Python programming for beginners

Stefan Zhauryd
Instructor

Module 4

Functions, Tuples, Dictionaries, and Data Processing

In this module, you will learn about:

- code structuring and the concept of function;
- function invocation and returning a result from a function;
- name scopes and variable shadowing;
- **tuples and their purpose, constructing and using tuples;**
- **dictionaries and their purpose, constructing and using dictionaries.**

Tuples and dictionaries can work together

Let's imagine the following problem:

- you need a program to evaluate the students' average scores;
- the program should ask for the student's name, followed by her/his single score;
- the names may be entered in any order;
- entering an empty name finishes the inputting of the data (note 1: entering an empty score will raise the ValueError exception, but don't worry about that now, you'll see how to handle such cases when we talk about exceptions
- a list of all names, together with the evaluated average score, should be then emitted.

Tuples and dictionaries can work together

type()
id()

{“Andy”: (3, 10, 3)}

```
school_class = {}

while True:
    name = input("Enter the student's name: ")
    if name == '':
        break

    score = int(input("Enter the student's score (0-10): "))
    if score not in range(0, 11):
        break

    if name in school_class:
        school_class[name] += (score,)
    else:
        school_class[name] = (score,)

for name in sorted(school_class.keys()):
    adding = 0
    counter = 0
    for score in school_class[name]:
        adding += score
        counter += 1
    print(name, ":", adding / counter)
```

```
Enter the student's name: Bob
Enter the student's score (0-10): 7
Enter the student's name: Andy
Enter the student's score (0-10): 3
Enter the student's name: Bob
Enter the student's score (0-10): 2
Enter the student's name: Andy
Enter the student's score (0-10): 10
Enter the student's name: Andy
Enter the student's score (0-10): 3
Enter the student's name: Bob
Enter the student's score (0-10): 9
Enter the student's name:
Andy : 5.333333333333333
Bob : 6.0
```

Key takeaways: tuples

```
my_tuple = (1, 2, True, "a string", (3, 4), [5, 6], None)
print(my_tuple)

my_list = [1, 2, True, "a string", (3, 4), [5, 6], None]
print(my_list)
```

```
my_tuple_1 = 1,
print(type(my_tuple_1))     # outputs: <class 'tuple'>

my_tuple_2 = 1              # This is not a tuple.
print(type(my_tuple_2))     # outputs: <class 'int'>
```

```
empty_tuple = ()
print(type(empty_tuple))     # outputs: <class 'tuple'>
```

```
my_tuple = 1, 2, 3,
del my_tuple
print(my_tuple)     # NameError: name 'my_tuple' is not defined
```

```
one_elem_tuple_1 = ("one", )     # Brackets and a comma.
one_elem_tuple_2 = "one",        # No brackets, just a comma.
```

```
my_tuple = (1, 2.0, "string", [3, 4], (5, ), True)
my_tuple[2] = "guitar"     # The TypeError exception will be raised.
```


https://www.w3schools.com/python/python_ref_tuple.asp

Key takeaways: tuples

```
# Example 1
tuple_1 = (1, 2, 3)
for elem in tuple_1:
    print(elem)

# Example 2
tuple_2 = (1, 2, 3, 4)
print(5 in tuple_2)
print(5 not in tuple_2)

# Example 3
tuple_3 = (1, 2, 3, 5)
print(len(tuple_3))

# Example 4
tuple_4 = tuple_1 + tuple_2
tuple_5 = tuple_3 * 2

print(tuple_4)
print(tuple_5)
```

int()
tuple()
list()
dict()

```
my_tuple = tuple((1, 2, "string"))
print(my_tuple)

my_list = [2, 4, 6]
print(my_list)     # outputs: [2, 4, 6]
print(type(my_list))     # outputs: <class 'list'>
tup = tuple(my_list)
print(tup)     # outputs: (2, 4, 6)
print(type(tup))     # outputs: <class 'tuple'>
```

https://www.w3schools.com/python/python_ref_tuple.asp

Key takeaways: dictionaries

```
my_dictionary = {
    key1: value1,
    key2: value2,
    key3: value3,
    }
```

```
pol_eng_dictionary = {
    "zamek": "castle",
    "woda": "water",
    "gleba": "soil"
    }

for item in pol_eng_dictionary:
    print(item)
```

```
pol_eng_dictionary = {
    "kwiat": "flower",
    "woda": "water",
    "gleba": "soil"
    }

item_1 = pol_eng_dictionary["gleba"]    # ex. 1
print(item_1)    # outputs: soil

item_2 = pol_eng_dictionary.get("woda")
print(item_2)    # outputs: water
```

```
pol_eng_dictionary = {"kwiat": "flower"}

pol_eng_dictionary.update({"gleba": "soil"})
print(pol_eng_dictionary)    # outputs: {'kwiat': 'flower', 'gleba': 'soil'}

pol_eng_dictionary.popitem()
print(pol_eng_dictionary)    # outputs: {'kwiat': 'flower'}
```

https

```
pol_eng_dictionary = {
    "zamek": "castle",
    "woda": "water",
    "gleba": "soil"
    }

for key, value in pol_eng_dictionary.items():
    print("Pol/Eng ->", key, ":", value)
```

Key takeaways: dictionaries

```
pol_eng_dictionary = {
    "zamek": "castle",
    "woda": "water",
    "gleba": "soil"
    }

if "zamek" in pol_eng_dictionary:
    print("Yes")
else:
    print("No")
```

```
pol_eng_dictionary = {
    "zamek": "castle",
    "woda": "water",
    "gleba": "soil"
    }

pol_eng_dictionary["zamek"] = "lock"
item = pol_eng_dictionary["zamek"]
print(item)  # outputs: lock
```

```
pol_eng_dictionary = {
    "zamek": "castle",
    "woda": "water",
    "gleba": "soil"
    }

print(len(pol_eng_dictionary))    # outputs: 3
del pol_eng_dictionary["zamek"]    # remove an item
print(len(pol_eng_dictionary))    # outputs: 2

pol_eng_dictionary.clear()   # removes all the items
print(len(pol_eng_dictionary))    # outputs: 0

del pol_eng_dictionary    # removes the dictionary
```

```
pol_eng_dictionary = {
    "zamek": "castle",
    "woda": "water",
    "gleba": "soil"
    }

copy_dictionary = pol_eng_dictionary.copy()
```

https://www.w3schools.com/python/python_ref_dictionary.asp

Key t

```python
st = '2323fdsfsdfsd'
li = list(st)
tupl = tuple(st)
# di = dict(st)  # Will be error.
# i = int(st)  # Will be error.
# f = float(st)  # Will be error.
# b = bin(st)

# 2323fdsfsdfsd <class 'str'>.
print(st, type(st))

# ['2', '3', '2', '3', 'f', 'd', 's', 'f', 's', 'd', 'f', 's', 'd'] <class 'list'>.
print(li, type(li))

# ('2', '3', '2', '3', 'f', 'd', 's', 'f', 's', 'd', 'f', 's', 'd') <class 'tuple'>.
print(tupl, type(tupl))


ss = str(li)
# ['2', '3', '2', '3', 'f', 'd', 's', 'f', 's', 'd', 'f', 's', 'd'] <class 'str'>.
print(ss, type(ss))

ss = str(tupl)
# ('2', '3', '2', '3', 'f', 'd', 's', 'f', 's', 'd', 'f', 's', 'd') <class 'str'>.
print(ss, type(ss))

```

Home work 6.1

ToDO list

"Mon": ()

[task_number] 1-15

[task_name] != "" len > 7

[priority] = 1-100

Data:

- format: ‘task_number’+’task_name’+’priority’
- Input what you want to do while input not equal "stop"

I want to eat
I want to play
and so on

- Exit by word - **stop**

Functions:

- display_task_list(list_task)
- display_cond_list()
- enter_task(list_task)
- main()

Home work 6.2

ToDO list

- format:
- Dict {task_number : ”task_name”}
- Input what you want to do while input not equal "stop"

I want to eat
I want to play
and so on

- **stop**

ЗАДАНИЯ

1) Прорешать всю классную работу
2) Выполнить все домашние задания

Почитать:

1) Byte of Python
Прочитать страницы -
стр. 88-94

Крайний срок сдачи 07/10 в 21:00 (можно раньше, но не позже)

ЗАДАНИЯ

Название файлов, которые вы отправляете мне в telegram:

Vasia_Pupkin_class_work_L6_P1.py
Vasia_Pupkin_L6_1.py
Vasia_Pupkin_L6_2.py

Формат сообщения которое вы присылаете мне
(после полного выполнения домашнего задания, только один раз) в Telegram:
Добрый день/вечер. Я Вася Пупкин, и это мои домашние задания к лекции 6 часть 1 про кортежи и словари.
И отправляете файлы

Крайний срок сдачи 07/10 в 21:00 (можно раньше, но не позже)

https://docs.github.com/articles/using-pull-requests

Create your possibilities. Bye bye.