Złożone typy danych

Zbiory i operacje na nich

`tuple` - Krotka

```
a = (1, 2, 3)
a = tuple(1, 2, 3)
```

```
>>> def return_arguments(a, b):
...     return (a, b)

>>> out = return_arguments(10, 20)
>>> print(out)
(10, 20)
```

`list` - Lista

```
my_list = []
my_list = list()
my_list = [1, 2, None, False, 'hej']
```

```
>>> my_list = [1, 2, None, False, 'hej']
>>> my_list[2]
None
```

```
## Performance - Method concatenates strings using + in a loop
def make_html1(lista):
    html = '<table>'

    for element in lista:
        html += '<tr><td>%s</td></tr>' % element
    html += '</table>'

    return html

## Problem solved
def make_html2(lista):
    html = ['<table>']
    for element in lista:
        html.append('<tr><td>%s</td></tr>' % element)
    html.append('</table>')
    return '\r\n'.join(html)
```

`set` - Zbiór

```
>>> a = set([1, 3, 1])
>>> a
{1, 3}
```

Przykład trochę bardziej zaawansowany:

```
class Adres:
    def __init__(self, miasto):
        self.miasto = miasto


Adres(miasto='...')


{}
{'klucz': 'wartość'}
{'klucz', 'wartość'}
{'klucz'}
print({Adres(miasto='...'), Adres(miasto='...')})

a = Adres(miasto='...')
print({a, a})



print(dict(foo='bar'))
```

`dict` - Słownik

```
my_data = {
    "imie": "José",
    "nazwisko": 'Jiménez',
    'wiek': 10,
}

print(my_data['nazwisko'])
```

Dobieranie się do wartości elementów

`[...]` i `.get(...)`

Do zawartości zmiennej słownikowej możemy uzyskać dostęp używając nawiasów kwadratowych wraz z kluczem, albo funkcji `.get(klucz)`. Różnica między tymi podejściami polega na tym, że jeżeli dana zmienna słownikowa nie zawiera pewnego klucza, używanie nawiasów kwadratowych wygeneruje wyjątek KeyError, natomiast użycie funkcji `.get(klucz)` nie zwróci nic. Do funkcji `.get(klucz)` możemy dodatkowo dopisać wartość domyślną która zostanie zwrócona, jeżeli słownik nie posiada danego klucza.

```
>>> dane = {'imie': 'José', 'nazwisko': 'Jiménez'}

```

```
>>> dane['nazwisko']
'Jiménez'

>>> dane.get('nazwisko')
'Jiménez'

>>> dane['wiek']
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
KeyError: 'wiek'

>>> dane.get('wiek')

>>> dane.get('wiek', 'n/d')
'n/d'
```

Złożone typy danych

Lista słowników

```
studenci = [
    {'imie': 'Max'},
    {'imie': 'José', 'nazwisko': 'Jiménez'},
    {'imie': 'Ivan', 'nazwisko': None},
    {'imie': 'Buster', 'programuje w': ['python', 'java', 'c/c++']},
]

dane = studenci[0]['nazwisko']
dane = studenci[0].get('nazwisko', 'n/d')
dane = '\n'.join(studenci[4].get('programuje w'))
print(dane)
```

Listy wielowymiarowe

```
array = [
    [0, 1, 2],
    [1, 2, 3],
]
```

Mieszane typy

```
array = [
    [0, 1, 2],
    (1, 2, 3),
    set([1, 3, 1]),
    {'imie': 'José', 'nazwisko': 'Jiménez'}
]
```

Jak inicjować poszczególne typy?

- `list()` czy `[]`
- `tuple()` czy `()`
- `dict()` czy `{}`
- `set()` czy `{}`

Zadania kontrolne

Wyrazy

Napisz program, który na podstawie paragrafu tekstu "Lorem Ipsum" podzieli go na zdania () i dla każdego zdania wyświetli ile jest w nim wyrazów.

Założenia

: - kropka rozdziela zdania - spacja oddziela wyrazy w zdaniu

Podpowiedź

:

> - `str.split()`
> - `len()`

Przeliczanie odległości

Napisz program który przekonwertuje odległości (podane w metrach) i zwróci `dict`, zgodnie z szablonem:

```
{
    'kilometers': int(),
    'miles': float(),
    'nautical miles': float(),
}
```

Podpowiedź

: - 1000 m = 1 km - 1608 m = 1 mila - 1852 m = 1 mila morska