat:

    def __init__(self, wysokosc, szerokosc):
        self.wys = wysokosc
        self.szer = szerokosc

    def pole(self):
        return self.wys * self.szer

    def czy_kwadrat(self):
        return self.wys == self.szer


prostokat = Prostokat(5, 10)
print(prostokat.pole())
print(prostokat.czy_kwadrat())

prostokat.wys = 10
print(prostokat.czy_kwadrat())
```