Session 5 - Loops

1. Importance of loops in Programming
2. For loop
3. range() function
4. While loop
5. Loop control statement

Importance of loops in Programming

lets say we wanted to print `hello world` 5 times. The below cell shows how it can be implimented without for loops

In [ ]:
```
print("Hello world")
print("Hello world")
print("Hello world")
print("Hello world")
print("Hello world")
```

The Idea behind using loops is to

- Reduce the copy paste work
- Reduce the number of lines of code (the memory footprnit)

The above output can be acheived using for loop as follows :

In [ ]:
```
for x in range(0,5):
    print("hello world")
```

We can also use while loop to get the same output

In [ ]:
```
x = 0
while x<5:
    print("hello world")
    x = x+1
```

As we can see, loops reduces copy and pasting required and also reduces the number of lines of code

For Loop in Python

For loop in python is a bit different that other languages.
In python the for loop is used to loop through another iterable datatype or collection datatype (list, string etc)
This means we must have a iterable datatype declared first if we want to use a for loop.

In [ ]:
```
l = [1,2,3,4,5]       # declaring a list with 5 elements in it

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

The `x` is just a variable name which is used to hold the elements from the list `l` one at a time.
We can chose to use or not use the variable in our code.

In [ ]:
```
l = [1,2,3,4,5]       # declaring a list with 5 elements in it

for abcd in l:
    print("hello world")
```

As soon as the elements in the iterable obejct runs out the for loop stops.
which means if we want to have a for loop run for 100 times we will have to create a list or any other iterable which has that many elements in it.

range() function

Range function helps up to overcome the above stated probelems with the for loop.
The range function is a generator (we will be locking at geenrators at a later time).
All we need to know as of now is that the range function return the value between the limits entered in the stpes provided.

Syntax for range
`range(start number, stop number+1, step size)`

- The step size is 1 by default so we can skip it :)
- The stop number should always be added by one as the range is not inclusive.

In [ ]:
```
print(range(0,10))
```

The above code returns nothing beacuse the range fucntion is a generator.
We must either

- cast a range fucntion
- Use it inside a loop

casting a range function to create list

In [ ]:
```
my_list = list(range(0,10))
print(my_list)
```

This gives us a quick and handy way of generatig lists. which can be used in a loop. But that is not efefctive as we can direclty use range inside a loop.

using Range directly inside the loop

In [ ]:
```
for var in range(1,11):
    print(var)
```

TASK 1

- **create a list of all even numbers between 0 and 100**

In [ ]:
```

```

Using range() function gives us a for loop which is similar to most other programming languages

In [ ]:
```
for x in range(0,10):
    print("hello world")
```

In [ ]:
```

```

While Loop in Python

- While loops are useful when we don't know the number of times we might need to execute a certain task
- A While loop in python is similar to other languages.
- By default a while loop is an Infinite loop.
- We need to have control statement to control the execution of a while loop

Syntax for while loop

```
while (condition):
  program statement 1
  program statement 1
  ...
  ...
  Control statement
```

Warning!!

Before you execute the below code you must know a few things

- The below while loop is an infinite loop without the control statement.
- This means the loop will start executing the statement inside the loop indefinitely.
- This in rare cases can cause the PC to hang a bit.
- You can interrupt the while loop by going to `Kernel(menu bar)>Restart>`

In [ ]:
```
while True:
    print("hello")
```

While loop should be used with a control variable which make it a finite loop.
In the below example we are using the variable `x` to control the while loop.
Control varibales are constantly updated inside the while loop (similar to a counter)

In [ ]:
```
x = 0
while x<5:
    print("hello")
    x = x + 1
```

Control Statements in Python

1. pass
2. break
3. continue

These keywords interrupt the normal flow of a loop.
The pass keyword can be used with other python objects like if else statements and fucntions

In [ ]:
```
for x in range(0,10):

print("hello")
```

Pass keyword helps us to have empty for loops or if else statements

In [ ]:
```
for x in range(0,10):
    pass
print("hello")
```

In [ ]:
```
a = 10
b = 11
if a<b:
    pass
print("hello")
```

The break statement stops the execution of the closest enclosing loop
The continue statement skips the loop for that instance.

In [ ]:
```
for x in range(0,10):
    if x == 5:
        continue
```

```
    if x == 8:
        break
    print(x)
```

TASK 2

- **Given a sentence print out only the words which start with the letter 'f'**

In [ ]:
```
mystring = "the quick brown fox jumped over the fence and dissapeared into the forest"

## Write the code here
```

Try printing the following pattern

```
*
**
***
****
*****
```

Before we start with the pattern problem we need to understand 2 main concepts

- nested loops
- the `print()` fucntion in python

Nested loops

In [ ]:
```
for i in range(0,3):
    for j in range(0,4):
        print(i,j)
```

The `print()` function

The print fucntion always put a enter key press at the end of the string being printed (more technical explanation would be the print statements puta a new line charater at the end of string being printed). This is what causes the next print statement to be on the next line.

Just to clarify things.....

new line character == enter key press == `\n`
whenever we press the enter we are entering a new line character which is invisible to us but is there at the end of that line.

Although very importnat we might have not noticed this behaviour. Lets just look at an example.

In [ ]:
```
print('hello')
print('world')
```

Look how 'hello' and 'world' truned out to be in new lines even though we didn't specify them to be on new lines. This points out that python is entering the newline character(\n) at the end of string.

We can stop python from doing this by specifying the end keyword to be empyt.

In [ ]:
```
print('hello',end = '')
print('world')
```

By default the end is equal to '\n' so the new line appears. we can confirm this with an example.

In [ ]:
```
print('hello',end = '\n')
print('world')
```

Now we have all the information to tackle the pattern problems

In [ ]:
```
for x in range(1,6):
    for i in range(0,x):
        print("*",end = '')
    print()
```

In [ ]:
```

```

HOMEWORK

1. Print the following star pattern using nested for loop

```
*****
****
***
**
*
```

2. print the above pattern but the number of rows depends on the user input

3. print the above pattern but the number of rows depends on the user input

```
----*
---**
--***
-****
*****
```

4. Write a program that prints the integers from 1 to 100.

```
- But for multiples of three print "Fizz" instead of the number
- For the multiples of five print "Buzz".
- For numbers which are multiples of both three and five print
"FizzBuzz".
```

5. Create an application which keeps taking input from the user unless he enters a `q`

6. Create an Guessing Game

- The user has 5 chance to guess a random number.
- After every guess the user should be given a hint if the guessd number was lesser than or greater than the actual random number
- If the user wins the game by guessing the correct number before his 5 chance is over. "you have won" should be printed.
- If the user is not able to guess the number. Print "you lost" along with the actual number.

HOMEWORK SOLUTION

In [8]:
```
# TASK 1 :

for x in range(1,6):
    for i in range(x,6):
        print("*",end = '')
    print()
```

```
*****
****
***
**
*
```

In [9]:
```
#TASK 2:
rows = int(input("enter number of rows: "))

for x in range(rows):
    for i in range(x,rows):
        print("*",end = '')
    print()
```

```
enter number of rows: 5
*****
****
***
**
*
```

In [11]:
```
#TASK 3:

rows = int(input("enter number of rows: "))

star =1
dash =4
for x in range(rows):
```

```
    for i in range(dash):
        print("-",end = '')


    for j in range(star):
        print("*",end = '')

    star+=1
    dash-=1

    print()
```

```
enter number of rows: 5
----*
---**
--***
-****
*****
```

In [2]:

```
#TASK 5:

for i in range(1,100):
    if i%3 ==0 and i %5 ==0:
        print(i,end=" ")
        print("FizzBuzz")

    elif i % 5 ==0:
        print(i,end=" ")
        print("Buzz")

    elif i%3 ==0:
        print(i,end=" ")
        print("Fizz")
```

```
3 Fizz
5 Buzz
6 Fizz
9 Fizz
10 Buzz
12 Fizz
15 FizzBuzz
18 Fizz
20 Buzz
21 Fizz
24 Fizz
25 Buzz
27 Fizz
30 FizzBuzz
33 Fizz
35 Buzz
36 Fizz
39 Fizz
40 Buzz
42 Fizz
45 FizzBuzz
48 Fizz
50 Buzz
51 Fizz
54 Fizz
55 Buzz
57 Fizz
60 FizzBuzz
```

```
63 Fizz
65 Buzz
66 Fizz
69 Fizz
70 Buzz
72 Fizz
75 FizzBuzz
78 Fizz
80 Buzz
81 Fizz
84 Fizz
85 Buzz
87 Fizz
90 FizzBuzz
93 Fizz
95 Buzz
96 Fizz
99 Fizz
```

In [ ]:

```python
#TASK 6:
while True:
    choice=input("Do you want to stop press q or else press enter")
    if choice =="q":
        print("everything stopped")
        break

    else:
        print("Everything is allright ")
```

In [ ]:

```python
#TASK 7:

import random

random_choice = random.randint(1,6)

for i in range(5):
    user_choice = int(input("Enter your choice from 1 to 6     "))
    if user_choice == random_choice:
        print("you have won ")
        break
    elif user_choice < random_choice:
        print("Your number is less than the actual number")

    else:
        print("Your number is greater than the actual number")

print("You lost !! Actual number is ",random_choice)
```

In [ ]: