1 Python Classes Summary 1

1.1 Comments

Comments are started with hashtag (#) symbols and cause following text on the same line to be ignored by python. This allows the programmer to add explanations / notes to their code such that other programmers can understand it more easily. (Comments also serve as useful reminders for the programmers themselves)

```
Eg [1]: # Hashtags denote comments - These are ignored by python
         # It is good practice to use comments to explain what the code is doing
```

1.2 Taking user input

Sometimes we want the user to give us some information, which we then use somehow in our code. Luckily, python has a built in *function* (aka method) which easily allows us to do this:

```
Eg [2]: # This code will compliment our human user

        name_of_user = input("Hi! What is your name? ") # Get user input
        output_string = "Nice to meet you " + name_of_user + " - Looking good! ;)"
        print(output_string) # Print our compliment string
```

```
Hi! What is your name? Alex
Nice to meet you Alex - Looking good! ;)
```

1.3 Logic

Mathematical logic essentially allows us to compare things in a well-defined, consistent way. The ability to do this is essential for allowing us to build useful computer programmes. The types of comparison operations that we can perform in python are outlines in the following table:

operator	Description	Example	Output
<	less than	5 < 8	False
>	greater than	4.3 > 2	True
==	equals	"Hello" == "there"	False
<=	less than or equal to	5 <= 5	True
>=	greater than or equal to	8 >= 110	False
!=	not equal	"General" != "Kenobi"	True

```
Eg [3]: # An example of using logic

        is_this_true = 3 > 10 # Let us see if 3 is > 10; The result should be false
        print(is_this_true) # Print out the result, hopefully it will say false
```

```
False
```

1.4 Control - If Statements

Even with just these few logical comparisons, we can get our code to perform arbitrarily complex tasks. There are numerous ways of using logic in python, but so far we've only covered if statements. These allow us to only execute code when a chosen condition is satisfied.

```
Eg [4]: # Example of simple if-else

        user_answer = int(input("What is 7 + 3? ")) # NOTE convert string to int

        if (user_answer == 10):
            print("Check you out!")
        else: # Catch all inputs which aren't 10
            print("Happens to the best of us") # Lie to the user
```

```
What is 7 + 3? 10
Check you out!
```

In the script above, we used an "if-else" statement, but in some cases we might want to multiple possible specific answers differently. For this we can use "elif" (short for else if) :

```
Eg [5]: # Example of if statement with else-if

        user_name = "Augustus" # We don't need humans!

        if (user_name == "Alex"):
            print("I've been expecting you...")
        elif (user_name == "Augustus"):
            print("Ad omnes Ave Caesar")
        else:
            print("Nice to meet you " + user_name)
```

```
Ad omnes Ave Caesar
```

1.4.1 Extra Fun

There are many ways we can improve our handling of user inputs:

```
Eg [6]: user_name = input("What's your name? ")
        user_name = user_name.lower() # Make the string lowercase
        # Note, we just overwrote the variable!

        if (user_name in ['alex', 'luke']): # Can check multiple cases in one
            print("Your Majesty...")
        elif ("z" in user_name): # Can check if a substring is in the name!
            print("Funky name!")
        else:
            print("Heyo")
```