Programowanie Python - notatki

Anna Sztyber

Wydział Mechatroniki Politechniki Warszawskiej

Spis treści

Zmienne

- Zmienna - **nazwane** miejsce w pamięci, pozwala na zapisywanie i odczytywanie danych
- Nazwy zmiennych mogą zawierać litery, cyfry i _, muszą zaczynać się literą lub _
- Python rozróżnia wielkość liter
- Definicja zmiennej:

```
a = 3
```

Typy zmiennych

- int (integer) liczby całkowite: 3
- float liczby zmiennoprzecinkowe: 3.14
- str (string) napisy: 'napis', "napis"
- bool logiczny (0 lub 1, prawda lub fałsz): True, False

Wydruk

Funkcja print() drukuje:

```
print('hello world')
```

Przypisanie

znak = oznacza przypisanie, czyli podstawienie prawej strony do lewej

```
a = 1
b = 2
c = a + b
x = 3
x = x + 5
```

Konwersja typów

Do sprawdzania typów służy funkcja type:

```
type('napis')
```

Zamiana (konwersja typów):

```
str(2)
int('1')
```

Spis treści

Operatory arytmetyczne

- dodawanie +
- odejmowanie -
- mnożenie *
- dzielenie /
- modulo (reszta z dzielnia)
- potęgowanie **

```
1 + 3.5 * 2 / 7 * 2**2 - 1
```

Kolejność działań

- nawiasy ()
- potęgowanie
- mnożenie i dzielenie
- dodawanie i odejmowanie
- od lewej do prawej

```
(1 + 3.5) * 2 / (7 * 2**(2 - 1))
```

Biblioteka math

Biblioteki (zewnętrzne, gotowe funkcje, które chcemy wykorzystać w swoim programie) dodajemy poleceniem import:

```
import math
```

Pierwiastek:

```
math.sqrt(4)
```

Liczba π:

```
math.pi
```

Spis treści

Wczytywanie danych od użytkownika

- Wczytujemy za pomocą funkcji input
- Argumentem funkcji jest napis (komunikat dla użytkownika)
- Funkcja zwraca napis (str)
- Jeżeli chcemy wykorzystywać wczytane dane np. jako liczby to trzeba dokonać konwersji

```
x = input("Podaj wartość x: ")
```

Instrukcje warunkowe - ify

Warunki sprawdzamy poprzez if [warunek]:

```
x = 2
if x > 1:
    print("x jest większe niż 1")
```

Wcięcia

wyznaczają bloki kodu w Pythonie, pojedyncze wcięcie to 4 spacje

```
x = 2
if x > 1:
    print("x jest większe niż 1")
    print("jestem wewnątrz if")
print("a ja nie")
```

Porównywanie

- $>$ większy
- $<$ mniejszy
- $>=$ większy lub równy
- $<=$ mniejszy lub równy
- $==$ porównanie ($=$ to przypisanie)
- $!=$ nierówny

Złożone warunki

else (w przeciwnym przypadku)

```
x = 3
if x == 2:
    print("x jest równe 2")
else:
    print("x nie jest równe 2")
```

elif (w przeciwnym przypadku, jeżeli)

```
x = 5
if x == 2:
    print("x jest równe 2")
elif x == 3:
    print("x jest równe 3")
else:
    print("inna liczba")
```

Warunki logiczne

- and - i, iloczyn logiczny
- or - lub, alternatywa logiczna
- not - negacja

```
x == 3 or x == 4
x >= 10 and x <= 15
not (x >= 10 and x <= 15)
```

Spis treści

Pętla while

Wykonuje się dopóki warunek jest spełniony

```
x = 5
while x > 0:
    print(x)
    x = x - 1
```

Przerywanie pętli

- break - przerywa pętlę
- continue - przerywa bieżącą iterację pętli

```
x = 5
while x > 0:
    x = x - 1
    if x == 3:
        break
    print(x)
```

```
x = 5
while x > 0:
    x = x - 1
    if x == 3:
        continue
    print(x)
```

None

None - nic, brak wartości, wygodne do ustawiania wartości początkowych

```
x = None
if x is None:
    print('Brak wartości początkowej')
```

Listy

- lista uporządkowanych elementów
- zapisujemy z zastosowaniem [], rozdzielając elementy przecinkami
- może zawierać elementy różnych typów (również inne listy)

```
lista_ciastek = ['brownie', 'shortbread',
                 'owsiane', 'beza']
```

Pętle for

- zazwyczaj o znanej liczbie iteracji
- "zrób coś dla każdego elementu z"
- służy do przetworzenia wszystkich elementów listy, pliku, ...

```
for ciastko in lista_ciastek:
    print(ciastko, ' am am am')
```

Range

- zakres liczb całkowitych od 0 do n-1 (0, 1, 2, ..., n-1)
- range jest generatorem (generatory są leniwe, elementy są tworzone tylko jeśli je wykorzystujemy) - dlatego w celu wydruku zamieniamy na listę

```
print(range(5)) # daje range(0, 5)
print(list(range(5))) # daje [0, 1, 2, 3, 4]

for i in range(5):
    print(i)
```

Losowanie

- do losowania służy biblioteka random
- random.randint(a, b) - zwraca losową liczę z zakresu ¡a, b¿

```
import random
random.randint(2, 5)
```

Pętle zagnieżdżone

- pętla w pętli
- po zakończeniu pętli wewnętrznej rozpoczyna się kolejna iteracja pętli zewnętrznej

```
for i in range(5):
    for j in range(3):
        print("i = ", i,", j = ", j)
```

Listy list

- listy zawierające listy
- możemy coś robić z każdym elementem za pomocą pętli zagnieżdżonej

```
lista_list = [['a', 'b', 'c'], [1, 2]]

for lista in lista_list:
    print(lista)

for lista in lista_list:
    for el in lista:
        print(el)
```

Listy cd

- indeksowanie - lista o rozmiarze n ma indeksy od 0 do n - 1
- długość listy sprawdzamy funkcją len

```
lista = [1, 2, 3]

lista[0] # pierwszy element listy

len(lista) # długość listy
```

Wycinki listy

- lista[pierwszy indeks:drugi indeks]
- pierwszy indeks jest uwzględniany, drugi nie
- pominięcie pierwszego indeksu oznacza, że bierzemy elementy od początku
- pominięcie drugiego indeksu oznacza, że bierzemy elementy do końca

```
lista = ['a', 'b', 'c', 'd']

lista[1:3] # daje ['b', 'c']

lista[:2] # daje ['a', 'b']

lista[1:] # daje ['b', 'c', 'd']
```

Dodawanie i usuwanie elementów

- funkcja append dodaje element do listy
- funkcja remove usuwa element listy (próba usunięcia elementu, którego nie ma na liście powoduje błąd)
- in pozwala sprawdzić, czy lista zawiera dany element

```
lista.append(6)

lista.remove('a')

if 'b' in lista:
    ...
```

Enumerate

- pozwala przejść przez elementy i indeksy listy

```
for i, element in enumerate(lista):
    print("i=", i, ", element=", element)
```

Indeksy list zagnieżdżonych

- pierwszy indeks dotyczy listy wewnętrznej
- drugi indeks dotyczy elementu listy wewnętrznej

```
l2 = [[1, 2], [5, 6]]

l2[0] # daje [1, 2]

l2[0][0] # daje 1

l2[1][1] # daje 6

if 'b' in lista:
    ...
```

Funkcje

Funkcja - ciąg instrukcji zapisany pod jakąś nazwą i wywoływany za pomocą tej nazwy w programie Unikamy powtarzanie kodu:

- programy są krótsze
- łatwiejsza modyfikacja
- trudniej o błędy
- można napisać coś raz i wykorzystywać wielokrotnie

```
def funkcja():
    print("jestem w funkcji")
    print("ja też")
print("a ja na zewnątrz")
```

Przekazywanie parametrów do funkcji

```
def drukuj_dwa_razy(napis):
    print(napis)
    print(napis)

drukuj_dwa_razy("ala ma kota")
```

Zwracanie wartości przez funkcję

- wartość do zwrócenia podajemy po słowie kluczowym return
- return kończy funkcję (nic, co jest później już się nie wykona)

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

wynik = suma(2, 3)
print(wynik)
```

Zwracanie wartości przez funkcję

- Funkcja może zwracać kilka wartości

```
def mnozenie_i_dodawanie(a, b):
    s = a + b
    i = a * b
    return s, i

mnozenie_i_dodawanie(2, 3) # daje (5, 6)
```

Zakres zmiennych

- zmienne zadeklarowane wewnątrz funkcji są dostępne tylko w funkcji (na zewnątrz funkcji już nie)

```
def funkcja_ze_zmiennymi(x):
    a = x + 1
    print(a)

y = 3
funkcja_ze_zmiennymi(y)
a # daje NameError: name 'a' is not defined
```