ADAM MICKIEWICZ UNIVERSITY IN POZNAŃ
Faculty of English

Robert Dyzman, M.Sc.Eng.

PYTHON PROGRAMMING
CLASS 05

Run „Teams”

Start your IDE

AGENDA:

- Create a file *„class_pp_05.py”*
- Quiz 03
- Dictionary comprehension
- Exercises
- Programming Task 01 (9.10 – 9.30 AM) -> Copy/Paste to Teams

DICTIONARY REVISITED

```
#####################################
# # DICTIONARY REVISITED
# # (word length dictionary)
# #####################################
words = ['data', 'science', 'machine', 'learning']
word_length = {}
for word in words:
    word_length[word] = len(word)
print(word_length)
```

DICTIONARY REVISITED

```
###################################
# DICTIONARY REVISITED
# (shakespeare text length analyzer)
# work with 'chaining'
###################################
shakespeare = 'When forty winters shall besiege thy brow,
And dig deep trenches in thy beautys field, ....'
shakespeare = shakespeare.replace(',', '').replace('.', '')
words = shakespeare.split()
word_length = {}
for word in words:
    word_length[word] = len(word)
print(word_length)
```

DICTIONARY COMPREHENSION

- Syntax
- **{item[0]:item[1] for item in iterable}**
- **{key: value for (key, value) in dict.items()}**

→ for each item in items, an expression is applied

→ Result: dictionary of key:value pairs

```
###############
# # # DICTIONARY COMPREHENSIONS
# # (word length dictionary)
# # ###############
words = ['data', 'science', 'machine', 'learning']
word_length = {word: len(word) for word in words}
print(word_length)
```

DICTIONARY COMPREHENSION

```
###############
# # DICTIONARY COMPREHENSION
# ###############
shakespeare = 'When forty winters shall besiege thy brow,
And dig deep trenches in thy beautys field, ....'
words = shakespeare.replace(',', '').replace('.',
'').split()
word_legth = {word: len(word) for word in words}
print(word_legth)
```

DICTIONARY COMPREHENSION

```
# # so far in loop section we had lists
# # now let's try with dictionary
# # have to use items() method
product_price_dic = {
    'Bread': 10,
    'Butter': 20,
    'Chocolate dark': 15,
    'Chocolate white': 17,
    'Cakes': 19
}

product_price_increased = {k: v*1.2 for k, v in
product_price_dic.items()}
print(product_price_increased)
```

SORTING DICTIONARIES

```
############################
# SORTING DICTIONARIES ON VALUE
# If we want to get a sorted copy of the entire dictionary,
# we need to use the dictionary items() method
# which pass in the entire dictionary as the iterable to
#the sorted() function:
############################
shakespeare = 'When forty winters shall besiege thy brow, And dig
deep trenches in thy beautys field, ....'
words = shakespeare.replace(',', '').replace('.', '').split()
word_legth = {word: len(word) for word in words}
sorted_word_length = sorted(
    word_legth.items(), key=lambda item: item[1], reverse=True)
print(sorted_word_length)
print(dict(sorted_word_length))
```

EXERCISE 60 (file „pp_60.py”) – USING LOOP

```
'''
Exercise 60
Write a program that will print to display a list of
tuples that will consist of the most frequent character
in the sentence / sentences and its frequency,
sorted in a descending order.
In order to complete the task
1. define a function
def most_freq_character(sentence):
    characters_freq = {}
    # {'r':7, 'l':5, 'o':1}
    # your code
    return sorted_characters_freq
```

EXERCISE 60

```
'''
2. As a data structure use a dictionary
3. In your analyze, don't use spaces

For testing purposes, you can  use
sentence = "The robbed that smiles, steals something from
the thief."
Expected result:
[('t', 7), ('e', 7), ('h', 5), ('s', 5), ('o', 3), ('m',
3), ('i', 3), ('r', 2), ('b', 2), ('a', 2), ('l', 2),
('f', 2), ('d', 1), (',', 1), ('n', 1), ('g', 1), ('.',
1)]
'''
```

EXERCISE 70 – USING DICTIONARY COMPREHENSION

```
'''
Exercise 70, file "pp_70.py"
LEARNING DICTIONARY COMPREHESION
Write a program that will print to display a list of tuples that will consist of
the most frequent character in the sentence / sentences and it's frequency,
sorted in a descending order.
In order to complete the task
1. define a function
def most_freq_character(sentence):
2. As a data structure use a dictionary
3. In your analize, don't use spaces, coma, dots
4. use count() method of string class to count characters

For testing purposes you can  use
sentence = "The robbed that smiles, steals something from the thief."
Expected result:
[('t', 7), ('e', 7), ('h', 5), ('s', 5), ('o', 3), ('m', 3), ('i', 3), ('r', 2),
('b', 2), ('a', 2), ('l', 2), ('f', 2), ('d', 1), ('n', 1), ('g', 1)]
'''
```

EXERCISE 80 USING DICTIONARY COMPREHENSION

```
'''
Exercise 80, file "pp_80.py"
The text is given:
shakespeare = 'When forty winters shall besiege thy brow, And dig deep
trenches in thy beautys field, ....'
Write a program that will find and print to the terminal the longest
word in the text.
In order to achieve it, implement the function that will delete all '.',
',', get the list of all words,
use dictionary comprehension to get word:length pairs and use sorted()
function.

Expected result:
The longest word is 'trenches' with the length 8 characters.

'''
```

EXERCISE 80

```
def get_longest_word(text: str) -> tuple[str, int]:
    '''
    Function returns the longest word together with its length

    Parameters:
        text: any text
    Returns:

    Exemplary result:
    ('trenches',8)
    '''
```

EXERCISE 90 (file pp_90.py)

```
'''
Write a function that will check if a word/sentence is a
palindrome or not. (20 min.)
Use iterative way
These are palindromes, test with them:
Dad
Evil olive.
Never odd or even.
Amore, Roma.
Not palindromes:
test
ad
a
'''
```

PROGRAMMING TASK 01

```
'''
Programming task 01, file pt_01.py, 20 min.
Grade : 5 points

Write a program that will remove punctuation from any string,
optionally it should remove spaces.
Implement the function definition

Expected exemplary result:
Here is a text without punctuations: 'Wow Thats amazing How did
you do it Im so impressed Youre awesome brilliant and talented
Congratulations Well done Bravo'

Send to Teams !!!
'''
```

PROGRAMMING TASK 01

```
sample_text = "Wow! That's amazing. How did you do it? I'm so impressed. You're
awesome, brilliant, and talented. Congratulations! Well done. Bravo!"


def remove_punctuation(text: str, remove_space: bool = False) -> str:
    '''
    Function removes punctuations !"#$%&'()*+,-./:;<=>?@[\]^_`{|}~ optionally it
can remove spaces

    Parameters:
        text: any text
        remove_space: if True, spaces are removed, if False, spaces are not
removed
    Returns:
        text without punctuation, optionally spaces.
    '''
    import string
    punct = string.punctuation
```