Zmienne w Pythonie

M@я3k Pų𝐝€£kØ

Programowanie w Pythonie

Spis treści

Zmienna

Zmienna programistyczna

- Zmienna to miejsce w programie, gdzie przechowuje się informacje, wykorzystywane przez wykonujący się kod.
- Fizycznie to nazwane miejsce w pamięci, gdzie przechowywane są różne dane jak liczby, tekst, listy.

```
x = 15
```

f	15	f231	As	!@#	678	787
Sf	0x78	080	Lista[]	Null	Null	Null

Zmienna w Pythonie

- Tworzy się ona poprzez przypisanie wartości za pomocą znaku równości =.

```
x = 15
```

- Nie ma konieczności podawania typu danych.
- Python rozpoznaje automatycznie typ danych (np. **int**, **float**, **str**).

```
x = 15      #int
x = '15'    #str
```

Deklarowanie zmiennych w Pythonie

```
1  print (x)
```

```
Traceback (most recent call last):
  File "/home/main.py", line 1, in <module>
    print (x)
          ^
NameError: name 'x' is not defined
```

Próba użycia zmiennej której nie zdefiniowano (nie przypisano do niej żadnych wartości)

```
1  x = 15
2  print (x)
3  x = '15'
4  print (x)
```

```
15
15
```

Do zmiennej x przypisano liczbę całkowitą 15

Do zmiennej x przypisano tekst o wartości '15'

Reguły tworzenia nazw zmiennych

- Nazwy można budować z liter (wielkich i małych), cyfr, i znaku podkreślenia (_)

```
Zmienna_nr_04
```

- Pierwszym znakiem nazwy nie może być cyfra

```
1_zmienna = 0        #błąd
```

- Litery wielkie i małe są rozróżniane.
  - Nazwy różniące się jedynie wielkością liter są różne.

```
Nazwa ≠ NAZWA ≠ nazwa ≠ NazwA ≠ nazwA
```

- Długość nazwy nie podlega ograniczeniu
  - Dla lepszej czytelności zaleca się, aby nazwy były zwięzłe i liczyły maksymalnie 79 znaków.
- Zabronione jest użycie jako nazwy któregokolwiek z tzw. słów zastrzeżonych języka Python.

```
print = 24            #błąd
```

Umowne reguły tworzenia nazw zmiennych

- W nazwach zmiennych należy używać małych liter (a nie wielkich).
- Wielkie litery są zarezerwowane dla nazw wartości stałych, które nie mogą się zmienić w trakcie pracy programu.
- Nazwy mają mieć znaczenie i ułatwiać zrozumienie kodu programu.
- Można stosować nazwy składające się z 2 lub więcej słów.
  - Spację zastępujemy znakiem podkreślenia _. Spacja nie jest dozwolona jako część nazwy.
- Unikanie nazw przesadnie długich. Zmienne potrzebne jedynie "chwilowo" mogą być jednoliterowe (np. **i**, **j**, **k**).
- Nazwy zaczynające się od podkreślenia (_) mają specjalne znaczenia. Nie należy ich tworzyć.

Ćwiczenie 1

Które nazwy zmiennych są poprawne, a które nie? Dlaczego?

1. Antyk_123
2. print
3. exclusion@
4. euler271
5. user-name-surname
6. moja zmienna
7. Wiek_ucznia
8. _
9. x
10. X
11. imię
12. liczba#
13. s
14. stay
15. @nazwa
16. default_user_host_name
17. aBrAkAdAbRa
18. temperatura°C
19. gracz-1
20. Liczba_do_2^potęgi
21. For_do_danych_z_badania
22. Liczba+2
23. Predkosc m/s
24. __init__
25. Stop()
26. i

Ćwiczenie 2

Popraw błędne nazwy zmiennych

1. 1wiek
2. print
3. for
4. imie-nazwisko
5. moja zmienna
6. @_w_afryce
7. imię ucznia z klasy
8. data-urodzenia
9. liczba#punktów
10. suma ocen
11. liczba#
12. nadawca@poczta.onet.pl
13. www.gogle.com
14. temperatura°C
15. ileWynosiWygrana?
16. aBrAkAdAbRa
17. Liczba+2
18. gracz-1
19. najlepszy wynik
20. moje.dane
21. Predkosc m/s
22. złoty_medal
23. czyWygral?
24. M&Ms
25. %wyniki
26. zarobki_$

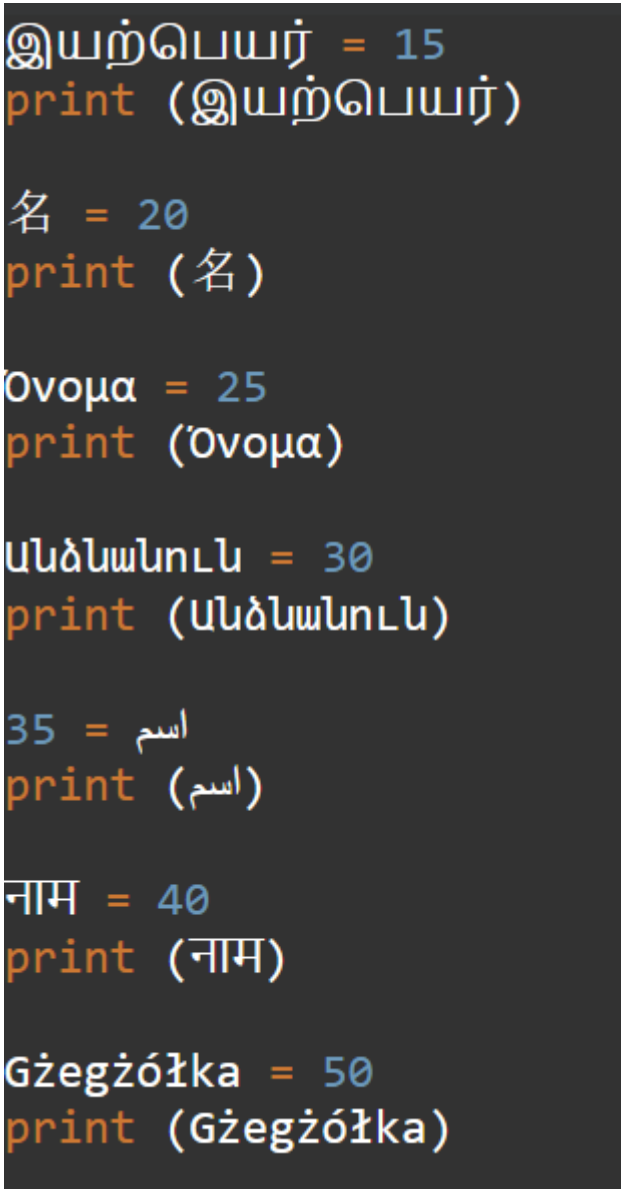
Alfabet Unicode

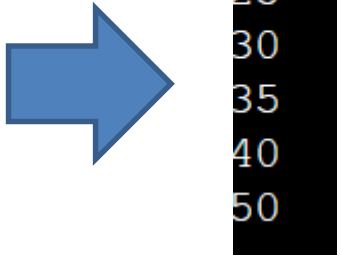
- Nazwy można utworzyć korzystając z szerokiego zestawu znaków Unicode.
- Możliwe wykorzystać znaki z rozszerzonego alfabetu łacińskiego (w tym litery polskie), alfabetów azjatyckich, afrykańskich czy amerykańskich.

- Nie jest to rekomendowane
  - Problem z odczytywaniem takich znaków przez niektóre edytory programowania.
- Lepiej ograniczyć się do liter podstawowego alfabetu łacińskiego.

Użycie znaków Unicode

```
இயற்பெயர் = 15
print (இயற்பெயர்)

名 = 20
print (名)

Όνομα = 25
print (Όνομα)

Անձնանուն = 30
print (Անձնանուն)

35 = اسم
print (اسم)

नाम = 40
print (नाम)

Gżegżółka = 50
print (Gżegżółka)
```





Ćwiczenie 3

1. Przećwicz znaki z rozszerzonego alfabetu łacińskiego.
2. Zbadaj jakie inne alfabety można użyć w Pythonie.
3. Sprawdź czy jako nazw zmiennych możesz użyć emotek dostępnych w Unicode.
4. Zorientuj się, czy da się użyć fikcyjnych alfabetów:
    1. pIQaD (Klingoński)
    2. Quenya
    3. Aurebesh

Konwencje notacji nazw

Snake_case

- Nazwy definiowane są przy użyciu małych liter, a słowa oddzielane znakiem podkreślenia _.
- W wypadku występowania kilka zmiennych o podobnym znaczeniu, na końcu dodaje się liczby (na przykład `zmienna1`, `zmienna2`).
- W programowaniu stosuje się nazwy anglojęzyczne. Wyjątkiem mogą być nazwy własne.

```
input_data
time_for_testing
user_name
data_group_key
user1, user2
```

Notacja węgierska

- Sposób zapisu nazw zmiennych oraz obiektów, polegający na poprzedzaniu właściwej nazwy małą literą (literami) określającą rodzaj tej zmiennej (obiektu). Notację węgierską wymyślił Charles Simonyi, programista z Microsoft.

```
iLiczba, i_Liczba, i_liczba

sImie, iCount, fWynik, cPlec, sUser_data_source
```

- Przykład użycia notacji węgierskiej do nazywania zmiennych w C++:
  - **s** - string (łańcuch znaków), **c** - char (jeden znak), **i** - int
- Charakterystyczne są złożenia przedrostków, zbliżone do języka węgierskiego.
  - **lpcsz** - „długi” („daleki”) wskaźnik na stały ciąg znaków zakończony bajtem zerowym
  - **pfn** - wskaźnik na funkcję
- Największą wadą tego systemu jest konieczność poprawy nazwy w każdym miejscu jej występowania przy zmianie typu zmiennej (np. z int na float).
- Notację węgierską stosuje się także w celu zaznaczenia zasięgu zmiennej, np.:
  - **g_iZmienna** - zmienna globalna, integer, **_Zmienna** - zmienna lokalna

PascalCase

- System notacji, w którym kolejne wyrazy nazwy pisane są łącznie, rozpoczynając każdy z nich wielką literą.

`TotalValue`, `MaxSpeed`, `URLName`, `UserName`.

- Odmiany
  - **PascalCase** - zawsze rozpoczyna się wielką literą.

`BackColor`, `FatalError`, `DataStream`.

  - **camelCase** - zawsze rozpoczyna się małą literą.

`backColor`, `fatalError`, `dataStream`.

Ćwiczenie 4

W jakiej konwencji została napisana nazwa danej zmiennej:

1. iLiczba = 5
2. wiek_ucznia = 12
3. TemperaturaDnia = 23
4. bZalogowany = True
5. srednia_ocen = 4.5
6. sImie = "Jan "
7. fCena = 19.99
8. dictUzytkownik = {"imie": "Marek", "wiek": 30}
9. liczba_punktow = 100
10. lstWyniki = [1, 2, 3]
11. AdresEmail = "test@example.com"
12. PunktGracza = 50
13. iNumer = 14
14. imie_uzytkownika = "Marek"
15. czy_zalogowany = True
16. nazwa_pliku ="dane.txt"
17. DaneUzytkownika
18. sHasloUzytkownika
19. ilosc_prob
20. HistoriaZakupow

Operacje na zmiennych

Definiowanie i inicjalizacja zmiennych

- W momencie definiowania zmiennej (podanie nazwy) inicjalizujemy ją jakąś wartością.

```
x = 12 # x typu całkowitego
print(x)
x = 'ala ma kota' # x string
print(x)
```

Wczytywanie zmiennych

- Zmienną możemy wczytać z klawiatury. Służy do tego polecenie `input`.

```
input (x)
print (x)
```

- Polecenie `input` wczytuje domyślnie dane jako tekst. Należy dokonać konwersji na określony typ.

```
x = int(input ())
print(x)        #int
```

Zmienna pusta

- Aby zainicjować zmienną wystarczy podać jej nazwę. Można utworzyć zmienną bez ustalonej wartości lub przypisać wartość.

```
x = None
print(type(x))
                # <class 'NoneType'>
x = 10
print(type(x)) # <class 'int'>
```

Wartość None

- Istnieje specjalna wartość `None`, która symbolizuje brak wartości.
- Często używana jako wartość nowej zmiennej, gdy nie jeszcze przypisano jej żadnej wartości.
- None używana jest także, by pozbyć się wartości zmiennej.

```
name = None
print(name)   # None


name = „Uzytkownik"
print(name)   # Marcin


name = None
print(name)   # None
```

- Aby sprawdzić, czy zmienna zawiera wartość None standardowo używa się `is None` zamiast `== None`.
- Słowo kluczowe `is` sprawdza, czy po obu stronach znajduje się dokładnie ten sam obiekt.

```
name = None
print(name is None)
                              #True

name = " Uzytkownik"
print(name is None)
                             #False

name = None
print(name is None)
                              #True
```

Ćwiczenie 5

- Uzupełnij brakujące fragmenty:

```
name = _____
surname = „Kowalski"
print(name is None)  # ____
print(name)  # None
print(surname)  # Kowalski
name = surname
surname = „Nowak"
print(name is None)  # ____
print(name)  # _______
print(surname)  # _________
```

Ustalenie typu zmiennej

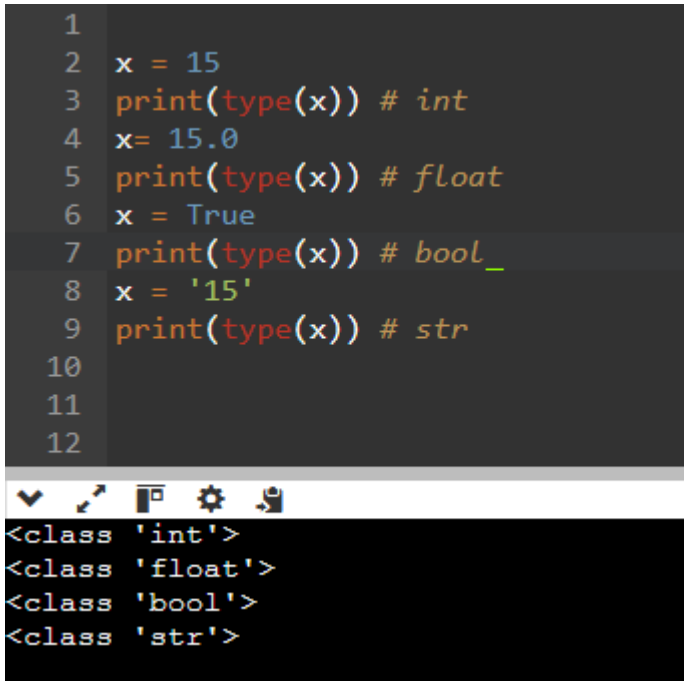
- Instrukcja `type(zmienna)`
- Instrukcja `type(zmienna)` zwraca typ zmiennej podanej jako argument obiektu `type`.

```
x = 22
print(type(x)) # <class 'int'>
```

Zmiana typu zmiennej

- Nie ma konieczności definiowania typu danych. Zmienna w Pythonie ma typ zależny od zawartości.
- Typ zmiennej jest dynamiczny tzn. że w momencie zmiany zawartości na inny rodzaj, typ zmiennej ulega automatycznie zmianie.

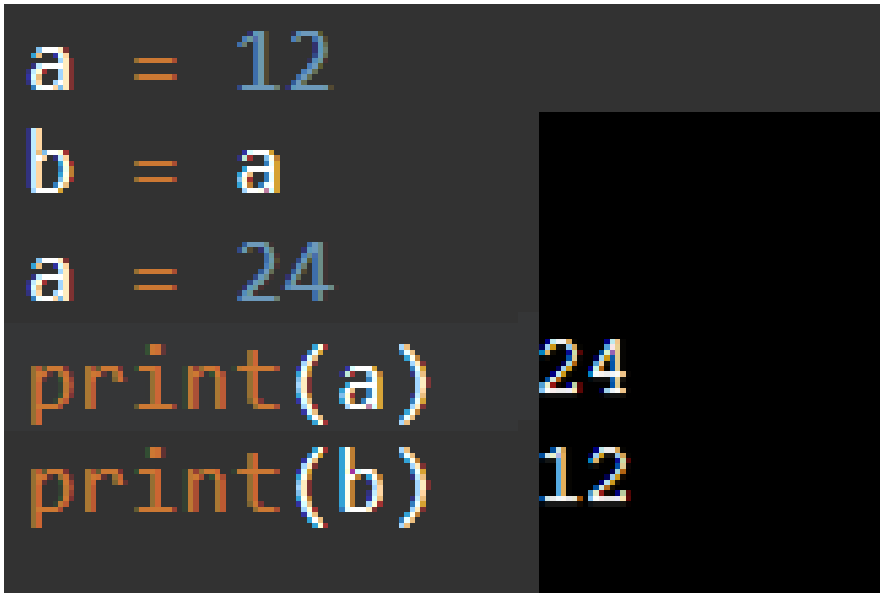
```
x = 15
print(type(x))     # int
x= 15.0
print(type(x))     # float
x = True
print(type(x))     # bool
x = '15'
print(type(x))     # str
```



Przypisywanie zmiennych

- Sytuacja, gdy do zmiennej zostanie przypisana wartość innej zmiennej.
  - Potencjalne ryzyko pomyłki w programie.
- Zmienne wskazują na wartości, a nie na inne zmienne.

```
a = 12
b = a
a = 24
print(a)   # 24


print(b)   # 12,
# b wskazuje na wartość 12, a nie na zmienną a
```



Kasowanie zmiennej

- Do skasowania zmienną używana jest instrukcja „del”. Jeśli po skasowaniu zmiennej będziemy chcieli wyświetlić jej wartość, pojawi się błąd.

```
imie = 'Marek'
print(imie)


del imie
print(imie)
```

```
imię = 'Marek'
print(imię)

del imię
print(imię)
```

```
Marek
Traceback (most recent call last):
  File "/home/main.py", line 6, in <module>
    print(imię)
          ^^^^
NameError: name 'imię' is not defined
```

Kasowanie zmiennej

- Do skasowania zmienną używana jest instrukcja „del”. Jeśli po skasowaniu zmiennej będziemy chcieli wyświetlić jej wartość, pojawi się błąd.

```
imie = 'Marek'
print(imie)


del imie
print(imie)
```

```
imię = 'Marek'
print(imię)

del imię
print(imię)
```

```
Marek
Traceback (most recent call last):
  File "/home/main.py", line 6, in <module>
    print(imię)
          ^^^^
NameError: name 'imię' is not defined
```

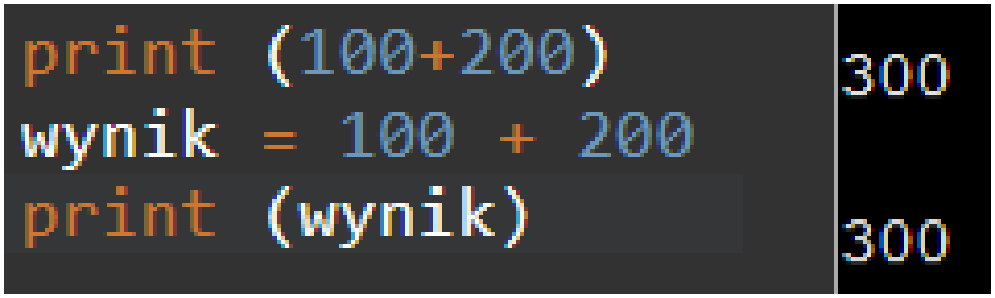
Przypisywanie wyrażeń do zmiennej

- Zmienna może być użyta wewnątrz wyrażenia w miejsce wartości. Wówczas wartość wzięta ze zmiennej będzie użyta przy obliczaniu wartości wyrażenia.

```
                        # 100+200
print (100+200)   #wynik 300
```

- Wyrażenie, w którym do zmiennej o nazwie `wynik`" przypisywana jest suma dodawania liczb 100 i 200. Zmienna wynik jest wyświetlana poleceniem `print ()`.

```
wynik = 100 + 200       # 100+200
print (wynik)           #wynik 300
```



Ćwiczenie 6

1. Napisz program wczytujący 2 liczby całkowite i wyświetlający wynik czterech podstawowych działań arytmetycznych.
2. Napisz program, który pobierze od użytkownika dane personalne: imię, nazwisko, miejscowość, ulicę, numer domu, numer mieszkania a potem wyświetli na ekranie je według poniższego szablonu:

```
Jan Kowalski
ul. Nowakowa 1/1
miejscowość: Warszawa
```

3. Napisz program, który wczyta od użytkownika jego wagę oraz wzrost, a następnie obliczy i wyświetli na ekranie BMI według wzoru: BMI = waga/wzrost2
4. Napisz program, przeliczający wartość kwoty w złotówkach na dolary. Kurs dolara poszukaj w aktualnych danych NBP.

Powtórzenie

1. Co to jest zmienna?
2. Jak tworzy się zmienną w Pythonie
3. Jak zadeklarować typ zmiennej w Pythonie?
4. Czy można wyświetlić zmienną, do której nie przypisano żadnej wartości?
5. Jakie reguły musi spełnić nazwa zmiennej w Pythonie?
6. Omów następujące konwencje tworzenia nazw
   a) Snake_case
   b) Notacja węgierska
   c) PascalCase
7. Czy można utworzyć pustą zmienną?
8. Do czego słuzy wartość None?
9. Podaj instrukcję, która pozwala ustalić typ zmiennej.
10. Czy zmienna może zmienić swój typ? W jakiej sytuacji?
11. Jak skasować zmienną?