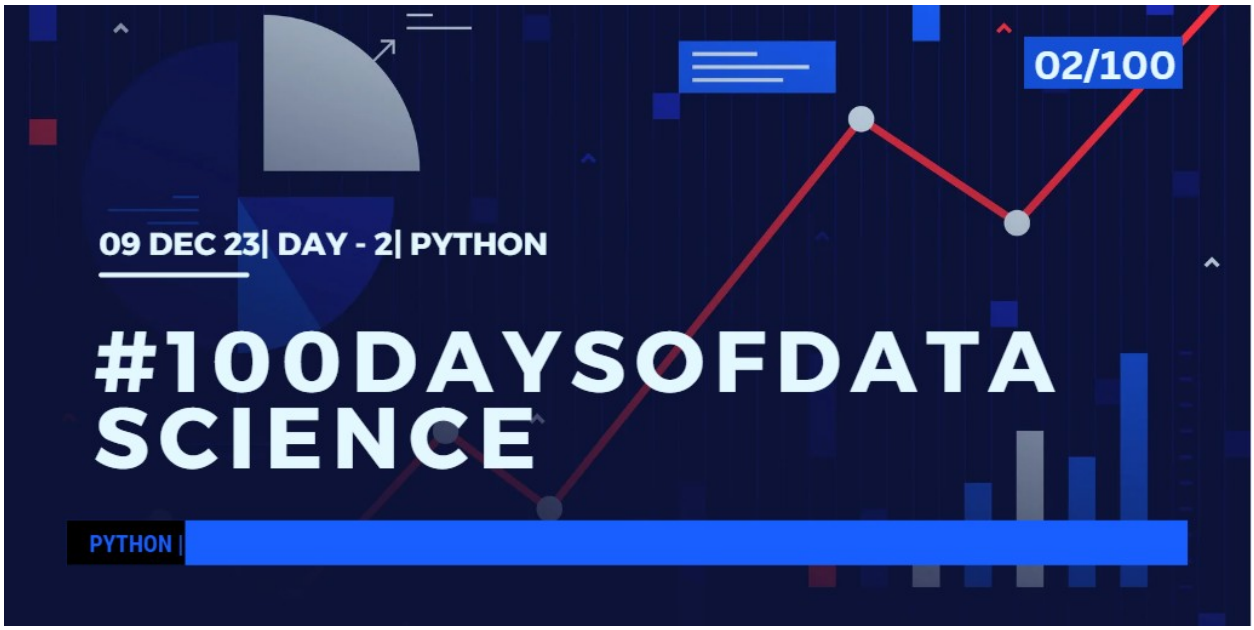

LOOPS IN PYTHON

Loops in Python are fundamental programming constructs that allow you to execute a block of code repeatedly until a certain condition is met. They play a crucial role in automating repetitive tasks and handling sequences of data. Python offers two primary types of loops:

1. FOR LOOP
2. WHILE LOOP

For Loop

A for loop is a control flow statement that allows you to iterate over a sequence of items. The sequence can be a list, tuple, or string. The for loop will execute the code block for each item in the sequence.

```
x = [1,2,3,4,5]

for i in x:
    print(i)
```

```
1
2
3
4
5
```

range()

The range() function in Python is a built-in function that generates a sequence of numbers. It is a powerful tool for iterating over a specific range of values and performing various operations on them.

range(start,stop,step)

start:

- The start parameter specifies the starting value of the sequence.
- If not provided, the default value is 0. For example, range(5) generates a sequence starting from 0 and ending at 4.

stop:

- The stop parameter specifies the ending value of the sequence, excluding the value itself.
- For instance, range(5) generates a sequence from 0 to 4, but not 5.

step:

- The step parameter determines the increment or decrement between consecutive values in the sequence. It defaults to 1, indicating that the sequence increases by 1 with each iteration.
- For example, range(1, 10, 2) generates a sequence from 1 to 9, incrementing by 2. Negative values for step indicate a decreasing sequence.

```
for i in range(0,11,1):
    print(i)
```

```
0
1
2
3
4
5
6
7
8
9
10
```

The break Statement

With the break statement we can stop the loop before it has looped through all the items:

```
cities = ["Mumbai", "Delhi", "Bangalore", "Hyderabad"]
for x in cities:
    print(x)
```

```
    if x == "Delhi":
        break
```

```
Mumbai
Delhi
```

The continue Statement

With the continue statement we can stop the current iteration of the loop, and continue with the next:

```
cities = ["Mumbai", "Delhi", "Bangalore", "Hyderabad"]
for x in cities:
    if x == "Delhi":
        continue
    print(x)
```

```
Mumbai
Bangalore
Hyderabad
```

Nested Loops

- A nested loop is a loop inside a loop.
- The "inner loop" will be executed one time for each iteration of the "outer loop":

```
country = ["India"]
cities = ["Mumbai", "Delhi", "Bangalore", "Hyderabad"]

for i in country:
    for j in cities:
        print(i,j)
```

```
India Mumbai
India Delhi
India Bangalore
India Hyderabad
```

The pass Statement

for loops cannot be empty, but if you for some reason have a for loop with no content, put in the pass statement to avoid getting an error.

```
for i in range(10):
    pass
```

Learning Progress:

For Loops: I have successfully grasped the concept of for loops in Python and their application in iterating over sequences. I understand how to utilize the range() function to generate sequences of numbers.

break, continue, and pass Statements: I have thoroughly explored the usage of break, continue, and pass statements within for loops. I can effectively employ these statements to control the flow of execution and modify the behavior of the loop.

Next Steps:

Multiplication Table: I am ready to develop a program that prints the multiplication table of a number given by user in both ascending and descending order. I will utilize for loops and the range() function to generate the desired sequence of numbers and their corresponding products.

Ascending Order:

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

for i in range(1,11):
    print(num,"X",i,"=",num*i)
```

```
Enter a number: 7
7 X 1 = 7
7 X 2 = 14
7 X 3 = 21
7 X 4 = 28
7 X 5 = 35
7 X 6 = 42
7 X 7 = 49
7 X 8 = 56
7 X 9 = 63
7 X 10 = 70
```

Descending Order:

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

for i in range(10,0,-1):
    print(num,"X",i,"=",num*i)
```

```
Enter a number: 7
7 X 10 = 70
7 X 9 = 63
7 X 8 = 56
7 X 7 = 49
7 X 6 = 42
7 X 5 = 35
7 X 4 = 28
```

```
7 X 3 = 21
7 X 2 = 14
7 X 1 = 7
```

While Loop: A while loop is a control flow statement that allows you to execute a block of code repeatedly as long as a certain condition is true. The condition is checked at the beginning of each iteration of the loop. If the condition is false, the loop will terminate. It is a powerful tool that can be used to automate repetitive tasks. They are also a concise way to execute a block of code an unknown number of times.

It will print from 0 to 9 number because we have printed the number then we increment the number

```
i = 0  ##assignmnet  i = 0, ... , 9
while i<10: ##condition
    print(i) ##Body
    i = i + 1 ##Increment
```

```
0
1
2
3
4
5
6
7
8
9
```

It will print from 1 to 10 because we have increment the variable before printing it

```
i = 0  ##assignmnet  i = 1, ... ,10
while i<10: ##condition
    i = i + 1 ##Increment
    print(i) ##Body
```

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

Break Statement

```
i = 0
while i < 7:
    print(i)
    if i == 5:
        break
    i = i + 1
```

```
0
1
2
3
4
5
```

Continue Statement

```
i = 0
while i < 7:
    i = i + 1
    if i == 5:
        continue
    print(i)
```

```
1
2
3
4
6
7
```

Learning Progress:

While Loops: I have thoroughly understood the concept of while loops in Python and their application in iterating as long as a condition is true. I can effectively utilize break and continue statements within while loops to control the flow of execution.

break and continue Statements: I have gained a comprehensive understanding of the usage of break and continue statements within while loops. I can effectively employ these statements to terminate the loop or skip specific iterations based on certain conditions.

Next Steps:

Multiplication Table: I am prepared to develop a program that prints the multiplication table of a number in both ascending and descending order using while loops. I will utilize the break and continue statements to control the loop's behavior and generate the desired sequence of numbers and their corresponding products.

Ascending Order:

```
i = 1
j = 13
while i < 13:
    i = i + 1
    print(i*j)
```

```
26
39
52
65
78
91
104
117
130
143
156
169
```

Descending Order:

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

for i in range(10,0,-1):
    print(num,"X",i,"=",num*i)
```