Activity 1.1   Python Challenge

Name: Muyo, Mark Danielle L.

Course and Section: CPE31S9

Date Submitted: January 24, 2024

Date Performed: January 24, 2024

Instructor: Engr. Roman Richard

Instructions: Answer the questions or complete the tasks outlined below, use the specific method described if applicable. Insert a code cell after each text cell.

Objectives:

- Test understanding of Python basics
- Review Python basic to intermediate commands

1. What is 3 to the power of 5?

```
3**5
```

```
243
```

2. Create a variable, s, containing the string "This course is amazing!". Using the variable, split the string into a list.

```
s = "This course is amazing!"
split_list = s.split()
print(split_list)
```

```
['This', 'course', 'is', 'amazing!']
```

3. Given the variables height and mountain, use .format() to print the following string: `The height of Mt. Everest is 8848 meters.`

```
height = 8848
mountain = "Mt. Everest"
print("The height of {} is {} meters.".format(mountain, height))
```

```
The height of Mt. Everest is 8848 meters.
```

4. Given the following nested list, use indexing to grab the word "this".

```
lst = ['a','b',[4,10,11],['c',[1,66,['this']],2,111],'e',7]
```

```
lst = ['a','b',[4,10,11],['c',[1,66,['this']],2,111],'e',7]
word = lst[3][1][2][0]
print(word)
```

```
this
```

5. Given the following nested dictionary, grab the word "that". This exercise is a little more difficult.

```
d = {'k1':['val1','val2','val3',{'we':['need','to','go',{'deeper':[1,2,3,'that']}]}]}
```

```
d = {'k1':['val1','val2','val3',{'we':['need','to','go',{'deeper':[1,2,3,'that']}]}]}
d ['k1'][3]['we'][3]['deeper'][3]
```

```
'that'
```

6. Create a function, GetDomain(), that grabs the email website domain from a string in the form: user@domain.com. So for example, passing "user@domain.com" would return: domain.com

```
def GetDomain(email):
    domain = email.split('@')[1]
    return domain

email = "user@domain.com"
domain = GetDomain(email)
print(domain)
```

```
    domain.com
```

7. Create a basic function, findInternet(), that returns True if the word 'Internet' is contained in the input string. Don't worry about edge cases like punctuation being attached to the word, but account for capitalization. (Hint: Please see https://docs.python.org/2/reference/expressions.html#in)

```
def findInternet(input_string):
    return 'Internet' in input_string

input_string = "The Internet is not accessible to everyone."
contains_internet = findInternet(input_string)
print(contains_internet)
```

➦ True

— + Code — + Text —

8. Create a function, countIoT(), that counts the number of times the word "IoT" occurs in a string. Ignore edge cases but take into account capitalization.

```
def countIoT(input_string):
    count = input_string.count("IoT")
    return count

input_string = "IoT devices are becoming more popular, and IoT is relevant to data."
count = countIoT(input_string)
print(count)
```

```
    2
```

9. Use lambda expressions and the filter() function to filter out words from a list that do not start with the letter 'd'. For example:

```
    seq = ['data','salt' ,'dairy','cat', 'dog']
```

should be filtered down to:

```
    ['data', 'dairy', 'dog']
```

```
seq = ['data','salt','dairy','cat','dog']
lowercase_seq = list(filter(lambda word: word.startswith('d'), seq))
print(lowercase_seq)
```

```
    ['data', 'dairy', 'dog']
```

10. Use lambda expressions and the map() function to convert a list of words to upper case. For example:

```
    seq = ['data','salt' ,'dairy','cat', 'dog']
```

should become:

```
    ['DATA', 'SALT', 'DAIRY', 'CAT', 'DOG']
```

```
seq = ['data', 'salt', 'dairy', 'cat', 'dog']
uppercase_seq = list(map(lambda word: word.upper(), seq))
print(uppercase_seq)
```

```
['DATA', 'SALT', 'DAIRY', 'CAT', 'DOG']
```

11. Imagine a smart thermostat that is connected to the door, so that it can detect, in addition to the temperature, when people enter or leave the house.

Write a function that, if the temperature is lower than 20 degrees Celsius, and there are people in the house (encoded as a boolean value to be passed as a parameter to the function), turns on the heating by returning the string "Heating on". When the temperature reaches 23 degrees or there are no people in the house, it returns the string "Heating off". When none of these conditions are met, the function returns "Do nothing".

```
def controlHeating(temperature, people_present):
    if temperature < 20 and people_present:
        return "Heating on"
    elif temperature >= 23 or not people_present:
        return "Heating off"
    else:
        return "Do nothing"
```

```
temperature = 15
people_present = True
heating_status = controlHeating(temperature, people_present)
print(heating_status)
```

```
Heating on
```

12. The function zip(list1, list2) returns a list of tuples, where the i-th tuple contains the i-th element from each of the argument lists. Use the zip function to create the following list of tuples:

```
zipped = [('Parking', -1), ('Shops',0), ('Food Court',1), ('Offices',2)]
```

```
list1 = ['Parking', 'Shops', 'Food Court', 'Offices']
list2 = [-1, 0, 1, 2]
zipped = list(zip(list1, list2))
print(zipped)
```

```
[('Parking', -1), ('Shops', 0), ('Food Court', 1), ('Offices', 2)]
```

13. Use the zip function and dict() to create a dictionary, elevator_dict, where the keys are the floor types and the values are the corresponding floor number so that:

```
elevator_dict[-1] = 'Parking'
```

```
list1 = ['Parking', 'Shops', 'Food Court', 'Offices']
list2 = [-1, 0, 1, 2]
elevator_dict = dict(zip(list2, list1))
print(elevator_dict)
```

```
{-1: 'Parking', 0: 'Shops', 1: 'Food Court', 2: 'Offices'}
```

```
elevator_dict[-1] = 'Parking'
print(elevator_dict)
```

```
{-1: 'Parking', 0: 'Shops', 1: 'Food Court', 2: 'Offices'}
```

14. Create an Elevator class. The constructor accepts the list of strings floor_types and the list of integers floor_numbers. The class implements the methods 'ask_which_floor' and 'go_to_floor'. The output of this methods should look as follows:

```
floor_types = ['Parking', 'Shops', 'Food Court', 'Offices']
floors_numbers = range(-1,4)
el = Elevator(floor_numbers, floor_types)
el.go_to_floor(1)
Going to Food Court floor!
el.go_to_floor(-2)
There is floor number -2 in this building.
el.ask_which_floor('Offices')
The floor Offices is the number: 2
el.ask_which_floor('Swimming Pool')
There is no Swimming Pool floor in this building.
```

```python
class Elevator:
    def __init__(self, floor_numbers, floor_types):
        self.floor_numbers = floor_numbers
        self.floor_types = floor_types

    def go_to_floor(self, floor_number):
        if floor_number in self.floor_numbers:
            floor_type = self.floor_types[floor_number]
            print(f"Going to {floor_type} floor!")
        else:
            print(f"There is no floor number {floor_number} in this building.")

    def ask_which_floor(self, floor_type):
        if floor_type in self.floor_types:
            floor_number = self.floor_numbers[self.floor_types.index(floor_type)]
            print(f"The floor {floor_type} is the number: {floor_number}")
        else:
            print(f"There is no {floor_type} floor in this building.")

floor_types = ['Parking', 'Shops', 'Food Court', 'Offices']
floor_numbers = list(range(-1, 4))
el = Elevator(floor_numbers, floor_types)

el.go_to_floor(1)
el.go_to_floor(-2)
el.ask_which_floor('Offices')
el.ask_which_floor('Swimming Pool')
```

```
    Going to Shops floor!
    There is no floor number -2 in this building.
    The floor Offices is the number: 2
    There is no Swimming Pool floor in this building.
```

Good Job!