# Listy składane
# Zbiory
# Słowniki

Uniwersytet Ekonomiczny w Poznaniu

23 lutego 2016

# Listy składane

- tworzenie listy pętlą for

```
przestepne=[]
for rok in range(1900,1940):
    if (rok % 4 ==0 and
    rok % 100 !=0) or (rok % 400 == 0):
        przestepne.append(rok)
```

- lista składana - `[element for element in iteracja]`

```
lata = [y for y in range(1900,1940)]
```

- lista składana z warunkiem

```
przestepne = [rok for rok in range(1900,1940)
    if (rok % 4 ==0 and rok % 100 !=0)
    or (rok % 400 == 0)]
```

# Przykłady list składanych

W matematyce często opisujemy zbiory w taki sposób:

$$S = \{x^2 | x \in \{0, ..., 9\}\}$$

$$V = \{2^i | i \in < 0, 13) \wedge i \in N\}$$

$$M = \{x | x \in S \wedge x \ parzyste\}$$

W Pythonie w naturalny sposób można wyrazić to tak:

```
S = [x**2 for x in range(10)]                    1
V = [2**i for i in range(13)]                    2
M = [x for x in S if x % 2 == 0]                 3
```

# Przykłady list składanych

Liczby pierwsze to takie,... które nie są niepierwsze (nie są wielokrotnością innej liczby niż 1)

```
npr=[j for i in range(2,8) for j in range(i*2,50,i)]
pr = [x for x in range(2, 50) if x not in npr]
```

# Przykłady list składanych

Przykład ze słowami:

```
words = 'The quick brown fox jumps'.split()                    1
stuff=[[w.upper(),w.lower(),len(w)] for w in words]            2
```

# Zbiór - set

- obsługuje `in`, `len`, iterację
- zbiory modyfikowalne - `set`, niemutowalne - `frozenset`
- dodane do zbioru mogą być tylko typy które dają hash tzn. `float`,`frozenset`,`int`,`str`,`tuple`
- typy mutowalne nie mogą być dodawane do zbioru bo ich hash byłby zmienny (`dict`,`list`,`set`)

```
S = {7,''welon'',(''a'',1),frozenset({3,6,8})}   1
```

- pusty zbiór - tylko przez funkcję `set()`
- zbiory nie przechowują duplikatów. Trick na usunięcie duplikatów z listy

```
x=list(set(x))   1
```

# Operatory i funkcje dla zbiorów

- suma

```
set(''pecan'') | set(''pie'')                    1
```

- iloczyn

```
set(''pecan'') & set(''pie'')                    1
```

- różnica

```
set(''pecan'') - set(''pie'')                    1
```

- różnica symetryczna

```
set(''pecan'') ^ set(''pie'')                    1
```

# Operatory i funkcje dla zbiorów

- powyższe operatory i funkcje aktualizujące im odpowiadające tworzą kopie. Są wersje aktualizujące lewy argument
- funkcje `isdisjoint`, `pop`, `remove`, `update`, `add`, `clear`, `copy`
- zbiory można tworzyć przez zbiór składany

# Frozenset

- tworzenie - tylko przez funkcję `frozenset()`
- obsługuje z API zbioru tylko funkcje nie zmieniające zbioru

# Słownik

- Słownik albo mapa (dictionary, map) to kolekcja par klucz - wartość
- obsługuje `in`, `len()` oraz iterację
- w Pythonie 3.0 jest `dict` oraz `collections.defaultdict` - słownik z wartościami domyślnymi
- słownik jest kolekcją nieuporządkowaną tak jak zbiór
- jako klucze mogą służyć tylko wartości hashowalne (tak jak przy zbiorach - `float`, `frozenset`, `int`, `str`, `tuple`)
- wartości mogą być dowolne
- słownik jest modyfikowalny
- można porównywać słowniki operatorami == i !=

# Słownik - tworzenie

```
d = dict({'id':1948, 'nazwa':'Zmywarka',
'rozmiar':3})
d2 = dict([('id',1948), ('nazwa','Zmywarka') ,
('rozmiar',3)])
d3 = dict(zip(('id','nazwa','rozmiar'),
(1948,'Zmywarka',3)))
d4 = {'id':1948, 'nazwa':'Zmywarka', 'rozmiar':3}
```

- d2 tworzy słownik z kolekcji par
- d3 używa funkcji zip która zwraca listę krotek odpowiednio sparowanych

# Słownik - wykorzystanie

- można odwoływać się do elementów słownika po kluczu

```
d1['nazwa']
```

- jeżeli nie ma klucza to wyjątek `KeyError`
- można wstawiać do słownika

```
d['x']=234
```

- usuwanie - `del` - uwaga na wyjątek `KeyError`

```
del d['x']
```

- słownik obsługuje funkcje `clear`, `copy`, `fromkeys`, `get`, `items`, `k eys`, `pop`, `popitem`, `setdefault`, `update`, `values`

# Iteracja po słowniku - po parach

```
for klucz,wartosc in d.items():
    print(klucz,wartosc)
```

```
for element in d.items():
    print(element[0],element[1])
```

# Iteracja po kluczach i wartościach

```
for klucz in d:
    print(klucz)
```

```
for klucz in d.keys():
    print(klucz)
```

```
for w in d.values():
    print(w)
```

# Widoki

- `keys`, `values`, `items` to widoki na słownik
- widok to niemutowalna sekwencja umożliwiająca iterację
- widok zostanie zaktualizowany w przypadku aktualizacji słownika
- na widokach można wykonywać operacje jak na zbiorach - suma, iloczyn, różnica, różnica symetryczna

```
d.fromkeys('ABCD',3)                    1
s=set('ACX')                            2
dopasowania = d.keys() & s              3
```

# Słowniki składane

- Słownik obsługuje podobnie jak lista i zbiór składanie

```
{klucz:wartosc for klucz,wartosc in iteracja if warunek}
wielkosci = {nazwa:os.path.getsize(nazwa) for nazwa in
  os.listdir('.') if os.path.isfile(nazwa)}
```

- odwracanie słownika - uwaga na `TypeError` przy mutowalnych wartościach

```
odwrocony = {v:k for k,v in d.items()}
```

# Słownik domyślny

- Słownik domyślny nigdy nie zgłasza `KeyError`
- jeżeli nie ma klucza, który jest wykorzystany w danej linii to zostanie wygenerowany wpis w słowniku z pustą lub domyślną wartością
- przy tworzeniu słownika domyślnego podaje się funkcję fabryczną - funkcję , która zwraca obiekt domyślny określonego typu

```
words = collections.defaultdict(int)
```

- uwaga - tutaj int jest referencją do funkcji! (funkcja jest w Pythonie obiektem)

```
words[word]+=1
```

- powyższy kod nie zgłosi wyjątku nawet jeżeli klucz word nie istnieje - zostanie utworzony wpis o wartości 0 i zwiększony o 1

# Zadanie 1 – konkurs jedzenia pączków

Konkurs w tłusty czwartek: kto zje najwięcej pączków w 10 minut? Na wejściu po kolei:

- unikalne imiona uczestników (w jednej linii)
- pojedyncza linia złożona z imion uczestników w takiej kolejności w jakiej kończyli jeść kolejne pączki

Na wyjściu:

- Lista krotek (imię uczestnika, liczba zjedzonych pączków) posortowana malejąco po liczbie zjedzonych pączków

Upraszczające założenie: każdy z uczestników zjadł inną liczbę pączków

# Zadanie 1 – przykładowe wejście i wyjście

Przykładowe wejście:

```
pawel kasia lukasz                                                    1
pawel kasia lukasz kasia pawel lukasz pawel kasia pawel
```

Przykładowe wyjście:

```
[('pawel', 4), ('kasia', 3), ('lukasz', 2)]                           1
```

# Zadanie 2 – Indeksowanie

- Wejście (w kolejnych linijkach):
  - liczba dokumentów do przetworzenia (n)
  - n linijek z wielowyrazowymi dokumentami (1 dokument na linijkę)
  - liczba zapytań do przetworzenia (m)
  - m linijek z jednowyrazowymi zapytaniami
- Wyjście:
  - m linijek (jedna na każde zapytanie)
  - w każdej linijce lista zawierająca numery (indeksy) dokumentów, w których wystąpił wyraz z danego zapytania
  - każda lista posortowana według częstości wystąpienia wyrazu z zapytania w danym dokumencie
  - przy równej częstości powinna być zachowana kolejność wczytywania dokumentów

# Zadanie 2 – przykład

```
3                                                                    1
Your care set up, do not pluck my care down.                         2
My care is loss of care with old care done.                          3
Your care is gain of care when new care is won.                      4
2                                                                    5
care                                                                 6
is                                                                   7
```

```
[1, 2, 0]                                                            1
[2, 1]                                                               2
```

# Zadanie 2 – dodatkowe założenia

- zapytania formułowane są małymi literami, ale odpowiedź powinna uwzględniać wyrazy, które w dokumentach zapisane są różną wielkością liter, np.:
  - dokument: A Rose is a rose is a roSE
  - zapytanie: rose
  - występowanie: 3 razy
- wyrazy powinny zawierać tylko znaki alfanumeryczne (bez interpunkcyjnych), tzn. wystąpienie znaku interpunkcyjnego lub spacji, tabulatora itp. powoduje podział ciągu znaków na oddzielne wyrazy,
- jeśli zapytanie nie zwraca rezultatów jako wynik na zapytanie powinna być wypisana pusta lista.

# Zadania – podpowiedź

```
#!/usr/bin/python3

import string

l = "aaa bbb ccc".split() # ["aaa","bbb","ccc"]
j = ",".join(l) # "aaa,bbb,ccc"

p = string.punctuation # ciag wszystkich
                       # znakow interpunkcyjnych
s = string.whitespace # ciag wszystkich
                      # znakow bialych

#jak posortowac slownik po wartosciach?
import operator
test_dict = dict({'a':1, 'c':50, 'f':17})
wynik = sorted(test_dict.items(),
               key = operator.itemgetter(1))
```

# Zadania – podpowiedź

Jak posortować słownik malejąco po wartościach i rosnąco po kluczach?
Sortujemy dwa razy:

- najpierw po kluczach rosnąco (w pierwszej linijce)
- następnie malejąco po wartościach (w drugiej linijce)

Odpowiednia kolejność w kluczach zostanie zachowana.