Portfolio

Name: Mohamed Tah

ID: M01056264

Program: Data Science & A.I

Introduction:

This portfolio will seek to demonstrate my progress and ability to write coherent python syntax to solve problems of varying degrees of difficulty.

Topics Covered:

- Variables, IO and Sys Info (week1)
- Control structures, casting and randomly generated numbers (week2)
- Loops and Lists (week3)
- Functions and strings (week4)
- Files IO (week5)
- Conclusion / summary

Progression of Learning:

In my experience the complexity increase week by week was reasonably gradual, we started with the very basics (variables, input/output etc.…) and moved on to control structures and lists.

Every week we covered an interesting topic in a thorough manner, and I found it interesting even though am very familiar with python, I've been using it for the past 5 years and currently the backend of my startup is built using Django which is a python framework.

If I were a beginner I would have paid more attention to the progression and could have provided an accurate progression of learning statement.

Skills Developed:

The main skill I developed throughout these couple of weeks that I did not previously possess is commenting, I never got into the habit of making comments because am so used to all the changes to a project being summarized in a git commit comment. But I no longer overlook comments in fact I've been using them religiously in all my work, personal and academic work.

Overall:

Overall my understanding of python has not changed, am very familiar with all the concepts covered as I've used them almost on a daily basis for the past five years working on projects that never truly saw the day of light (some did on my GitHub) but most of my startups failed namely (Voidback which is my 2024 startup powered by Django rest framework and nextjs) every project I've worked on I learned something new and I feel it bring me closer to my desired destination.

1) **Week 1 - (Variables, IO & Sys Info):**

   a. I was asked to ask the user for a hostname and print their operating system and python version as a simple banner.

   b.

```
import platform


hostname = input("Enter hostname: ")
osys = platform.system()
py_v = platform.python_version()

print(f"Host: {hostname}\nOS: {osys} \nPython: {py_v}")
```

```
Host: mo
OS: Darwin
Python: 3.13.3
```

   c. **Firstly,** I've imported the "platform" module which provides me with the necessary methods to complete this task.

   Secondly, I created a variable named "hostname" which will contain the host name entered by the user via the prompt made using the input() function (which is a built in function from the python standard library), furthermore, I've created 2 additional variables: "**osys**" which holds the operating system name and "**py_v**" which holds the python version both of which obtained through the platform module.

   Lastly, I neatly printed the simple banner using the print() function (builtin function from the python standard library)

d. I learned that the platform module is handy when it comes to system related information that could be helpful down the line.

2) Week 2 – (Control structures, casting and randomly generated numbers)

a. I was asked to write a program that randomly generates a 2 digit number, and asks the user to enter a 2 digit number to determine wither the user wins according to a given set of rules.

b.

```
lottery = str(randint(0, 99))


guess = input("enter a two digit number: ")


if guess == lottery:
    print("You won £10,000")

else:
    matched = 0
    for i in lottery:
        if i in guess:
            matched += 1

    if matched == 2:
        print("You won £3,000")

    elif matched == 1:
        print("You won £1,000")
```

```
You won £3,000
```

c. I've imported the "randint" function from the "random" module in a preceding cell which allows me to generate a random integer within a given range.
I then defined a few variables, namely lottery which holds the str version of the randomly generated int (using the str method to cast the int into a str). And then I asked the user to enter a 2-digit number and used a simple if statement to check if the 2-digit number is equal to the randomly generated 2-digit number stored in the "lottery" variable. Otherwise, we initialize a new variable called matched with zero and we iterate through the random 2-digit str ("lottery") and check if one of those digits is in the ("guess") string if so then we increment the matched variable. If we have 2 matched digits, then the user did guess it correctly but in the wrong (inverse) order otherwise the user only guessed one digit correctly.

d. I learned that the randint function is extremely useful for generating random numbers within a given range.

3) Week 3 – (Loops and lists)

a. I was asked to write a program that reads numbers from the user and stores them in a list and prints them in reverse order

b.

```
nums = []

while True:
    num = input("enter an integer (enter 0 to quit): ")

    nums.append(int(num))

    if num=="0":
        break

nums.pop() # remove zero

nums.reverse() # reverse nums

nums # referencing variables on the last line of a cell is equivalent to printing them
```

```
[40, 30, 20, 10]
```

c. I defined a variable called nums initialized with an empty list that will hold the numbers entered by the user, I then entered a while loop whereby I ask the user to enter a number that's not zero (sentinel value) and I then casted that number to type Int and appended it to the nums list. If the user enters 0 (the sentinel value) I break out of the loop.
I poped the last element of nums (which is the sentinel value) and used the list.reverse() method to reverse the nums list and printed the list in reverse order.

d. I learned that the reverse method can come in really handy in certain situations where the order of a sequence need to be reversed.

4) **Week 4 – (Strings and functions)**

a. I was asked to create a function that essentially validates an mdx student's id.

b.

```python
def checkStudent(idnum: str):
    if idnum[0:3] == "M00" and len(idnum) == 9:
        print("Valid id")

    else:
        print("Invalid id")


checkStudent("MS)Dk23")
checkStudent("M00493201")
```

c. I defined a function with one string parameter and checked the first 3 characters to make sure they are equal to "M00" and then I check the length of the string to make sure its 9 characters long and printed "valid id" or "invalid id" depending on the condition.

d. I learned that functions are extremely useful and allow for our code to be DRY (Don't Repeat Yourself) this form of compartmentalizing certain procedures allows us to reuse them and reduces overall redundancy in our programs.

5) Week 5 – (Files IO (week5)

a. I was asked to write a program that writes a shopping list to a text file and reads it back and prints to the user each item emphasizing the need to buy it and finally printing the count (total items).

b.

```
with open("shopping.txt", "w") as f:
    f.writelines(["Apple\n", "Orange\n", "Sugar\n", "Water\n", "Soda\n"])


with open("shopping.txt", "r") as f:
    items = f.readlines()
    for item in items:
        print(f"You need to buy: {item}")

    print(f"Total items: {len(items)}")
```

```
You need to buy: Apple

You need to buy: Orange

You need to buy: Sugar

You need to buy: Water

You need to buy: Soda

Total items: 5
```

c. Firstly, I opened the file in write mode, and used the writelines method to write each item in the list as a line of course I used the end of line character to make sure all items are line separated. Secondly, I reopened the file in reading mode and used the readlines method which will return a list exactly like the one we passed to the writelines method. Lastly, I iterated through the items using a for loop and

printed each one telling the user they need to buy it and then I printed the total items in the list.

d. I learned that file io operations are very useful and allow us to have this persistent storage system where if the user were to run the program again the list is still available on their hard drive.

Conclusion:

learned several **useful Python concepts** that can make programming much more effective.

I now know that the **platform module** is great for getting system information, and the **randint function** is handy for creating random numbers within a specific range.

I also discovered how the **reverse method** can quickly change the order of a sequence. Most importantly, I realized that using **functions** makes code much better it keeps the code **DRY (Don't Repeat Yourself)** by allowing me to reuse procedures and avoid redundancy.

Finally, I found that **file I/O operations** are essential for creating a **persistent storage system**, which means data, like a list, is saved to the hard drive and is still available the next time the program runs.