

Moduł 1

1. Wprowadzenie

Python jest dynamicznym językiem interpretowanym.

Interpretowany tzn. że kod, który napiszemy możemy natychmiast wykonać bez potrzeby tłumaczenia kodu programistycznego na język maszynowy (czyli na formę zrozumiałą przez komputer).

Interpreter tłumaczy nasz kod oraz natychmiast go uruchamia. Interpreter może również być używany w trybie interaktywnym do testowania „kawałków” kodu (np. IPython omawiany w kolejnych scenariuszach).

Naukę zaczynamy od poznania interpretera. Interpreter uruchamiamy z konsoli poleceniem

```
$ python
```

Po uruchomieniu interpretera komputer powinien wypisać trzy linie tekstu a w czwartej tak zwany znak zachęty, po którym wpisujemy komendy. Standardowym znakiem zachęty w interpreterze Pythona jest „>>>”.

2. Zmienne

W interpreterze wpisz poniższy przykład.

Przykład 1.

```
>>> a = 3
>>> b = 5
>>> a + b
8
>>> c = -6.5
>>> a + c
-3.5
>>> 7/2
3.5
>>> 2*6
12
>>> 2**6
64
>>> kwiatek = 'stokrotka'
>>> kwiatek2 = "róża"
>>> kwiatek + ' i ' + kwiatek2
'stokrotka i róża'
```

W powyższym przykładzie wykonujemy operacje na zmiennych. Zmienne są "pudełkami" trzymającymi różne informacje. Zmienna posiada swoją nazwę i wartość. Nazwa zmiennej w Pythonie nie może zawierać:

- polskich znaków
- cyfry na pierwszym miejscu
- spacji
- znaków specjalnych za wyjątkiem podkreślnikiem (_)

Wielkość liter jest rozróżniana (zmienna x i zmienna X to dwie różne zmienne).

**

Przykłady poprawnych:

i, _chodnik, nazwa_23, a2c4

Przykłady niepoprawnych:

1nazwa, 4_strony_świata, z-myslnikiem, nazwa ze spacją,

Główne typy zmiennych, którymi będziemy się zajmować to:

- liczby całkowite (*integer*)
- liczby zmiennoprzecinkowe (*float*)
- napisy, czyli tzw. łańcuchy znaków (*string*)

Istnieje wiele innych typów zmiennych. [odniesienie do zewn. Źródła]

Zróbmy proste zadanie - będąc w interpreterze wykonaj:

Przykład 2.

```
zmienna1 = raw_input("Podaj imię: ")
print("Witaj", zmienna1)
```

Stworzyliśmy właśnie pierwszy program w języku Python. Gratulacje!

Przy użyciu funkcji „raw_input” pobiera on do zmiennej to, co użytkownik wpisze z klawiatury a następnie używając funkcji „print” wyświetla na ekranie kawałek tekstu oraz zawartość wcześniej utworzonej zmiennej.

UWAGA: Funkcja „raw_input” zwraca zmienną typu napisowego (*string*). Nawet, gdy podamy liczbę, to będzie ona traktowana jak napis i nie można wykonywać na niej działań matematycznych. Aby zamienić podaną z klawiatury liczbę na zmienną liczbową (*int* lub *float*) musimy skorzystać z funkcji „int”.

Przykład 3.

```
zmienna1 = raw_input("Podaj 1 liczbę: ")
zmienna2 = raw_input("Podaj 2 liczbę: ")
wynik = int(zmienna1) + int(zmienna2)
print("Suma:", wynik)
```

Funkcje są to wcześniej zdefiniowane kawałki kodu, których możemy później użyć do wykonania określonej czynności, zamiast wpisywać ten sam kod po raz kolejny.

W okrągłych nawiasach po nazwach funkcji umieszczamy parametry lub argumenty funkcji (może być ich więcej niż jeden). Parametry i argumenty oddzielamy od siebie przecinkami.

Jak widać na przykładzie funkcji „raw_input” niektóre funkcje pozostawiają „coś” po sobie. W tym przypadku funkcja „raw_input” pozostawia po sobie to, co użytkownik wpisał z klawiatury a my wrzucamy to do naszej zmiennej „zmienna1”. Kiedy funkcja pozostawia po sobie jakieś dane, mówimy, że funkcja zwraca dane.

Kolejny program zapiszemy już w pliku aby prościej było go zmieniać oraz wykonywać wiele razy. W tym celu należy otworzyć edytor tekstu, wpisać do niego instrukcje języka Python, a następnie zapisać z rozszerzeniem „.py”. Aby uruchomić tak zapisany program należy będąc w linii poleceń (konsola / terminal) w tym samym katalogu gdzie zapisaliśmy nasz plik wpisać:

```
$python nazwa-pliku.py
```

3. Wyrażenia warunkowe

Do podejmowania decyzji w programowaniu służy instrukcja warunkowa „if”.

Blok kodu podany po instrukcji if zostanie wykonany wtedy, gdy wyrażenie warunkowe będzie prawdziwe. W przeciwnym przypadku blok kodu zostanie zignorowany.

Część „else” jest przydatna, jeśli chcemy, żeby nasz program sprawdził wyrażenie warunkowe i wykonał blok kodu jeśli wyrażenie warunkowe jest prawdziwe lub wykonał inny blok kodu jeśli wyrażenie warunkowe było fałszywe.

Python pozwala także na sprawdzenie większej liczby warunków w ramach jednej instrukcji „if”. Służy do tego instrukcja „elif” (skrót od else if).

```
if wyrażenie warunkowe:
    blok kodu 1
elif:
    blok kodu 2
else:
    blok kodu 3
```

Wszystkie instrukcje w bloku kodu muszą być wcięte względem instrukcji „if”. W ten sposób Python rozpoznaje, które instrukcje ma wykonać po sprawdzeniu prawdziwości wyrażenia. Tak samo po instrukcjach „elif” i „else” musimy wstawić dwukropek a instrukcje muszą być wcięte.

Głębokość wcięcia nie ma znaczenia (dobry zwyczaj programowania w Pythonie mówi, żeby używać czterech spacji) ale musi być ono w całym programie zawsze tej samej głębokości.

Pobawmy się instrukcjami „if”, „elif” i „else” na prostym przykładzie.

Przykład 4.

```
zmienna = raw_input('Podaj liczbę: ')
zmienna = int(zmienna)
if zmienna > 0:
    print('Wpisałeś liczbę dodatnią')
elif zmienna == 0:
    print('Wpisałeś zero')
else:
    print('Wpisałeś liczbę ujemną')
print('Koniec programu')
```

W programie na początku wczytywana jest wartość z klawiatury do zmiennej, a następnie dokonujemy zmiany jej typu na liczbę całkowitą. W dalszej części stosujemy instrukcję „if” sprawdzając czy wartość podanej liczby jest większa od 0. Jeśli wartość będzie większa od 0 na ekranie wyświetlony będzie napis „Wpisałeś liczbę dodatnią”, jeśli nie, program wykona kolejną instrukcję: „elif” sprawdzając czy liczba jest równa 0. Jeśli żaden z powyższych warunków nie będzie spełniony wykonane zostanie polecenie zawarte po instrukcji „else”. Program zakończy się wyświetlając: „Koniec programu”.

Jak również widać porównanie w Pythonie, wykonujemy poprzez podwójne użycie znaku równości: „==”. Matematyczne wyrażenie „nie równe” (\neq) w Pythonie zapisujemy jako „!=”.

Przykład 5.

Gra w „zgadnij liczbę”.

Napisz program, w którym:

- do zmiennej „dana” przypiszesz pewną liczbę
- użytkownik będzie mógł podać z klawiatury dowolną liczbę całkowitą
- jeżeli użytkownik trafi program wyświetli komunikat: „Gratulacje!”, a jeśli nie, to wyświetli napis określający czy podana liczba jest większa od danej czy mniejsza.

```
dana = 18
strzal = int(raw_input('Wpisz liczbę całkowitą'))
if strzal == dana:
    print('Gratulacje! Zgadłeś')
elif strzal < dana:
    print('Nie! Szukana liczba jest większa!')
else:
    print('Nie! Szukana liczba jest mniejsza!')
print('Koniec programu.')
```

Zadania dodatkowe

1. Za pomocą poznanych narzędzi stwórz program będący kalkulatorem.
2. Napisz program rozwiązujący równania kwadratowe.

ROZWIĄZANIE:

```
print 'Dla równania kwadratowego  $ax^2+bx+c=0$ '
a=int(raw_input('podaj wartość parametru a: '))
b=int(raw_input('podaj wartość parametru b: '))
c=int(raw_input('podaj wartość parametru c: '))
```

```
delta = b**2-4*a*c
if delta > 0:
    x1 = (-b-delta**(1/2))/(2*a)
    x2 = (-b+delta**(1/2))/(2*a)
    print 'x1 = ', x1, ', x2 = ', x2
elif delta == 0:
    x0 = -b/(2*a)
    print 'x0 = ', x0
else:
    print 'brak rozwiązań'
```

3. Napisz program, który spyta użytkownika ile ma lat, a następnie wyświetli czy osoba ta jest młodzieżą, dzieckiem czy dorosłym (załóżmy, że dziecko ma mniej niż 12 lat, a dorosły więcej niż 18).
4. Napisz program, który będzie sortował trzy podane przez użytkownika liczby.
5. Napisz program, który w odpowiedzi na podaną przez użytkownika liczbę będzie wyświetlał komunikat czy jest to liczba parzysta, czy nieparzysta.
6. Napisz program, który będzie sprawdzał czy z podanych przez użytkownika trzech długości można zbudować trójkąt.

4. Pętla WHILE

Pętla while służy do konstrukcji bloku instrukcji, które będą wykonywane warunkowo. W programie najpierw będzie sprawdzane czy warunek jest spełniony – jeśli tak, to wykonane będą wszystkie instrukcje zawarte w bloku. Następnie ponownie sprawdzany jest warunek, jeśli nadal jest spełniony to ponownie wykonuje wszystkie polecenia. Pętla jest wykonywana tak długo, jak długo warunek jest prawdziwy.

```
while wyrażenie warunkowe:  
    blok kodu
```

Zobaczmy działanie pętli „while” na poniższym przykładzie.

Przykład 6.

```
dana = 18  
kontynuuj = True  
while koniec:  
    strzal = int(raw_input('Wpisz liczbę całkowitą'))  
    if strzal == dana:  
        print('Gratulacje! Zgadłeś')  
        kontynuuj = False  
    elif strzal < dana:  
        print('Nie! Szukana liczba jest większa!')  
    else:  
        print('Nie! Szukana liczba jest mniejsza!')  
print('Koniec programu.')
```

Program będzie wykonywany do momentu, w którym użytkownik poda właściwą liczbę. Zatem nie trzeba do każdego strzału ponownie uruchamiać programu. Zmienna „kontynuuj” ma ustawioną wartość logiczną „True” (z angielskiego *prawda*). W momencie, w którym użytkownik poda właściwą liczbę zmienna przyjmie wartość logiczną „False” (z angielskiego *fałsz*), co spowoduje zakończenie wykonywania pętli while.

5. Wyrażenia break i continue

Wyrażenie „break” powoduje natychmiastowe zakończenie wykonywania pętli.

Przykład 7.

```
dana = 18
while True:
    strzal = int(raw_input('Wpisz liczbę całkowitą'))
    if strzal == dana:
        print('Gratulacje! Zgadłeś')
        break
    elif strzal < dana:
        print('Nie! Szukana liczba jest większa!')
    else:
        print('Nie! Szukana liczba jest mniejsza!')
print('Koniec programu.')
```

Wyrażenie „continue” powoduje ominięcie następujących po nim wyrażen w bloku, a następnie rozpoczyna ponowne wykonanie pętli.

Przykład 8.

```
dana = 18
while True:
    strzal = int(raw_input('Wpisz liczbę całkowitą'))
    if strzal > dana:
        print('Nie! Szukana liczba jest mniejsza!')
        continue
    elif strzal < dana:
        print('Nie! Szukana liczba jest większa!')
        continue
    print('Gratulacje! Zgadłeś')
    break
print('Koniec programu.')
```

Zadania dodatkowe

1. Napisz program, który sumuje liczby dodatnie podawane przez użytkownika – pętla pozwala użytkownikowi podawać liczby dopóki nie poda liczby niedodatniej.

Następnie obok podawanego wyniku będzie wyświetlana liczba określająca ilość podanych liczb.

2. Na podstawie wcześniejszego zadania napisz program obliczający średnią liczb dodatnich, a następnie zmodyfikuj go tak, aby obliczana była średnia również dla liczb ujemnych.

6. Pętla FOR

Pętla for służy do wykonywania tego samego bloku operacji dla każdego elementu z pewnej listy. Ilość wykonań tego bloku jest równa liczbie elementów tej listy. Wywoływana w pętli zmienna przyjmuje po kolei wartości każdego z elementów.

Przykłady list:

- lista liczb wpisanych ręcznie – elementy podane w nawiasach kwadratowych

```
[2,3,4,5]
```

- funkcja range – wywoła kolejno liczby naturalne zaczynając od podanej w nawiasie na pierwszym miejscu, kończąc na liczbie mniejszej o 1 od liczby na miejscu drugim

```
range(2,6)
```

Zobrazujemy działanie pętli „for” na prostym przykładzie, wymieniającym kolejno elementy z pewnej listy.

Przykład 9.

```
print('Mamy listę elementów: ', [5,6,7,8])  
for liczba in [5,6,7,8]:  
    print('element listy: ', liczba)
```

Zadania dodatkowe

- Napisz dwa programy, które wypisują liczby naturalne od 1 do 15. W pierwszym programie wykonaj pętlę for, a w drugim while.
- Zmodyfikuj powyższe zadanie, tak aby programy obliczały sumę liczb od 1 do 15.
- Za pomocą pętli for, napisz program, który oblicza silnię liczby podanej przez użytkownika.
- Oblicz sumę kwadratów liczb naturalnych z zakresu od 1 do 100.

Słowniczek

- Język interpretowany – język, który jest tłumaczony i wykonywany "w locie". Tłumaczeniem i wykonywaniem programu zajmuje się specjalny program nazwany interpreterem języka.
- Interpreter – program, który zajmuje się tłumaczeniem kodu języka programowania na język maszynowy i jego wykonywaniem.
- Zmienne – symbole zdefiniowane i nazwane przez programistę, które służą do przechowywania wartości, obliczeń na nich i odwoływanie się do wartości przez zdefiniowaną nazwę.
- Funkcje – fragmenty kodu zamknięte w określonym przez programistę symbolu, mogące przyjmować parametry oraz mogące zwracać wartości. Umożliwiają wielokrotne wywoływanie tego samego kodu, bez konieczności jego przepisywania za każdym razem, gdy zajdzie potrzeba jego wykonania.
- Typ zmiennych – rodzaj danych, który przypisany jest do zmiennej w momencie jej tworzenia.

Autorzy: Dorota Rybicka, Grzegorz Wilczek