Python Exercises

The goal of this assignment is to get you familiarized with basic python functionalities which were discussed in the class. You will have to apply the concepts taught in class today and feel free to go through the class notebook or the other sources mentioned in the notebook to gain more clarity of the concepts.
You will have to submit a pdf copy of the notebook in slack and the submission deadline will be announced to you in class. To submit, either print the page in pdf format or you can also use the 'download as' option in the 'file' menu(you may face some errors).

Practise makes a man perfect!!! So lets get started with the exercise problems.

Write a Python program to get the Python version you are using.

In [101…

```
# Its a good practice to keep all your imports in the first code-block!

# import <package name>
import sys
import math
import functools
```

In [102…

```
print(sys.version)
```

```
3.8.5 (default, Sep  4 2020, 07:30:14)
[GCC 7.3.0]
```

Write a Python program which accepts the radius of a circle from the user and compute the area. (Use math library)

In [103…

```
# Taking input from the user and casting it to the float
radius  = float(input('Please enter the radius of the circle: '))

# Area of circle is calculated as follows
area_of_circle = math.pi * (radius**2)

# Printing the output
area_of_circle
```

```
Please enter the radius of the circle: 500
```

Out[103…

```
785398.1633974483
```

Write a Python program to concatenate all elements in a list into a string and return it.

In [104…

```
'''
This is the function that will take a list as an input and also a optional pa
It will return a single string which is concatenated with the provided delimi

'''
def concat_elements_of_lists(list_of_elements,delimiter=""):
    resultant_string = ''
    for elem in list_of_elements:
        resultant_string += str(elem)+delimiter
    return resultant_string

# print(concat_elements_of_lists(['apple','banana']," "))
```

In [105…

```
# Here, we are initializing a random list of fruits for testing
list_of_fruits = ['banana','apple','pear','strawberry','orange']
```

```
# Testing our function
print(concat_elements_of_lists(list_of_fruits,""))

# similarly, using python in-built function, we can do it one line as follows
print("".join(list_of_fruits))
```

```
bananaapplepearstrawberryorange
bananaapplepearstrawberryorange
```

Write a Python program to calculate the area of a trapezoid.

Test Data

Height : 5

Base, first value : 5

Base, second value : 6

Expected Output: Area is : 27.5

In [106…

```
# We will be defining a function to calculate area of trapezoid

def calculate_area_of_trapezoid(height,first_base,second_base):
    return (((first_base + second_base)/2)* height)
```

In [107…

```
# Testing our trapezium's area calculator

area_of_trapezoid = calculate_area_of_trapezoid(5,5,6)
print('Area is : ',area_of_trapezoid)
```

```
Area is :  27.5
```

Write a Python program to remove and print every third number from a list of 50 natural numbers until the list cannot be reduced any further.

In [108…

```
# Natural numbers start from 1 and go-on till infinity

# Generating numbers from 1 to 50
natural_nums = list(num for num in range(1,51))

# Generating 50 multiples of 3
multiple_of_three = list((number*3-1) for number in range(1,51))

# Popping and printing every third element from natural_nums list with list-c
print([natural_nums.pop(index%len(natural_nums)) for index in multiple_of_thr
```

```
[3, 7, 11, 15, 19, 23, 27, 31, 35, 39, 43, 47, 1, 8, 14, 21, 28, 34, 41, 48,
5, 16, 25, 36, 45, 6, 20, 33, 49, 13, 32, 2, 26, 50, 29, 10, 44, 38, 37, 42,
12, 46, 4, 30, 18, 22, 9, 40, 17, 24]
```

Write a Python program to print the following text, convert the string to a list and print all the words and their frequencies

"Probability is the measure of the likelihood that an event will occur. Probability is quantified as a number between 0 and 1, where, loosely speaking, 0 indicates impossibility and 1 indicates certainty. The higher the probability of an event, the more likely it is that the event will occur."

In [109…

```
# Testing out the logic to filter out non alpha_numeric chars
("".join(filter(lambda x:x.isalnum(),"Hello.".strip()))).lower()

# Works fine till now!
```

Out[109…

```
'hello'
```

In [110…

```
"""
This function will convert the given string into lower case alpha numeric str
This helps us acquire consistency amongst all the words in our string.
"""

def convert_to_only_lower_alpha_num(string):

    # If the string is already alpha numeric, just convert into lower case ar
    if string.isalnum():
        return string.strip().lower()

    else:
        string = "".join(filter(lambda x:x.isalnum(),string.strip())).lower()
        return str(string)
```

In [111…

```
# Testing our function!
print(convert_to_only_lower_alpha_num("'Probability."))

# Works good!
```

```
probability
```

In [112…

```
"""Testing out dictionaries
test_dict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
year = 'year1'
if year in test_dict:
    print('Exists')
    test_dict[year]+=1
    print('\n',test_dict['year'])
else:
    print('Please add the key!')
    test_dict[year]=1
    print(test_dict)"""
```

Out[112…

```
'Testing out dictionaries\ntest_dict = {\n  "brand": "Ford",\n  "model": "Mus
tang",\n  "year": 1964\n}\nyear = \'year1\'\nif year in test_dict:\n    print
(\'Exists\')\n    test_dict[year]+=1\n    print(\'\n\',test_dict[\'year\'])\n
else:\n    print(\'Please add the key!\')\n    test_dict[year]=1\n    print(t
est_dict)'
```

In [113…

```
# Taking the given string in the problem statement.
given_string = "Probability is the measure of the likelihood that an event wi

# Splitting it wherever we find the spaces
splitted_string = given_string.split(" ")

# Converting all the strings in the list into lower case alpha_numeric string
updated_splitted_string=list(map(convert_to_only_lower_alpha_num,splitted_str

# Checkpoint -> Testing out if all the logic till this point works out as des
# print(updated_splitted_string)

# Initializing an empty dictionary to hold the word as keys as and the word_c
frequencies = dict()

for string in updated_splitted_string:
    if string in frequencies:
        frequencies[string]+=1
    else:
        frequencies[string]=1

print(frequencies)
```

```
{'probability': 3, 'is': 3, 'the': 6, 'measure': 1, 'of': 2, 'likelihood': 1,
'that': 2, 'an': 2, 'event': 3, 'will': 2, 'occur': 2, 'quantified': 1, 'as':
1, 'a': 1, 'number': 1, 'between': 1, '0': 2, 'and': 2, '1': 2, 'where': 1,
'loosely': 1, 'speaking': 1, 'indicates': 2, 'impossibility': 1, 'certainty':
1, 'higher': 1, 'more': 1, 'likely': 1, 'it': 1}
```

Write a Python program to reverse the digits of a given number and add it to the original, If the sum is not a palindrome repeat this procedure.

Note: A palindrome is a word, number, or other sequence of characters which reads the same backward as forward, such as madam or racecar.

In [114…

```python
# Commenting original logic
# reversed_string = reversed(given_string)
# return given_string==reversed_string

def is_palindrome(given_string):
    # Ensuring the given_string is actually a string
    given_string = str(given_string)

    # Using shorthand to reverse a string
    return given_string==given_string[::-1]
```

In [115…

```python
# Testing the function is_palindrome()

# Case 1: is_palindrome(123321) -> should return True
print(is_palindrome(123321))

# Case 2: is_palindrome(12321) -> should return True
print(is_palindrome(12321))

# Case 3: is_palindrome(123121) -> should return False
print(is_palindrome(123121))
```

```
True
True
False
```

In [116…

```python
def convert_digits_to_palindromic_string(number):
    reversed_number = int(str(number)[::-1])
    sum_of_nums =number+reversed_number
    if is_palindrome(sum_of_nums):
        print("Palindrome found!\nNumber is : ", sum_of_nums)
    else:
        convert_digits_to_palindromic_string(sum_of_nums)
```

In [117…

```python
# Running our function
convert_digits_to_palindromic_string(456)
```

```
Palindrome found!
Number is :  1221
```

Write a Python program to generate a list of squares of numbers between 1 and 20 (both nos included) and print ONLY the first and last 5 elements.

In [118…

```python
# Generating the square of numbers from 1 to 20
list_of_squares = [elem**2 for elem in range(1,21)]

# Checking if this is properly generated:
# print("List of Square of numbers from 1 to 20 : ", list_of_squares)

# Printing only the first five elements of the new list
print("\n\nFirst Five Elements are : ",list_of_squares[:5])
```

```
# Printing only the last five elements of the new list
print("\n\nLast Five Elements are : ",list_of_squares[-5:])
```

```
First Five Elements are :  [1, 4, 9, 16, 25]


Last Five Elements are :  [256, 289, 324, 361, 400]
```

Write a Python program to split the following sentence based on white space and then extract the first letter of each split string.

"Jack of all trades but master of none"

In [119…

```
# Taking the given string
given_string = "Jack of all trades but master of none"

# Splitting the string into a list of strings based on spaces
splitted_string = given_string.split(" ")

# Initializing an empty list
list_of_first_chars = []

# Adding all the first chars to our new list
for elem in splitted_string:
    list_of_first_chars.append(elem[0])

# Printing the final list
print ( "".join(list_of_first_chars))

# Or a shorter way is a one-liner
print("".join([elem[0] for elem in splitted_string]))
```

```
Joatbmon
Joatbmon
```

Write a Python program to remove the duplicate elements in the list but preserve the order. Eg: [2,3,5,4,2,3,8,4] should give an output [2,3,5,4,8]

In [120…

```
original_list = [2,3,5,4,2,3,8,4,2,20,12312,12313,12312,2,2]

# Original Logic - Start
# list_with_removed_duplicate_elems = []

# for elem in original_list:
#     if elem not in list_with_removed_duplicate_elems:
#         list_with_removed_duplicate_elems.append(elem)

# Original Logic - End

# With List Comprehension
list_comp = []
[list_comp.append(num) for num in original_list if num not in list_comp]


print(list_comp)
```

```
[2, 3, 5, 4, 8, 20, 12312, 12313]
```

Write a Python program to solve the Fibonacci sequence using recursion.

In [121…

```
# Initialising the seed values n1=0 and n2=1
n1 = 0
n2 = 1

# Creating a list with our seed points as the first two elements in it.
fibonacci_series=[0,1]
```

```
"""
Generating the fibonacci series using the recursion.
num1 is the first number.
num2 is the second number.
count is the number of numbers we want to generate in fibonacci series.


We add the num1 and num2 and the sum is the next number in the series.
We print the number for each addition cycle
After each addition operation we decrement the count, and once the count reac

"""
def fibonacci(num1,num2,count):
    if count==1:
        fibonacci_series.append(num1+num2)
    else:
        count-=1
        sum = num1+num2
        num1=num2
        num2 = sum
        fibonacci_series.append(sum)
        fibonacci(num1,num2,count)

# Calling our fibonacci function with the seed values n1 and n2 and the count
fibonacci(n1,n2,10)

# Printing the generated fibonacci_serieabss
print(fibonacci_series)
```

```
[0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89]
```

Given an array of integers of size n, calculate the maximum sum of k consecutive elements in the array.

- Input: [100, 200, 300, 400]; k = 2
- Output: 700

In [122…

```
"""
This method returns the maximum sum of k consecutive elements in the array.
Firstly, we arrange the list elements in descending order.
Then, we sum up the k consecutive number, which ensure that it is the maximum
"""
def max_sum_of_consecutive_elems(given_list,k):
    # Sorting the list in descending order
    given_list.sort(reverse=True)

    # Sum up the first k consecutive numbers
    return sum(given_list[:k])
```

In [123…

```
my_input_list = [100,200,300,400] # List to test our function
my_input_count=2 # k to select consecutive elements

# calling and testing our function
print(max_sum_of_consecutive_elems(my_input_list,my_input_count))
```

```
700
```

Write a Python program to print a dictionary with keys are the names of the numbers between one and five (both included) and values are the cubes of the numbers.
Note: You should use a list to create the keys and create the dictionary using list comprehension.

In [124…

```
# Step1 : Generate numbers 1 to 5 using list comprehension
number_list = [element for element in range(1,6)]

# To test if the numbers are generated properly
# print(number_list)

# Step2 : Creation of dictionary with comprehension where ->
#         'key' is the number from above generated list,
#         'value' would be the cube of the key

dict_with_nums_and_cubes = {key:key**3 for key in number_list}

# Checking if the above dict comprehension worked properly
print(dict_with_nums_and_cubes)
```

```
{1: 1, 2: 8, 3: 27, 4: 64, 5: 125}
```

Write a program which accepts a sequence of comma separated 4 digit binary numbers as its input and then check whether they are divisible by 5 or not. The numbers that are divisible by 5 are to be printed in a comma separated sequence.

Example:

0110,1001,0100,1010

Then the output should be:

1010 Hint: Look up how to convert from integer to binary/hex/oct in python

In [125…

```
# Testing the conversion of binary string into an integer

my_binary = "0110"
print(int(my_binary,base=2))
```

```
6
```

In [126…

```
# Sample Input-> 0110,1001,0100,1010,1010,0101

# Taking the binary numbers from standard input
input_string = input("Please enter 4-digit comma-seperated Binary numbers : '

# splitting the numbers based on comma
list_binary_nums_in_string = input_string.split(',')

# print(list_binary_nums_in_string)

# List comprehension
final_list_with_binary_nums_divisible_by_5 = [binary for binary in list_binar

# Final output ->
print(",".join(final_list_with_binary_nums_divisible_by_5))
```

```
Please enter 4-digit comma-seperated Binary numbers : 0110,1001,0100,1010,101
0,0101
1010,1010,0101
```

Write a python program to check whether a given number is an Armstrong number using a function.

Note : An Armstrong number is one whose sum of cubes of digits is equal to the number itself.

Eg: 371

In [127…

```
"""# This block tests out the logic with 2 methods!

# We take a number and convert it to a string!
number_string = str(371)

# Method 1 - without list comprehension
```

```
sum = 0
for char in number_string:
    sum+=(int(char)**3)

# Method 2 - with list comprehension
print(functools.reduce(lambda x,y:x+y,[(int(char)**3) for char in str(number_
```

Out[127…

```
'# This block tests out the logic with 2 methods!\n\n# We take a number and c
onvert it to a string!\nnumber_string = str(371)\n\n# Method 1 - without list
comprehension\nsum = 0\nfor char in number_string:\n    sum+=(int(char)**3)\n
\n# Method 2 - with list comprehension\nprint(functools.reduce(lambda x,y:x+
y,[(int(char)**3) for char in str(number_string)]))'
```

In [128…

```
"""
This function checks whether a number is a Armstrong number or not.
We take a number, and sum up the cube of each digits.
If the sum of cube of digits is equal to the given number, then it is conside
"""
def check_if_is_armstrong_number(given_number):
    # Converting to string. This would allow us to use list comprehension
    number_string=str(given_number)
    sum_of_cubed_digits = functools.reduce(lambda x,y:x+y,[(int(char)**3) for
    return sum_of_cubed_digits == given_number
```

In [129…

```
# Testing the function
print(371," : ",check_if_is_armstrong_number(371))
print(173," : ",check_if_is_armstrong_number(173))
```

```
371  :  True
173  :  False
```

Flatten a nested sequence into a single list of values. Note: the logic should work for any number of nested sequences.

- Input: [1, 2, [3, 4, [5, 6], 7], 8]
- Output: [1, 2, 3, 4, 5, 6, 7, 8]

In [130…

```
# This method will take a nested_list as an input and create a flattened_list
def nested_list_flattener(nested_list,flattened_list=[]):
    for item in nested_list:
        if type(item)==list:
            nested_list_flattener(item,flattened_list)
        else:
            flattened_list.append(item)
    return flattened_list
```

In [131…

```
# Taking a very complex nested-list
nested_list = [1, 2, [3, 4, [5, 6], 7], [8],[9,[10,[11,[12,[13,[14,15,16,17,[

# Displaying the final output of the flattened_list
print(nested_list_flattener(nested_list))
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18]
```

Write a python program to find if a number is Automorphic or not.

Note: A number is Automorphic if and only if its square ends with the same digit as itself. Eg: The number 25 has its square 625 ending with the same last digit.

In [132…

```
# A number is Automorphic if and only if its square ends with the same digit
# Eg: The number 25 has its square 625 ending with the same last digit.
def is_number_automorphic(number):
    return (number%10) == ((number**2)%10)
```

In [133…
```
print(is_number_automorphic(6))
# Six is indeed an automorphic number

# while 4 is not an automorphic number
print(is_number_automorphic(4))
```

```
True
False
```

Write a program to sort the dictionary based on value in reverse order. data = {"maths":50, "physics":90, "chemistry:"80, "english":70, "history":85} Hint: Use itemgetter

In [134…
```
data = {"maths":50, "physics":90, "chemistry":80, "english":70, "history":85}

new_data = {key:value for key,value in sorted(data.items(),key=lambda item:it
print(new_data)
```

```
{'maths': 50, 'english': 70, 'chemistry': 80, 'history': 85, 'physics': 90}
```

Implement a switch case in python to display the following chocolate manufacturing company names when their corresponding chocolates are selected based on their capitals. kitkat - nestle

dairymilk - cadbury

kisses - hershley

raffaello - ferroro

kinderjoy - kinder

Hint : Switch case in python are implemented using dictionaries

In [135…
```
def my_switch_statement(switch_key):
    switch_stmt_from_dict={
        "kitkat"    : "nestle",
        "dairymilk" : "cadbury",
        "kisses"    : "hershley" ,
        "raffaello" : "ferroro",
        "kinderjoy" : "kinder"
    }

    print(switch_stmt_from_dict.get(switch_key,"Please provide a valid select
```

In [136…
```
my_switch_statement("kinderjoy")
```

```
kinder
```

Create a matrix $A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$ and find its transpose using

1. List comprehension
2. Zip

In [137…
```
matrix = [[1,2,3],[4,5,6]]

# 1. Using List-comprehension
transposed_matrix = [[matrix[0][index],matrix[1][index]] for index in range(l
print("1. Using List-comprehension : ", transposed_matrix)

# 2. Using Zip
print("2. Using Zip : ", list(zip(*matrix)))
```

```
1. Using List-comprehension :  [[1, 4], [2, 5], [3, 6]]
2. Using Zip :  [(1, 4), (2, 5), (3, 6)]
```

Write a program to find the largest N items in a list?

- Input: [1, 8, 2, 23, 7, -4, 18, 23, 42, 37, 2] and 3 (N)

- Output: [42, 37, 23]

In [138…
```python
def remove_n_largest_elems_from_list(target_list,count_of_elems):
    new_list=[]
    for count in range(count_of_elems):
        new_list.append(max(target_list))
        target_list.remove(max(target_list))
    return new_list

input_list=[1, 8, 2, 23, 7, -4, 18, 23, 42, 37, 2]
input_count=3
print(remove_n_largest_elems_from_list(input_list,input_count))
```

```
[42, 37, 23]
```

Write a sigmoid function which takes a matrix (x) as input and outputs a matrix. The formula for sigmoid function is $\frac{1}{1+e^{-x}}$

$$X = \begin{bmatrix} 10 & 20 & 30 \\ 4 & 5 & 6 \end{bmatrix}$$

Note: Please DO NOT use numpy.

In [139…
```python
def my_sigmoid_function(number):
    return (1.0/(1+(math.e**(-1*number))))
```

In [140…
```python
def apply_sigmoid_funtion_to_matrix(given_matrix):
    return [list(map(my_sigmoid_function,row)) for row in given_matrix]
```

In [141…
```python
test_matrix=[[10,20,30],[4,5,6]]
print(apply_sigmoid_funtion_to_matrix(test_matrix))
```

```
[[0.9999546021312976, 0.9999999979388463, 0.9999999999999065], [0.98201379003
79085, 0.9933071490757153, 0.9975273768433653]]
```

Simulate a fast-food ordering scenario by defining four classes

- Lunch: A container and controller class
- Customer: The actor who buys food
- Employee: The actor from whom a customer orders
- Food: What the customer buys

  The order simulation should work as follows

1.The Lunch class's constructor should make and embed an instance of Customer and an instance of Employee, and it should export a method called order. When called, this order method should ask the Customer to place an order by calling its place_order method. The Customer's place_order method should in turn ask the Employee object for a new Food object by calling Employee's take_order method.
2.Food objects should store a food name string (e.g., "dosa"), passed down from Lunch.order, to Customer.place_order, to Employee.take_order, and finally to Food's constructor. The top-level Lunch class should also export a method called result, which asks the customer to print the name of the food it received from the Employee via the order (this can be used to test your simulation).

In [142…
```python
class Food():
    def __init__(self,food_name):
        self.food_name=food_name
```

In [143…
```
class Employee():
    def __init__(self):
        self.food = None

    def take_order(self,food_name):
        self.food = Food(food_name)

    def set_name(self,emp_name):
        self.name=emp_name
```

In [144…
```
class Customer():

    def place_order(self,employee,food_name):
        self.food_name=food_name
        employee.take_order(self.food_name)

    def print_food_received(self,employee):
        print(f"{self.food_name} delivered by {employee.name}")
```

In [145…
```
class Lunch():
    def __init__(self):
        self.customer = Customer()
        self.employee = Employee()

    def order(self,food_name,employee_name):
        self.employee.set_name(employee_name)
        self.customer.place_order(self.employee,food_name)

    def result(self):
        self.customer.print_food_received(self.employee)
```

In [146…
```
my_lunch=Lunch()
```

In [147…
```
my_lunch.order("Dosa","Mihir")
```

In [148…
```
my_lunch.result()
```

```
Dosa delivered by Mihir
```

In [149…
```
my_lunch.order("Cake","Mihir")
```

In [150…
```
my_lunch.result()
```

```
Cake delivered by Mihir
```