Wprowadzenie do programowania w Pythonie

Projekt „Matematyka dla ciekawych świata”
Robert Paciorek

2018-04-05

Python:

- skryptowy
- interpretowany
- interaktywna linia poleceń
- wykonywanie plików z treścią skryptu

Python:

- skryptowy
- interpretowany
- interaktywna linia poleceń
- wykonywanie plików z treścią skryptu

Dlaczego Python?

W trybie interaktywnym można korzystać z Pythona jak kalkulatora:

```
>>> (2+3)/2
2.5
>>> 12*876 + 3
10515
>>> 2**5
32
>>>
>>> from math import *
>>> sqrt(33)
5.744562646538029
>>> sin(30)
-0.988031624092861 8
```

```
>>> a = 5
>>> b = 10; c =8
>>> 2 * b
20
>>> c + a
13
>>>
>>> x, y = 12, 13
>>> x + a*y
77
>>>
>>> a, b = x, x + a
```

```
>>> a = 5
>>> b = 10; c =8
>>> 2 * b
20
>>> c + a
13
>>>
>>> x, y = 12, 13
>>> x + a*y
77
>>>
>>> a, b = x, x + a
>>> a
```

```
>>> a = 5
>>> b = 10; c =8
>>> 2 * b
20
>>> c + a
13
>>>
>>> x, y = 12, 13
>>> x + a*y
77
>>>
>>> a, b = x, x + a
>>> a
12
```

```
>>> a = 5
>>> b = 10; c =8
>>> 2 * b
20
>>> c + a
13
>>>
>>> x, y = 12, 13
>>> x + a*y
77
>>>
>>> a, b = x, x + a
>>> a
12
>>> b
```

```
>>> a = 5
>>> b = 10; c =8
>>> 2 * b
20
>>> c + a
13
>>>
>>> x, y = 12, 13
>>> x + a*y
77
>>>
>>> a, b = x, x + a
>>> a
12
>>> b
17
```

- instrukcje kończy znak nowej linii
- bloki wydzielane wcięciami
- rozpoczęcie bloku poprzedza dwukropek

Pętla **for** służy do iterowania:

- po zakresie wartości numerycznych (całkowitych)
- po elementach jakiejś kolekcji

Pętla **for** służy do iterowania:

- po zakresie wartości numerycznych (całkowitych)
- po elementach jakiejś kolekcji

W Pythonie iterowanie po zakresie wartości całkowitych to iterowanie po kolekcji takich liczb.

```
>>> for x in [1, 2, 3, 4]:
...     print(x)
...
1
2
3
4
```

```
>>> for x in [1, 2, 3, 4]:
...     print(x)
...
1
2
3
4
```

```
>>> for x in range(1,5):
...     print(x)
...
1
2
3
4
```

Prawdziwe iterowanie po zakresie wartości numerycznych (oparte na sprawdzaniu warunku) można uzyskać korzystając z pętli **while**:

```
>>> x = 1
>>> while(x<5):
...     print(x)
...     x += 1
...
1
2
3
4
```

Warunkowe wykonywanie bloku kodu umożliwia instrukcja **if**:

```
>>> x = 0
>>>
>>> if x > 5:
...     print("większy od 5")
... elif x == 4:
...     print("równy 4")
... else:
...     print("jakiś inny")
...
jakiś inny
```

Blok kodu do którego możemy odwołać się z użyciem nazwy:

```
>>> def aaa():
...     print("AAA")
...
>>> aaa()
AAA
```

Blok kodu do którego możemy odwołać się z użyciem nazwy:

```
>>> def aaa():
...     print("AAA")
...
>>> aaa()
AAA
```

```
>>> def bbb(n):
...     for i in range(n):
...         print("BBB")
...
>>> bbb(2)
BBB
BBB
```

Podstawowe, wbudowane typy zmiennych w Pythonie:

- liczby całkowite
- liczby zmiennoprzecinkowe
- wartości logiczne (True/False)
- napisy
- kontenery

Podstawowe, wbudowane typy zmiennych w Pythonie:

- liczby całkowite
- liczby zmiennoprzecinkowe
- wartości logiczne (True/False)
- napisy
- kontenery

Python ustala typ zmiennej na podstawie przypisanej wartości.

Podstawowe, wbudowane typy zmiennych w Pythonie:

- liczby całkowite
- liczby zmiennoprzecinkowe
- wartości logiczne (True/False)
- napisy
- kontenery

Python ustala typ zmiennej na podstawie przypisanej wartości.

Można tworzyć i wykorzystywać własne typy poprzez definiowanie klas i tworzenie obiektów danej klasy.

Lista jest kolekcją uporządkowanych kolejno elementów. Możemy odwołać się do pierwszego, ostatniego, bądź n-tego elementu.

```
>>> a = [ 13, 15, 16, 17 ]
>>> a[0]
13
>>> a[-1]
17
>>> a[2]
16
```

Lista jest kolekcją uporządkowanych kolejno elementów. Możemy odwołać się do pierwszego, ostatniego, bądź n-tego elementu.

```
>>> a = [ 13, 15, 16, 17 ]
>>> a[0]
13
>>> a[-1]
17
>>> a[2]
16
```

Elementy mogą być różnych typów. Możemy usuwać i wstawiać elementy zarówno na początku, końcu jak i we wnętrzu listy. Do iteracji po listach (i obiektach do nich podobnych) służy pętla for.

Słownik jest kolekcją par klucz - wartość. Klucz jest unikalny i służy do identyfikacji pary (odwoływanie się do elementów słownika odbywa się z użyciem klucza).

```
>>> b = { 'a': 13, 'b': 16, 21: 17 }
>>> b["a"]
13
>>> b[21]
17
```

Słownik jest kolekcją par klucz - wartość. Klucz jest unikalny i służy do identyfikacji pary (odwoływanie się do elementów słownika odbywa się z użyciem klucza).

```
>>> b = { 'a': 13, 'b': 16, 21: 17 }
>>> b["a"]
13
>>> b[21]
17
```

Zarówno klucze jak i wartości mogą być różnych typów. Możemy sprawdzać istnienie klucza w słowniku, iterować po elementach słownika itd.