PYTHON TRAINING

Map, Filter, Sorted

Document

- https://www.learnpython.org/en/Map,_Filter,_Reduce
- https://book.pythontips.com/en/latest/map_filter.html
- https://holypython.com/intermediate-python-lessons/lesson-13-python-map/
- https://holypython.com/intermediate-python-lessons/lesson-14-filter/

Exercise

Q1: Use "map" function for data type conversion

Input:
list = [10.8932, 12.434, 13.65656]

Output:
list = [10, 12, 13]

Q2: Use "filter" function to get positive numbers

Input:
list = [4, -3, 2]

Output:
list = [4, 2]

Q3: Use "sorted" function to reverse a list

Input:
list = [4, -3, 2]

Output:
list = [2, -3, 4]

Dictionary

Document

- https://www.geeksforgeeks.org/python-dictionary
- https://www.pythonlikeyoumeanit.com/Module2_EssentialsOfPython/DataStructures_II_Dictionaries.html
- https://www.datacamp.com/community/tutorials/python-dictionary-comprehension
- https://www.geeksforgeeks.org/python-dictionary-exercise/

Exercise

Q1: Use "zip" function to map two lists into a dictionary

Input:
Keys = ['Key 1', 'Key 2', 'Key 3']
Values = ['Value 1', 'Value 2', 'Value 3']

Output:
Dict = {'Key 1': 'Value 1', 'Key 2': 'Value 2', 'Key 3': 'Value 3'}

Q2: Check multiple keys exists in a dictionary

Input:
Dict = {'Key 1': 'Value 1', 'Key 2': 'Value 2', 'Key 3': 'Value 3'}
list1 = ['Key 1', 'Key 5']
list2 = ['Key 2', 'Key 3']

Output:
Check list1 -> False
Check list2 -> True

Q3: Remove a key from a dictionary

Input:
Dict = {'Key 1': 'Value 1', 'Key 2': 'Value 2'}
Key = 'Key 1'

Output:
Dict = {'Key 2': 'Value 2'}

Q4: Iterate over dictionaries using for loops.

Input:
Dict = {‘Key 1’: ‘Value 1’, ‘Key 2’: ‘Value 2’}

Output:
Key 1 corresponds to Value 1
Key 2 corresponds to Value 2

Q5: Count number of items in a dictionary value that is a list by using sum and map

Input:
Dict = {‘Key 1’: [‘Value 1’, ‘Value 2”], ‘Key 2’: [‘Value 3’, ‘Value 4’, ‘Value 5’]}

Output:
5

Q6: Store a given dictionary in a json file

Input:
Dict = {‘Key 1’: [‘Value 1’, ‘Value 2”], ‘Key 2’: [‘Value 3’, ‘Value 4’, ‘Value 5’]}

Output:
JSON file

List

Document

- https://www.geeksforgeeks.org/advanced-python-list-methods-and-techniques/
- https://realpython.com/list-comprehension-python/

Exercise

Q1: Concatenate two lists index-wise by using list comprehension

Input:
list1 = [‘W, ‘a’, ‘Novo’]
list2 = [‘e’, ‘re’, bi’]

Output:
List = [‘We’, ‘are’, ‘Novobi’]

Q2: Concatenate two lists in the following order

Input:
list1 = [‘Hello ’, ‘ take ‘]
list2 = [‘Dear’, ‘Sir‘]

Output:
List = [‘Hello Dear’, ‘Hello Sir’, ‘take Dear’, ‘take Sir’]

Q3: Iterate both lists simultaneously such that list1 should display item in original order and list2 in reverse order by using zip function

Input:
list1 = [‘10’, ‘20’, ‘30’]
list2 = [‘100’, ‘200’, ‘300’]

Output:
10 100
20 200
30 300

Q4: Remove empty strings from the list of strings by using filter function

Input:
List = [‘10’, ‘20’, ‘30’, ‘’, ‘40’]

Output:
List = [‘10’, ’20’, ‘30’, ‘40’]

Q5: Remove all occurrence of 20 from the list by using list comprehension

Input:
List = [‘10’, ’20’, ’30’, ‘40’, ‘20’, ‘50’]

Output:
List = [‘10’, ‘30’, ‘40’, ‘50’]

Q6: Sort a list of dictionaries using Lambda.

Input:
List = [{'make': 'Nokia', 'model': 216, 'color': 'Black'}, {'make': 'Mi Max', 'model': '2', 'color': 'Gold'}, {'make': 'Samsung', 'model': 7, 'color': 'Blue'}]

Output:
[{'make': 'Nokia', 'model': 216, 'color': 'Black'}, {'make': 'Samsung', 'model': 7, 'color': 'Blue'}, {'make': 'Mi Max', 'model': '2', 'color': 'Gold'}]

Iterator

Document

- https://www.programiz.com/python-programming/iterator
- https://www.geeksforgeeks.org/iterators-in-python/
- https://www.geeksforgeeks.org/python-difference-iterable-iterator/

Exercise

Instead of using “enumerate” function, create your own class. It will need to return a tuple with each iteration, with the first element in the tuple being the index (starting with 0), and the second element being the current element from the underlying data structure

Generators

Document

- https://www.programiz.com/python-programming/generator
- https://realpython.com/introduction-to-python-generators
- https://topdev.vn/blog/tai-sao-ban-nen-su-dung-python-generator/

Exercise

Use generator to complete the below Python program

```
def fibonacci_numbers(nums):
    # TODO
def square(nums):
    TODO

print(sum(square(fibonacci_numbers(10))))
```

Output:

4895

Itertools

Document

- https://realpython.com/python-itertools
- https://www.geeksforgeeks.org/python-itertools/

Exercise

Q1: Use itertools to create groups of similar items of a given list.

Input:

names = [‘Thomas Brown’, ‘Tom Smith’, ‘Jane Brown’, ‘John Smith’]

Output:

[[‘Thomas Brown’, ‘Jane Brown’], [‘Tom Smith’, ‘John Smith’]]

Q2: Use itertools to combine two list into a new list

Input:

list1 = [[‘We’, ‘are’], ‘Novobi’]

list2 = [‘We’, [‘are’, ‘Odoo’]]

Output:

[‘We’, ‘are’, ‘Novobi’, ‘We’, ‘are’, ‘Odoo’]

Q3: Generate unique combinations

Input:

list = [‘Red’, ‘Green’, ‘Blue’]

Output:

list = [‘Red and Green’, ‘Red and Blue’, ‘Green and Blue’]

Regular expressions

Document

- https://docs.python.org/3/library/re.html
- https://realpython.com/regex-python/

Exercise

Use regular expressions to check if email is valid or not

Input: ankitrai326@gmail.com

Output: Valid Email

Input: my.ownsite@ourearth.org

Output: Valid Email

Input: ankitrai326.com

Output: Invalid Email

Partial Functions

Document

- https://www.geeksforgeeks.org/partial-functions-python/
- https://www.udacity.com/blog/2020/12/how-to-create-partial-functions-in-python.html
- https://docs.python.org/3/library/functools.html

Exercise

Use partial function from functools module to rewrite the method for doubleNum and tripleNum

```
def multiply(x, y):
    return x * y
def doubleNum(x):
    return multiply(x, 2)
def tripleNum(x):
    return multiply(x, 3)
```

Decorators

Document

- https://gist.github.com/Zearin/2f40b7b9cfc51132851a
- https://realpython.com/primer-on-python-decorators/

Exercise

Q1: Write the decorator, let the user enter the user name, password, and give the user three chances before each execution of the decorated function. After the login is successful, the function can be accessed.

Q2: Make a chain of function decorators (bold, italic, underline etc.)