

Zmienne, wyrażenia i instrukcje

Rozdział 2



Python dla wszystkich
www.py4e.pl



Stałe

- Stałe wartości takie jak liczby, litery i ciągi znaków nazywane są "stałymi", ponieważ ich wartości się nie zmieniają
- Stałe liczbowe wyglądają normalnie
- Zapis stałych ciągów wykorzystuje apostrofy (') lub cudzysłowy (")

```
>>> print(123)
123
>>> print(98.6)
98.6
>>> print('Witaj świecie')
Witaj świecie
```

Słowa zastrzeżone

Nie wolno używać słów zastrzeżonych jako nazw zmiennych/ identyfikatorów

False	class	return	is	finally
None	if	for	lambda	continue
True	def	from	while	nonlocal
and	del	global	not	with
as	elif	try	or	yield
assert	else	import	pass	
break	except	in	raise	

Zmienne

- Zmienna to nazwane miejsce w pamięci komputera, gdzie programista może zapisać i odczytać dane, używając "nazwy" tej zmiennej.
- Programiści sami wybierają nazwy zmiennych
- Można zmienić zawartość zmiennej za pomocą dalszych instrukcji

```
x = 12.2
y = 14
```

x 12.2

y 14

Zmienne

- Zmienna to nazwane miejsce w pamięci komputera, gdzie programista może zapisać i odczytać dane, używając "nazwy" tej zmiennej.
- Programiści sami wybierają nazwy zmiennych
- Można zmienić zawartość zmiennej za pomocą dalszych instrukcji

```
x = 12.2
y = 14
x = 100
```

x	12.2 100
y	14

Mnemoniczne nazwy zmiennych

- Ponieważ to my, programiści, wybieramy nazwy zmiennych, mamy "najlepsze praktyki"
- Nazywamy zmienne tak, aby łatwiej zapamiętać, co w nich przechowujemy ("mnemonika" = "pomoc w zapamiętywaniu")
- Początkujący mogą się mylić, ponieważ dobre nazwy zmiennych często "brzmia" tak dobrze, że wydają się słowami kluczowymi

<http://en.wikipedia.org/wiki/Mnemonic>

Zasady nazywania zmiennych Pythona

- Muszą zaczynać się literą lub znakiem podkreślenia _
- Mogą zawierać litery, cyfry i znaki podkreślenia
- Wielkość znaków jest ważna

Dobre:	spam	eggs	spam23	_speed
Złe:	23spam	#sign	var.12	
Różne:	spam	Spam	SPAM	

```
x1q3z9ocd = 35.0
x1q3z9afd = 12.50
x1q3p9afd = x1q3z9ocd * x1q3z9afd
print(x1q3p9afd)
```

Co robi ten
kawałek kodu?

```
x1q3z9ocd = 35.0
x1q3z9afd = 12.50
x1q3p9afd = x1q3z9ocd * x1q3z9afd
print(x1q3p9afd)
```

```
a = 35.0
b = 12.50
c = a * b
print(c)
```

```
x1q3z9ocd = 35.0
x1q3z9afd = 12.50
x1q3p9afd = x1q3z9ocd * x1q3z9afd
print(x1q3p9afd)
```

```
a = 35.0
b = 12.50
c = a * b
print(c)
```

Co robią te kawałki
kodu?

Co robią te kawałki
kodu?

```
godziny = 35.0
stawka = 12.50
wynagrodzenie = godziny * stawka
print(wynagrodzenie)
```

Zdania lub linie

```
x = 2      ← Instrukcja przypisania
x = x + 2  ← Przypisanie wyrażenia
print(x)   ← Instrukcja "print"
```

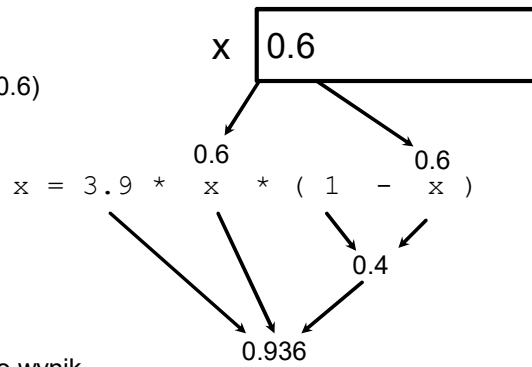
Zmienna Operator Stała Funkcja

Instrukcje przypisania

- Przypisujemy wartość do zmiennej instrukcją przypisania (=)
- Instrukcja przypisania składa się z wyrażenia po prawej stronie oraz ze zmiennej, która przyjmie wynik

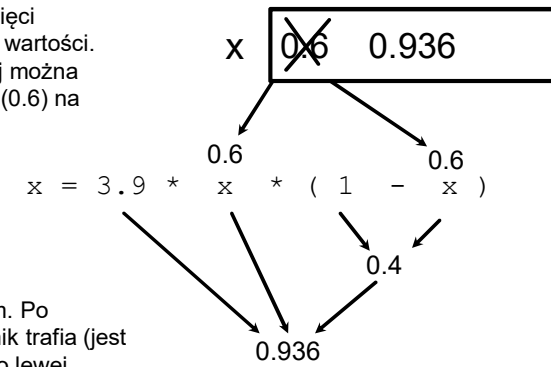
```
x = 3.9 * x * ( 1 - x )
```

Zmienna to lokalizacja w pamięci używana do przechowywania wartości (0.6)



Prawa strona wyrażenia.
Po ewaluacji wyrażenia jego wynik trafia (jest przypisywany) do x.

Zmienna to lokalizacja w pamięci używana do przechowywania wartości. Wartość zapisaną w zmiennej można uaktualnić, zamieniając starą (0.6) na nową (0.936).



Prawa strona jest wyrażeniem. Po ewaluacji wyrażenia jego wynik trafia (jest przypisywany) do zmiennej po lewej stronie (tj. x).

Wyrażenia...

Wyrażenia liczbowe

- Ponieważ na klawiaturze brakuje symboli matematycznych, używamy "kodo-mowy", aby zapisać klasyczne działania
- Gwiazdka to mnożenie
- Potęgowanie wygląda inaczej niż w matematyce

Operator	Operacja
+	Dodawanie
-	Odejmowanie
*	Mnożenie
/	Dzielenie
**	Potęga
%	Reszta

Wyrażenia liczbowe

```
>>> xx = 2
>>> xx = xx + 2
>>> print(xx)
4
>>> yy = 440 * 12
>>> print(yy)
5280
>>> zz = yy / 1000
>>> print(zz)
5.28
```

```
>>> jj = 23
>>> kk = jj % 5
>>> print(kk)
3
>>> print(4 ** 3)
64
```

```

      4 R 3
5 | 23
   20
   --
    3

```

Operator	Operacja
+	Dodawanie
-	Odejmowanie
*	Mnożenie
/	Dzielenie
**	Potęga
%	Reszta

Kolejność działań

- Kiedy wykonujemy wiele operacji, Python musi wiedzieć, w jakiej kolejności je wykonać
- Nazywamy to “kolejnością operatorów”
- Który operator “jest w kolejce” przed pozostałymi?

$x = 1 + 2 * 3 - 4 / 5 ** 6$

Kolejność operatorów

Od najwyższego do najniższego priorytetu:

- Zawsze zaczynamy od nawiasów
- Potęgowanie
- Mnożenie, dzielenie i reszta
- Dodawanie i odejmowanie
- Od lewej do prawej

Nawiasy
Potęga
Mnożenie
Dodawanie
Od lewej do
prawej



```
>>> x = 1 + 2 ** 3 / 4 * 5
>>> print(x)
11.0
>>>
```

Nawiasy
Potęga
Mnożenie
Dodawanie
Od lewej do
prawej



```


1 + 2 ** 3 / 4 * 5
    ↓
1 + 8 / 4 * 5
    ↓
1 + 2 * 5
    ↓
1 + 10
    ↓
11

```

Kolejność operatorów

- Zapamiętaj zasady – z góry do dołu
- Pisząc kod, używaj nawiasów
- Pisząc kod, nie twórz skomplikowanych wyrażeń matematycznych – niech będzie łatwo je zrozumieć
- Długie serie działań dziel na krótsze, bardziej zrozumiałe

Nawiasy
Potęga
Mnożenie
Dodawanie
Od lewej do
prawej



Co oznacza “typ”?

- W Pythonie zmienne, literały i stałe mają “typy”
- Python potrafi odróżnić liczbę całkowitą od ciągu znaków
- Na przykład “+” oznacza “dodawanie” w przypadku liczb, ale “konkatenację” w przypadku ciągu znaków

```
>>> ddd = 1 + 4
>>> print(ddd)
5
>>> eee = 'hej ' + 'tam'
>>> print(eee)
hej tam
```

konkatenacja = łączenie

Typ jest ważny

- Python zna “typ” każdego obiektu
- Niektóre operacje są zabronione
- Nie możesz “dodać 1” do ciągu znaków
- Możemy zapytać Pythona o typ jakiegoś obiektu, używając funkcji type()

```
>>> eee = 'hej ' + 'tam'
>>> eee = eee + 1
Traceback (most recent call last):
  File "<stdin>", line 1, in
<module>TypeError: Can't convert
'int' object to str implicitly
>>> type(eee)
<class'str'>
>>> type('hej')
<class'str'>
>>> type(1)
<class'int'>
>>>
```

Różne typy liczb

- Liczby mają dwa główne typy
 - Liczby całkowite (int): -14, -2, 0, 1, 100, 401233
 - Liczby zmiennoprzecinkowe (float): -2.5, 0.0, 98.6, 14.0
- Są też inne typy liczb – odmiany liczb całkowitych i zmiennoprzecinkowych

```
>>> xx = 1
>>> type(xx)
<class 'int'>
>>> temp = 98.6
>>> type(temp)
<class 'float'>
>>> type(1)
<class 'int'>
>>> type(1.0)
<class 'float'>
>>>
```

Konwersje typów

- Jeśli w wyrażeniu znajdzie się liczba zmiennoprzecinkowa i całkowita, Python automatycznie skonwertuje całkowitą na zmiennoprzecinkową
- Możesz też kontrolować konwersje wbudowanymi funkcjami `int()` i `float()`

```
>>> print(float(99) + 100)
199.0
>>> i = 42
>>> type(i)
<class 'int'>
>>> f = float(i)
>>> print(f)
42.0
>>> type(f)
<class 'float'>
>>>
```

Dzielenie liczb całkowitych

Wynikiem dzielenia liczb całkowitych jest liczba zmiennoprzecinkowa

```
>>> print(10 / 2)
5.0
>>> print(9 / 2)
4.5
>>> print(99 / 100)
0.99
>>> print(10.0 / 2.0)
5.0
>>> print(99.0 / 100.0)
0.99
```

W Pythonie 2.x było inaczej

Konwersje ciągów znaków

- Możesz też użyć `int()` i `float()` do konwersji pomiędzy ciągiem a liczbą całkowitą
- Otrzymasz błąd, jeśli ciąg nie zawiera cyfr

```
>>> sval = '123'
>>> type(sval)
<class 'str'>
>>> print(sval + 1)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: Can't convert 'int' object
to str implicitly
>>> ival = int(sval)
>>> type(ival)
<class 'int'>
>>> print(ival + 1)
124
>>> nsv = 'hej bob'
>>> niv = int(nsv)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: invalid literal for int()
with base 10: 'x'
```

Dane od użytkownika

- Dzięki funkcji `input()` możemy nakazać Pythonowi, aby zaczekał na wprowadzenie danych przez użytkownika
- Funkcja `input()` zwraca ciąg znaków

```
nam = input('Kim jesteś? ')
print('Witaj', nam)
```

Kim jesteś? Chuck
Witaj Chuck

Konwersja danych od użytkownika



- Jeśli chcemy odczytać liczbę wprowadzoną przez użytkownika, musimy skonwertować ciąg znaków na liczbę za pomocą funkcji konwersji.

```
inp = input('Europejskie piętro? ')
usf = int(inp) + 1
print('Amerykańskie piętro:', usf)
```

- Błędami we wprowadzaniu danych zajmiemy się później.

Europejskie piętro? 0
Amerykańskie piętro: 1

```
# Pobierz nazwę pliku i otwórz go
name = input('Podaj nazwę pliku: ')
handle = open(name, 'r')

# Zlicz częstość słów
counts = dict()
for line in handle:
    line = line.lower()
    words = line.split()
    for word in words:
        counts[word] = counts.get(word, 0) + 1

# Znajdź najczęstsze słowo
bigcount = None
bigword = None
for word, count in list(counts.items()):
    if bigcount is None or count > bigcount:
        bigword = word
        bigcount = count

# Wszystko gotowe
print(bigword, bigcount)
```

Komentarze w Pythonie

- Python ignoruje wszystko po znaku #
- Po co komentować?
 - Opisyj, co zrobi fragment kodu
 - Dokumentuj, kto jest autorem kodu, i dodawaj pomocne informacje
 - Wyłącz – może na moment – jedną linię kodu

Podsumowanie

- Typy
- Słowa zastrzeżone
- Zmienne (mnemonika)
- Operatory
- Kolejność operatorów
- Dzielenie liczb całkowitych
- Konwersja typów
- Dane od użytkownika
- Komentarze (1)

Ćwiczenie

Napisz program, który wyświetli użytkownikowi pytanie o liczbę godzin pracy i stawkę za godzinę w celu obliczenia wynagrodzenia.

Podaj liczbę godzin: 35

Podaj stawkę godzinową: 2.75

Wynagrodzenie: 96.25



Podziękowania dla współpracowników



Copyright slajdów 2010 - Charles R. Severance
(www.dr-chuck.com) University of Michigan School of Information
i open.umich.edu dostępne na licencji Creative Commons
Attribution 4.0. Aby zachować zgodność z wymaganiami licencji
należy pozostawić ten slajd na końcu każdej kopii tego
dokumentu. Po dokonaniu zmian, przy ponownej publikacji tych
materiałów można dodać swoje nazwisko i nazwę organizacji do
listy współpracowników

Autorstwo pierwszej wersji: Charles Severance,
University of Michigan School of Information

Polska wersja powstała z inicjatywy Wydziału Matematyki
i Informatyki Uniwersytetu im. Adama Mickiewicza w Poznaniu

Tłumaczenie: Agata i Krzysztof Wierzbiccy, EnglishT.eu

... wstaw tu nowych współpracowników i tłumaczy