

Słowniki w Pythonie

Rozdział 9



Python dla wszystkich
www.py4e.pl



Co nie jest “kolekcją”?

Większość z naszych zmiennych zawiera jedną wartość – kiedy przypisujemy nową wartość do zmiennej, stara wartość jest nadpisywana

```
$ python3
>>> x = 2
>>> x = 4
>>> print(x)
4
```

Co to jest “kolekcja”?



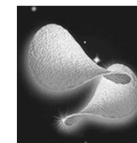
- Kolekcja to coś fajnego, bo możemy do niej zapakować więcej niż jedną wartość, jak do walizki
- Mamy wiele wartości w pojedynczej “zmiennej”
- Możemy tak zrobić, bo mamy więcej niż jedno miejsce “wewnątrz” zmiennej
- Mamy też sposób na odnajdywanie w zmiennej tych miejsc



O dwóch takich... kolekcjach

- Lista

- liniowa kolekcja uporządkowanych wartości

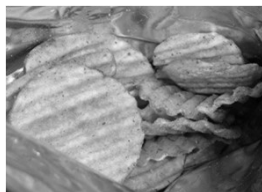


- Słownik

- “walizka” z wartościami, z których każda ma własną etykietę



Słowniki



https://pl.wikipedia.org/wiki/Tabela_asocjacyjna

Słowniki



- Słowniki to najpotężniejsza forma kolekcji danych w Pythonie
- Słowniki pozwalają nam wykonywać w Pythonie szybkie operacje, jak w bazach danych
- Słowniki w innych językach programowania mają różne nazwy:
 - Tablice asocjacyjne - Perl / PHP
 - Property, Map lub HashMap - Java
 - Property Bag - C# / .Net

Słowniki

- Listy indeksują swoje elementy w oparciu o ich kolejność
- Słowniki często są jak torby – nieuporządkowane
- Dlatego indeksujemy wszystko, co wrzucamy do słownika za pomocą “etykiety do wyszukiwania”

```
>>> purse = dict()
>>> purse['pieniądze'] = 12
>>> purse['słodkizy'] = 3
>>> purse['chusteczki'] = 75
>>> print(purse)
{'pieniądze': 12, 'słodkizy': 3, 'chusteczki': 75}
>>> print(purse['słodkizy'])
3
>>> purse['słodkizy'] = purse['słodkizy']
+ 2
>>> print(purse)
{'pieniądze': 12, 'słodkizy': 5, 'chusteczki': 75}
```

Porównanie list i słowników

Słowniki są podobne do list, za wyjątkiem tego, że używają kluczy zamiast liczb, aby wyszukiwać wartości

```
>>> lst = list()
>>> lst.append(21)
>>> lst.append(183)
>>> print(lst)
[21, 183]
>>> lst[0] = 23
>>> print(lst)
[23, 183]
```

```
>>> ddd = dict()
>>> ddd['wiek'] = 21
>>> ddd['kurs'] = 182
>>> print(ddd)
{'wiek': 21, 'kurs': 182}
>>> ddd['wiek'] = 23
>>> print(ddd)
{'wiek': 23, 'kurs': 182}
```

```
>>> lst = list()
>>> lst.append(21)
>>> lst.append(183)
>>> print(lst)
[21, 183]
>>> lst[0] = 23
>>> print(lst)
[23, 183]
```

Lista		
Klucz	Wartość	
[0]	21	lst
[1]	183	

```
>>> ddd = dict()
>>> ddd['wiek'] = 21
>>> ddd['kurs'] = 182
>>> print(ddd)
{'wiek': 21, 'kurs': 182}
>>> ddd['wiek'] = 23
>>> print(ddd)
{'wiek': 23, 'kurs': 182}
```

Słownik		
Klucz	Wartość	
['wiek']	21	ddd
['kurs']	182	

Literały (stałe) słownikowe

- Literały słownikowe zapisywane w nawiasach klamrowych składają się z listy będącej parą klucz : wartość
- Możesz stworzyć pusty słownik za pomocą pustych nawiasów klamrowych

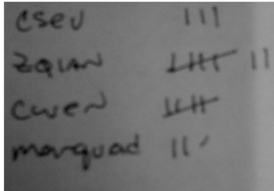
```
>>> jjj = { 'chuck' : 1 , 'fred' : 42, 'jan': 100}
>>> print(jjj)
{'chuck': 1, 'fred': 42, 'jan': 100}
>>> ooo = { }
>>> print(ooo)
{}
>>>
```

Najpopularniejsza nazwa
użytkownika?

Najpopularniejsza nazwa
użytkownika?

marquard	cwen	cwen
zhen	marquard	zhen
csev	zhen	csev
zhen	csev	marquard
		zhen

Najpopularniejsza nazwa użytkownika?

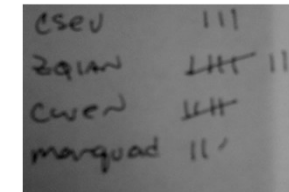
marquard	cwen	cwen
zhen		zhen
csev		csev
zhen	csev	marquard
		zhen

Wiele liczników w słowniku

Typowym zastosowaniem słowników jest zliczanie, jak często “widzimy” jakąś wartość

```
>>> ccc = dict()
>>> ccc['csev'] = 1
>>> ccc['cwen'] = 1
>>> print(ccc)
{'csev': 1, 'cwen': 1}
>>> ccc['cwen'] = ccc['cwen'] + 1
>>> print(ccc)
{'csev': 1, 'cwen': 2}
```

Klucz Wartość



Traceback w słownikach

- Odwołanie do klucza, którego nie ma w słowniku, powoduje błąd
- Możemy wykorzystać operator `in`, żeby sprawdzić, czy klucz jest w słowniku

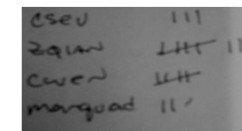
```
>>> ccc = dict()
>>> print(ccc['csev'])
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
KeyError: 'csev'
>>> 'csev' in ccc
False
```

Pierwsze spotkanie z nową nazwą

Gdy napotkamy nową nazwę, musimy dodać nowy element do słownika, a jeśli spotkaliśmy tę nazwę już raz czy dwa, to po prostu zwiększamy o jeden licznik w słowniku przypisany do tej nazwy

```
counts = dict()
names = ['csev', 'cwen', 'csev', 'zqian', 'cwen']
for name in names:
    if name not in counts:
        counts[name] = 1
    else:
        counts[name] = counts[name] + 1
print(counts)
```

{'csev': 2, 'cwen': 2, 'zqian': 1}



Metoda get() w słownikach

Schemat sprawdzania, czy klucz jest już w słowniku, i przypisywanie domyślnej wartości, jeśli klucza jeszcze nie ma, jest tak powszechny, że istnieje metoda `get()`, która robi to za nas

Domyślna wartość, jeśli klucz nie istnieje (i nie pojawia się Traceback)

```
if name in counts:
    x = counts[name]
else:
    x = 0

x = counts.get(name, 0)

{'csev': 2, 'cwen': 2, 'zqian': 1}
```

Prostsze zliczanie z get()

Możemy skorzystać z `get()` i podać wartość domyślną zero, kiedy klucza jeszcze nie ma w słowniku – a potem po prostu dodać jeden

```
counts = dict()
names = ['csev', 'cwen', 'csev', 'zqian', 'cwen']
for name in names:
    counts[name] = counts.get(name, 0) + 1
print(counts)
```

Domyślnie

{'csev': 2, 'cwen': 2, 'zqian': 1}

Prostsze zliczanie z get()

```
counts = dict()
names = ['csev', 'cwen', 'csev', 'zqian', 'cwen']
for name in names:
    counts[name] = counts.get(name, 0) + 1
print(counts)
```



<https://www.youtube.com/watch?v=EHJ9uYx5L58>

Zliczanie słów w tekście

Pisanie programów (lub programowanie) jest bardzo twórczą i satysfakcjonującą aktywnością. Możesz pisać programy z wielu powodów: od zarabiania na życie, przez rozwiązywanie trudnych zagadnień analizy danych, po zabawę i pomaganie komuś w rozwiązaniu jakiegoś problemu. Poniższa książka zakłada, że każdy powinien wiedzieć, jak się programuje, więc gdy już dowiesz się, jak programować, to zorientujesz się, co chcesz zrobić ze swoją nową umiejętnością.

W codziennym życiu jesteśmy otoczeni przez komputery, poczynając od laptopów po smartfony. Możemy myśleć o tych komputerach jak o naszych "osobistych asystentach", którzy w naszym imieniu mogą zająć się wieloma sprawami. Sprzęt we współczesnych komputerach jest zasadniczo zbudowany tak, aby nieustannie zadawać nam pytanie "Co mam teraz zrobić?".

Nasze komputery są szybkie i mają ogromne zasoby pamięci – ten fakt byłby dla nas bardzo pomocny, gdybyśmy tylko znali język do porozumiewania się i wyjaśniania komputerowi, co chcemy, aby dla nas "teraz zrobił". Gdybyśmy znali taki język, to moglibyśmy powiedzieć komputerowi, by wykonał za nas pewne powtarzające się czynności. Co ciekawe, to, co komputery potrafią najlepiej, to często rzeczy, które my – ludzie – uważamy za nudne i nużące.

```
$ python3 wordcount.py
Wpisz linię tekstu:
the clown ran after the car and the car ran into the tent
and the tent fell down on the clown and the car
```

```
Słowa: ['the', 'clown', 'ran', 'after', 'the', 'car',
'and', 'the', 'car', 'ran', 'into', 'the', 'tent', 'and',
'the', 'tent', 'fell', 'down', 'on', 'the', 'clown',
'and', 'the', 'car']
Zliczam...
```

```
Liczba: {'the': 7, 'clown': 2, 'ran': 2, 'after': 1,
'car': 3, 'and': 3, 'into': 1, 'tent': 2, 'fell': 1,
'down': 1, 'on': 1}
```



<https://www.flickr.com/photos/71502646@N00/2526007974/>

Schemat zliczania

```
counts = dict()
print('Wpisz linię tekstu:')
line = input('')

words = line.split()

print('Słowa:', words)

print('Zliczam...')
for word in words:
    counts[word] = counts.get(word, 0) + 1
print('Liczba:', counts)
```

Ogólnie schemat zliczania słów w linii polega na tym, żeby podzielić linię na słowa, następnie przejść przez słowa pętlą i użyć słownika, aby śledzić, ile razy pojawiło się każde słowo.

```
counts = dict()
line = input('Wpisz linię tekstu: ')
words = line.split()

print('Słowa:', words)
print('Zliczam...')

for word in words:
    counts[word] = counts.get(word,0) + 1
print('Liczba:', counts)
```



```
$ python3 wordcount.py
Wpisz linię tekstu: the clown ran after the
car and the car ran into the tent and the tent
fell down on the clown and the car
```

```
Słowa: ['the', 'clown', 'ran', 'after', 'the', 'car',
'and', 'the', 'car', 'ran', 'into', 'the', 'tent', 'and',
'the', 'tent', 'fell', 'down', 'on', 'the', 'clown',
'and', 'the', 'car']
Zliczam...
```

```
Liczba {'the': 7, 'clown': 2, 'ran': 2, 'after': 1,
'car': 3, 'and': 3, 'into': 1, 'tent': 2, 'fell': 1,
'down': 1, 'on': 1}
```

Pętle określone i słowniki

Słowniki w Pythonie od wersji 3.6 są uporządkowane zgodnie z kolejnością wstawiania par klucz-wartość. Możemy napisać pętlę `for`, która przejdzie po wszystkich elementach słownika – w rzeczywistości przechodzi po wszystkich kluczach w słowniku i przeszukuje wartości.

```
>>> counts = { 'chuck' : 1 , 'fred' : 42, 'jan': 100}
>>> for key in counts:
...     print(key, counts[key])
...
chuck 1
fred 42
jan 100
>>>
```

Pobieranie list kluczy i wartości

Możesz otrzymać listę kluczy, wartości lub elementów (obu) ze słownika

```
>>> jjj = { 'chuck' : 1 , 'fred' : 42, 'jan': 100}
>>> print(list(jjj))
['chuck', 'fred', 'jan']
>>> print(jjj.keys())
['chuck', 'fred', 'jan']
>>> print(jjj.values())
[1, 42, 100]
>>> print(jjj.items())
[('chuck', 1), ('fred', 42), ('jan', 100)]
>>>
```

Co to jest "krotka"? – dowiesz się niedługo...

Bonus: Dwie zmienne sterujące!

- Przechodzimy pętlą po parach klucz - wartość w słowniku, używając *dwóch* zmiennych sterujących

```
jjj = { 'chuck' : 1 , 'fred' : 42, 'jan': 100}
for aaa,bbb in jjj.items() :
    print(aaa, bbb)
```

```
chuck 1
fred 42
jan 100
```

	aaa	bbb
[chuck]	1	
[fred]	42	
[jan]	100	

- W każdej iteracji pierwszą zmienną jest klucz a drugą jest wartość przypisane do tego klucza

```
name = input('Podaj nazwę pliku: ')
handle = open(name)

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

bigcount = None
bigword = None
for word,count in counts.items():
    if bigcount is None or count > bigcount:
        bigword = word
        bigcount = count

print(bigword, bigcount)
```

```
$ python3 words.py
Nazwa pliku: words.txt
w 5
```

```
$ python3 words.py
Nazwa pliku: clown.txt
the 7
```

Używanie dwóch pętli zagnieżdżonych

Podsumowanie

- Czym jest kolekcja?
- Listy kontra słowniki
- Stałe w słownikach
- Najpopularniejsze słowa
- Używanie metody get()
- Haszowanie i brak porządku
- Pisanie pętli w słownikach
- Krotki: wstęp
- Sortowanie słowników



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