Python-Refresher-Notes

1. {} Format

```
name = input("Enter your name: ")
color = input("Enter your favorite colour: ")
hobby = input("Enter your favorite hobby: ")

print("Hello {}! Nice to meet you.".format(name))
print("")
print("I see that your favourite colour is {}, and your favourite hobby is
{}. It's great to learn more about you.".format(color, hobby))
```

Output

```
Enter your name: Jack
Enter your favorite colour: Teal
Enter your favorite hobby: Painting
```

```
Hello Jack! Nice to meet you.

I see that your favourite colour is Teal, and your favourite hobby is
Painting. It's great to learn more about you.
```

2. {} Format 2

```
pat_name = input("Enter the patient name: ")
illness = input("Illness: ")
admission_fee = float(input("Admission fee: "))
doctor_fee = float(input("Doctor fee: "))
nursing_fee = float(input("Nursing fee: "))
medicine_cost = float(input("Cost for medicines: "))

print("AUMA Hospital\n")
```

```
print("Patient Name: {}\n".format(pat_name))
print("Illness: {}\n".format(illness))
print("Admission fee: ${:.2f}\n".format(admission_fee))
print("Doctor fee: ${:.2f}\n".format(doctor_fee))
print("Nursing fee: ${:.2f}\n".format(nursing_fee))
print("Cost for medicines: ${:.2f}\n".format(medicine_cost))
print("Total amount: ${:.2f}\n".format(admission_fee + doctor_fee +
nursing_fee + medicine_cost))
```

- {:.2f} — Formatted Placeholder
  - This is used to format floating-point numbers to a specific number of decimal places.
  - :.2f means:
    - : → start of format spec
    - .2 → two digits after the decimal point
    - f → fixed-point notation (i.e., decimal number)

3. Lambda function

Before: function

```
def get_percentage(sub1, sub2, sub3):
     return ((sub1 + sub2 + sub3) / 300) * 100
```

How to change to lambda

- To turn it into lamba, do the following
  - Step 1:
    - Remove def
    - Remove brackets of parameters
    - remove return
    - We get
      - get_percentage sub1,sub2,sub3 : ((sub1,sub2,sub3)/300) * 100
  - Step 2:

- Add `=` and lambda after function name
- get_percentage = lambda sub1,sub2,sub3 : ((sub1 + sub2 + sub3) / 300) * 100

Lambda function

```
get_percentage = lambda sub1,sub2,sub3 : ((sub1 + sub2 + sub3) / 300) * 100
```

4. Text Files

Basic Overview of File Handling in Python

Opening a File

Python uses the built-in `open()` function:

```
file = open("filename.txt", "mode")
```

- `"filename.txt"` : Name of the file
- `"mode"` : Specifies the operation (read, write, etc.)

File Modes

Mode	Description
`'r'`	Read (default). Error if file doesn't exist.
`'w'`	Write. Creates file if not exists, overwrites if it does.
`'a'`	Append. Adds content to the end of the file.
`'x'`	Create. Error if file already exists.
`'b'`	Binary mode (e.g., `'rb'`, `'wb'`)
`'t'`	Text mode (default)
`'+'`	Read and write (e.g., `'r+'`, `'w+'`)

Common File Handling Functions

Once a file is opened, you can use:

Reading

```
file.read()         # Reads entire file
file.readline()     # Reads one line
file.readlines()    # Reads all lines into a list
```

Writing

```
file.write("text")  # Writes text to file
file.writelines([...])  # Writes list of strings
```

Other Operations

```
file.seek(offset)   # Moves cursor to specified position
file.tell()         # Returns current cursor position
file.close()        # Closes the file
```

Using `with` Statement (Best Practice)

```
with open("filename.txt", "r") as file:
    content = file.read()
```

Example: Writing and Reading a File

```
# Writing
with open("example.txt", "w") as f:
    f.write("Hello, world!")

# Reading
with open("example.txt", "r") as f:
    print(f.read())
```

5. CSV Files

What Does a CSV File Look Like?

```
Name,Age,City
Alice,30,New York
Bob,25,Los Angeles
Charlie,35,Chicago
```

Each line is a **record**, and each value is separated by a **comma**.

Working with CSV in Python

Python provides a built-in module called `csv` to handle CSV files.

Reading a CSV File

```python
import csv

with open('data.csv', 'r') as file:
    reader = csv.reader(file)
    for row in reader:
        print(row)
```

Writing to a CSV File

```python
import csv

with open('output.csv', 'w', newline='') as file:
    writer = csv.writer(file)
    writer.writerow(['Name', 'Age', 'City'])
    writer.writerow(['Alice', 30, 'New York'])
```

Using `DictReader` and `DictWriter`

These allow you to work with CSV data as dictionaries:

Reading:

```
with open('data.csv', 'r') as file:
    reader = csv.DictReader(file)
    for row in reader:
        print(row['Name'], row['Age'])
```

Writing:

```
with open('output.csv', 'w', newline='') as file:
    fieldnames = ['Name', 'Age', 'City']
    writer = csv.DictWriter(file, fieldnames=fieldnames)
    writer.writeheader()
    writer.writerow({'Name': 'Bob', 'Age': 25, 'City': 'LA'})
```

6. JSON files

What is JSON?

JSON (JavaScript Object Notation) is a lightweight data-interchange format that's easy for humans to read and write, and easy for machines to parse and generate. It's commonly used for APIs, configuration files, and data storage.

- 🔹 **Importing the Module**

```
import json
```

Reading JSON

1. From a JSON string

```
json_string = '{"name": "Alice", "age": 30}'
data = json.loads(json_string)
print(data["name"])  # Output: Alice
```

2. From a JSON file

```
with open('data.json', 'r') as file:
    data = json.load(file)
```

🔹 **Writing JSON**

1. To a JSON string

```
person = {"name": "Bob", "age": 25}
json_string = json.dumps(person)
print(json_string)
```

2. To a JSON file

```
with open('output.json', 'w') as file:
    json.dump(person, file)
```

Formatting Options with `dumps()`

```
json.dumps(person, indent=4)       # Pretty print with indentation
json.dumps(person, sort_keys=True) # Sort keys alphabetically
```

Common Use Cases

- Parsing API responses
- Saving configuration settings
- Logging structured data
- Data exchange between systems

7. Classes in python - Example 1

Here, classes serve as blueprints to define the structure and behaviour of vehicles, while objects, instantiated from these classes, encapsulate specific vehicle data and operations, facilitating the management of vehicle information and premium calculations.

```
class Vehicle:
    def __init__(self, cost, vehicle_type):
        self.__cost = cost
```

```
        self.__type = vehicle_type
        self.__premium = 0

    def calculate_premium(self):
        if self.__type == 1:
            premium_amount = (self.__cost) * 0.02
            self.__premium = premium_amount
        elif self.__type == 2:
            premium_amount = (self.__cost) * 0.06
            self.__premium = premium_amount
        elif self.__type == 3:
            premium_amount = (self.__cost) * 0.08
            self.__premium = premium_amount

    def get_premium(self):
        return self.__premium


vehicle_cost = float(input("Enter the vehicle cost: "))
vehicle_type = int(input("Enter the type of the vehicle (1 for 2 wheeler, 2
for 4 wheeler, 3 for other types): "))

vehicle_obj = Vehicle(vehicle_cost, vehicle_type)
vehicle_obj.calculate_premium()

vehicle_premium_amount = vehicle_obj.get_premium()
print("The premium amount is: {}".format(vehicle_premium_amount))
```

8. Classes in Python - Example 2

```
class Addition:
    def __init__(self):
        self.__real = 0
        self.__img = 0

    def get_real(self):
        return self.__real
```

```
    def set_real(self, real):
        self.__real = real

    def get_img(self):
        return self.__img

    def set_img(self, img):
        self.__img = img

    def addRealPart(self, obj1, obj2):
        real_sum = obj1.get_real() + obj2.get_real()
        self.set_real(real_sum)

    def addImaginaryPart(self, obj1, obj2):
        img_sum = obj1.get_img() + obj2.get_img()
        self.set_img(img_sum)

    def display(self):
        print(f"Sum: {self.__real} + {self.__img}i")


# Example usage
obj1 = Addition()
obj1.set_real(3)
obj1.set_img(4)

obj2 = Addition()
obj2.set_real(5)
obj2.set_img(6)

result = Addition()
result.addRealPart(obj1, obj2)
result.addImaginaryPart(obj1, obj2)
result.display()
```