#Python Basics

1. What is Python and why it is popular?

-> Python is a programming language and it is popular due to a) Ease of learning b) It's versatility c) Easy to interpret d) Extensive libraries e) Active community

1. What is an interpreter in Python?

-> Interpreter in python is a program that executes a code line by line instead of executing all the code at once (as compiler does compiler)

1. What are pre-defined keywords in Python?

-> Pre-defined keywords are reserved words in python that have a special meaning and purpose. Ex: if,else,elif etc.

1. Can keywords be used as variable names?

-> No keywords cannot be used as variable names

1. What is mutability in Python?

-> Mutability is the ability of the object to be changed after it has been created. Ex: List

1. Why are lists mutable, but tuples are immutable?

-> Lists are mutable because they are designed for dynamic uses ie) Adding, Updating or Deleting the elements. Tuples are immutable to insure data integrity while working with fixed objects.

1. What is the difference between “==” and “is” operators in Python?

-> "==" checks if both the values of the objects are same. Ex. a = [1,2,3] b = [1,2,3] so if we do print(a==b) it will return true as both the values are same. -> "is" checks if both objects refer to same memory location. Ex. a = [1,2,3] b = [1,2,3] c = a . If we do print(a is b) we will get False as both are different objects in memory but if we do print (a is c) we will get True as both are refering to same object in memory.

1. What are logical operators in Python?

-> Logical operators (and,or,not) are used to combine conditional statement and return boolean value (True or False) Ex: a = 10 b = 20 print (a < b and b > a). This will give True as both the conditions are satisfied

1. What is type casting in Python?

-> Type Casting is the process of converting one data type to another data type by using fucntions like int(), str() , float() etc.

1. What is the difference between implicit and explicit type casting?

-> Implicit type casting is when python automatically converts one data type to another whenever it is required. Ex. a=10(integer) b=2.0(float) print (a+b) which will give 12.0(float).

-> Explicit type casting is when the programmer converts one data type to another by himself. Ex a=2.3 b=int(a) print(b) will give 2(integer) as we have converted the float to integer

1. What is the purpose of conditional statements in Python?

-> Conditional statements in python are used to execute different set of codes depending on the conditions satisfied.

1. How does the elif statement work?

-> Elif statement works such that it check multiple conditions in a sequence. If the first condition (if) fails it checks the next condtion (elif). If the elif condition fails it checks the next condition elif or else untill the condition is satisfied . If condition is not satisfied then it returns the value of (else). Ex. x = 10 if (x<5): print("x is less than 5") elif (x == 10): print("x is equal to 10) else: print("x is greater than 10")

-> Here x is equal to 10 will be given as output.

1. What is the difference between for and while loops?

-> For loop is used when the number of iterations is known. It also iterates through the items in a sequence -> While loop is preferred when the number of iterations are unknown. It continues untill the condition becomes False.

1. Describe a scenario where a while loop is more suitable than a for loop.

-> When we want to guess a password. password = "" while (password!= "pywalla"): input("Enter password:") print("Correct password")

Here we don't know how many iterations will the user take to guess the password. So using a while loop will run untill the condition is satisfied.

Practical Questions

1. Write a Python program to print "Hello, World!"

```
print("Hello, World!")

Hello, World!
```

1. Write a Python program that displays your name and age0

```
print("Sai")
print("21 years old")

Sai
21 years old
```

1. Write code to print all the pre-defined keywords in Python using the keyword library

```
import keyword
```

```
for i in keyword.kwlist:
  print(i)
```

```
False
None
True
and
as
assert
async
await
break
class
continue
def
del
elif
else
except
finally
for
from
global
if
import
in
is
lambda
nonlocal
not
or
pass
raise
return
try
while
with
yield
```

1. Write a program that checks if a given word is a Python keyword

```
import keyword

word = input("Enter word: ")

if (word not in keyword.kwlist):
  print(word,"is not a keyword")
else:
  print(word,"is a keyword")
```

```
Enter word: None
None is a keyword
```

1. Create a list and tuple in Python, and demonstrate how attempting to change an element works differently

```
list_1 = [1,2,3,4]
tuple_1 = (1,2,3)

list_1[1] = 42 ## able to change the 2nd element to 42
print(list_1)

tuple_1[1] = 32 ## getting error 'tuple' object does not support item
assignment
print(tuple_1)

[1, 42, 3, 4]

----------------------------------------------------------------------
-----
TypeError                                 Traceback (most recent call
last)
<ipython-input-11-0eca746051cb> in <cell line: 7>()
      5 print(list_1)
      6 
----> 7 tuple_1[1] = 32
      8 print(tuple_1)

TypeError: 'tuple' object does not support item assignment
```

1. Write a function to demonstrate the behavior of mutable and immutable arguments

```
def modify_args(mutable, immutable):
    # Modifying the mutable argument (list)
    mutable.append(4)

    # Modifying the immutable argument (integer)
    immutable += 5

# Initial values
list_1 = [1, 2, 3]  # Mutable (list)
num = 10       # Immutable (integer)

# Before calling the function
print("Before function")
print("Mutable:", list_1)
print("Immutable:", num)

# Function call
modify_args(list_1, num)
```

```
# After calling the function
print("After function")
print("Mutable:", list_1)
print("Immutable:", num)

Before function
Mutable: [1, 2, 3]
Immutable: 10
After function
Mutable: [1, 2, 3, 4]
Immutable: 10
```

1. Write a function to demonstrate the behavior of mutable and immutable arguments

```
def modify_args(mutable, immutable):
    # Modifying the mutable argument (list)
    mutable.append(4)

    # Modifying the immutable argument (integer)
    immutable += 5

# Initial values
list_1 = [1, 2, 3]  # Mutable (list)
num = 10       # Immutable (integer)

# Before calling the function
print("Before function")
print("Mutable:", list_1)
print("Immutable:", num)

# Function call
modify_args(list_1, num)

# After calling the function
print("After function")
print("Mutable:", list_1)
print("Immutable:", num)

Before function
Mutable: [1, 2, 3]
Immutable: 10
After function
Mutable: [1, 2, 3, 4]
Immutable: 10
```

1. Write a program to demonstrate the use of logical operators

```
x = 10
y = 20
z = 30
```

```
if x < y and y < z:
    print("Both conditions are True")

if x > y or y < z:
    print("At least one condition is True")

if not(x > z): # as x=10 is smaller than z=30 we use not to make it false thus making the condition true
    print("Condition x > z is False ")

Both conditions are True
At least one condition is True
Condition x > z is False
```

1. Write a Python program to convert user input from string to integer, float, and boolean types.

```
input = "234"

print(int(input))
print(float(input))

234
234.0
```

1. Write code to demonstrate type casting with list elements.

```
list_1 = ['12','2.5','true']

print("Before type casting: ", list_1)

list_1[0] = int(list_1[0])
list_1[1] = float(list_1[1])
list_1[2] = bool(list_1[2])

print("After type casting: ", list_1)

Before type casting:  ['12', '2.5', 'true']
After type casting:  [12, 2.5, True]
```

1. Write a program that checks if a number is positive, negative, or zero.

```
num = int(input("Enter number: "))

if num > 0:
    print(num, "is positive")
elif num < 0:
    print(num, "is negative")
else:
    print("Zero")
```

```
Enter number: 0
Zero
```

1. Write a for loop to print numbers from 1 to 10.

```
for i in range(1,11):
  print(i)

1
2
3
4
5
6
7
8
9
10
```

1. Write a Python program to find the sum of all even numbers between 1 and 50.

```
sum_of_even = 0

for i in range(1,51):
  if i%2 == 0:
    sum_of_even += i

print(sum_of_even)

650
```

1. Write a program to reverse a string using a while loop.

```
string_normal = input("Enter string: ")
string_rev = ""

index = 0

while index < len(string_normal):
  string_rev = string_normal[index] + string_rev
  index += 1

print(string_rev)


Enter string: cloud
duolc
```

1. Write a Python program to calculate the factorial of a number provided by the user using a while loop.

```python
number_1 = int(input("Enter a number: "))
factorial = 1

while number_1 > 0:
    factorial *= number_1
    number_1 -= 1

print("Factorial:", factorial)
```

```
Enter a number: 5
Factorial: 120
```