19th April 2021

Previous day:-

Python if-else
if/elif/else

- Conditional execution of code:-

```
if logical_operation:
        conditional code
elif second_logical_operation:
        conditional code
else :
        conditional code
```

- Operators:-
    - ==
    - !=
    - >
    - <
    - >=
    - <=
    - and
    - or
    - not

Python Loops

Lecture Flow:-

- loops
    - for loop
    - while loop
- right angle pyramid using loop
    - printing pyramid using for loop
    - printing pyramid using while loop
- MCQs

Topics & Explanation

For better understanding and visualization loops please use Python Tutor.

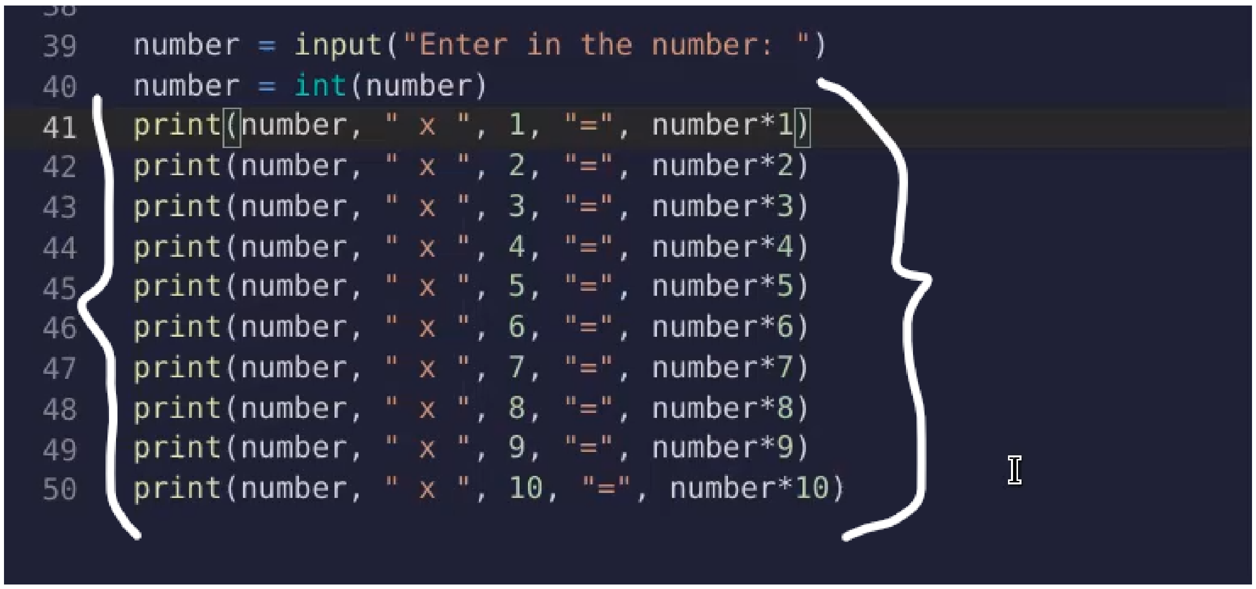
Conditional code:- code that will always be executed which depends on any value or variable and will only execute if the condition is satisfied.

Unconditional code:-code that will always be executed which depends on any value or variable and will only execute if the condition is satisfied.

Repetitive code:- it repeats a particular section of code until a particular end and start condition and with minor changes.

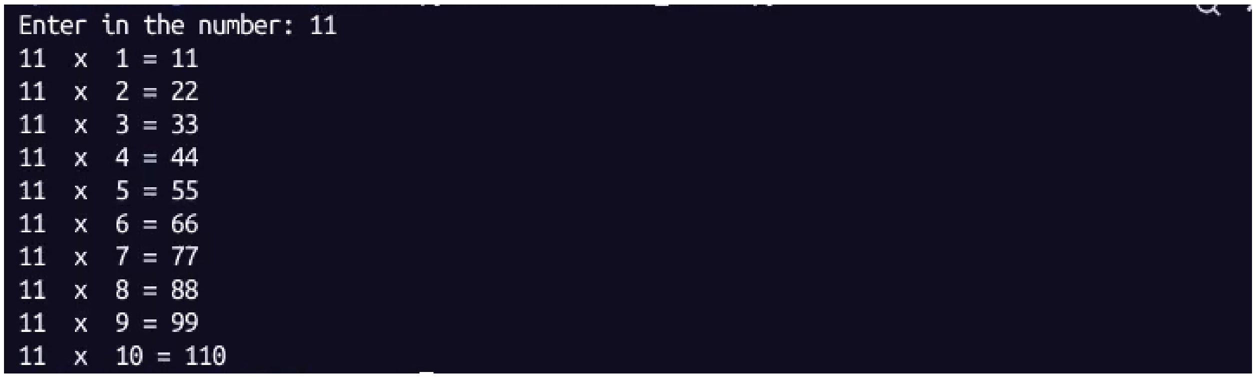
It is also known as Loop. It means to repeat the same thing over and over again.

Ex:- In the below image lines which are highlighted by the curly braces are the repetitive lines.



```python
number = input("Enter in the number: ")
number = int(number)
print(number, " x ", 1, "=", number*1)
print(number, " x ", 2, "=", number*2)
print(number, " x ", 3, "=", number*3)
print(number, " x ", 4, "=", number*4)
print(number, " x ", 5, "=", number*5)
print(number, " x ", 6, "=", number*6)
print(number, " x ", 7, "=", number*7)
print(number, " x ", 8, "=", number*8)
print(number, " x ", 9, "=", number*9)
print(number, " x ", 10, "=", number*10)
```

Output:-



```
Enter in the number: 11
11  x  1 = 11
11  x  2 = 22
11  x  3 = 33
11  x  4 = 44
11  x  5 = 55
11  x  6 = 66
11  x  7 = 77
11  x  8 = 88
11  x  9 = 99
11  x  10 = 110
```

Loops:-

1. for loop:-

click here for reference

Concept

For loops, in general, are used for sequential traversal. It falls under the category of **definite** iteration. Definite iterations means the number of repetitions is specified explicitly in advance.

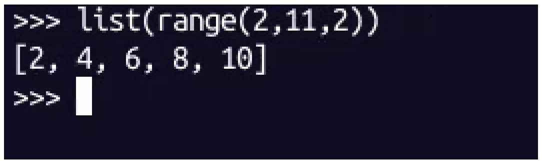
For better understanding let's say, there is a coke factory and there we need to fill the bottles with coke. We were given how much to fill in each bottle of coke and we were also given the total amount of coke.

Syntax

```
for any_variable in range(start, end+1, jump)
        some code here
```

range():- it will create a list of numbers and the loop will work on those lists of numbers.

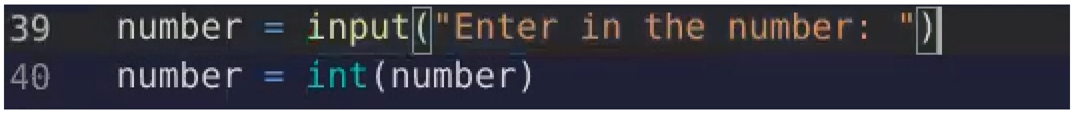
Ex:-



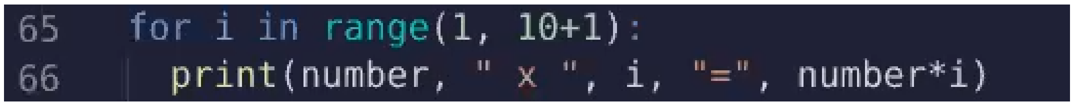
```
>>> list(range(2,11,2))
[2, 4, 6, 8, 10]
>>>
```

in the above image let's say we want to print the list of even numbers from 1 to 10. So "range(2,11,2)", you can try in the terminal window to check what range() function does by typing "list(range(2,11,2))" and press enter. The output will be [2,4,6,8,10]. Thus it creates the list of numbers from 1 to 10.

Example



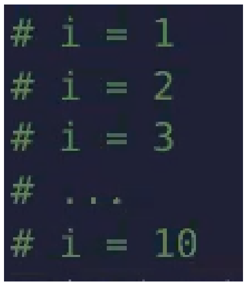
```
39    number = input("Enter in the number: ")
40    number = int(number)
```



```
65    for i in range(1, 10+1):
66        print(number, " x ", i, "=", number*i)
```

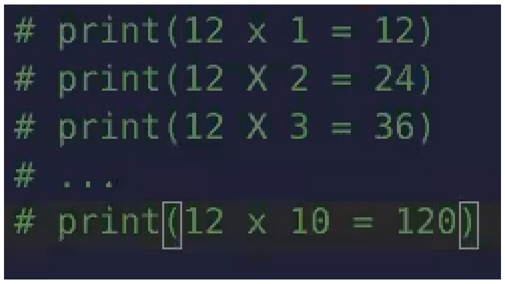
We took a number as an input in line39 and we typecast it into integer in line40 because by default it is in string. In line65 we ran a loop from 1 to 11 we will run from 1 to 10 and by default it will step over as 1. In line66 we have print the “number” which is input from user multiplied by “i” which will run from 1 to 10 is equal to “number x i”.

Dry run:-



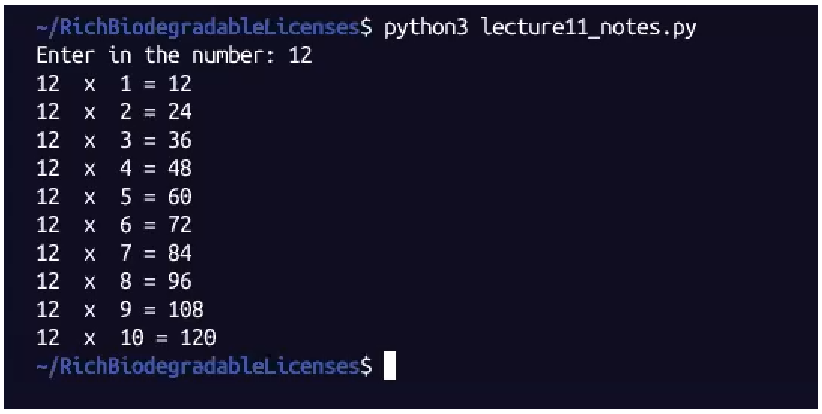
```
# i = 1
# i = 2
# i = 3
# ...
# i = 10
```

--------->



```
# print(12 x 1 = 12)
# print(12 X 2 = 24)
# print(12 X 3 = 36)
# ...
# print(12 x 10 = 120)
```

Output:-
we took the input as 12 and press enter



```
~/RichBiodegradableLicenses$ python3 lecture11_notes.py
Enter in the number: 12
12  x  1 = 12
12  x  2 = 24
12  x  3 = 36
12  x  4 = 48
12  x  5 = 60
12  x  6 = 72
12  x  7 = 84
12  x  8 = 96
12  x  9 = 108
12  x  10 = 120
~/RichBiodegradableLicenses$
```

2. while loop:-

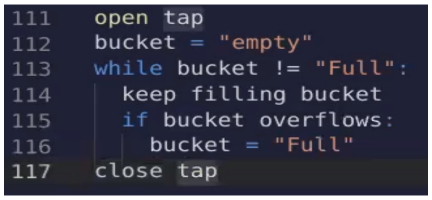
click here for reference

For loops, in general, are used for sequential traversal. It falls under the category of **definite** iteration. Definite iterations means the number of repetitions is specified explicitly in advance.

For better understanding let’s say, we have the bucket and we need to fill the water with the tap. We will use this while loop when we don’t know how much water the bucket has in litres also we don't know how much water the tap fills in one second. So we will keep the tap running until the bucket is full.

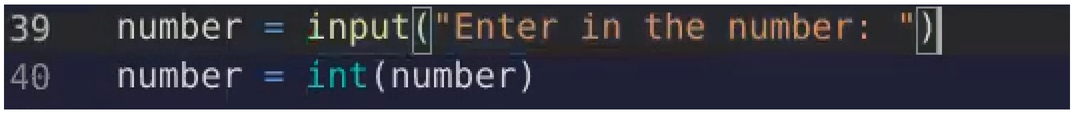
<u>Syntax</u>

```
while condition:        ←---
        statement          |
        increment loop  _|
```

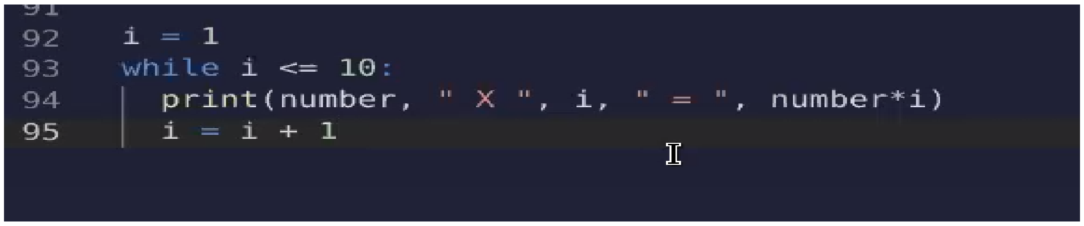


```
open tap
bucket = "empty"
while bucket != "Full":
  keep filling bucket
  if bucket overflows:
    bucket = "Full"
close tap
```

<u>Example</u>



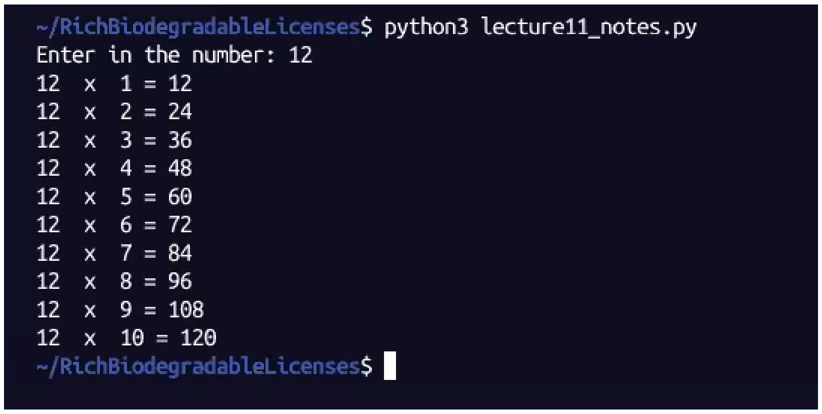
```
39    number = input("Enter in the number: ")
40    number = int(number)
```



```
92    i = 1
93    while i <= 10:
94      print(number, " X ", i, " = ", number*i)
95      i = i + 1
```

We took a number as an input in line39 and we typecast it into integer in line40 because by default it is in string. We initialize the i=1 in line92. We run the while loop in line93 and check the condition every time and if the condition is true we will move to the while statement.

Output:-
we took the input as 12 and press enter



```
~/RichBiodegradableLicenses$ python3 lecture11_notes.py
Enter in the number: 12
12  x  1 = 12
12  x  2 = 24
12  x  3 = 36
12  x  4 = 48
12  x  5 = 60
12  x  6 = 72
12  x  7 = 84
12  x  8 = 96
12  x  9 = 108
12  x  10 = 120
~/RichBiodegradableLicenses$
```

Dry run
i=1 --- check condition line93: true ---> line96->line97 --- 12 x1 = 12

i=2 ---> check condition line93: true ---> line96->line97 --- 12 x2 = 24

i=3 ---> check condition line93: true ---> line96->line97 --- 12 x3 = 36
.
.
.

i=9 ---> check condition line93: true ---> line96->line97 --- 12 x9 = 108

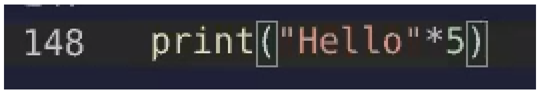
i=10 ---> check condition line93: TRUE ---> line96->line97 --- 12 x10 = 120

i=11 ---> check condition line93: FALSE ---> out of the while loop
Note:-In while loop the value of the variable in the loop condition, remains unchanged and as a result the logical operation always remains True, then it is called as an infinite loop.

Note:- In python loops are interchangeable i.e., everything you can do with a for loop, you can also do with a while loop as well and vice versa.

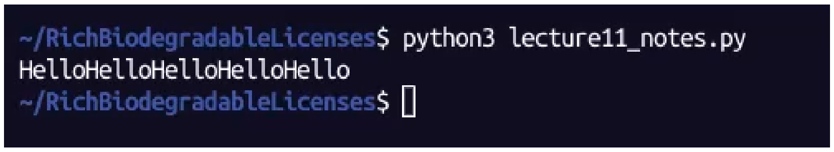
Right angle Pyramid using loop:-

Note:- String has a property called string repetition. So, if you multiply a string by a number, the string gets repeated those many times.
Ex-



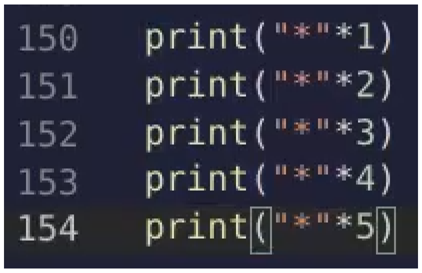
```
print("Hello"*5)
```

Output:-



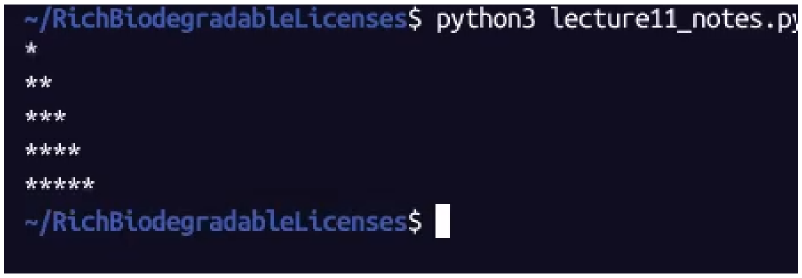
```
~/RichBiodegradableLicenses$ python3 lecture11_notes.py
HelloHelloHelloHelloHello
~/RichBiodegradableLicenses$ 
```

Printing pyramid using print statement to understand the logic:-



```
print("*"*1)
print("*"*2)
print("*"*3)
print("*"*4)
print("*"*5)
```

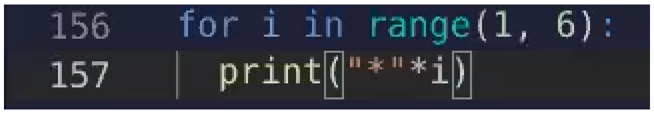
using print statement we are printing the “*” in every line to make the pyramid.
Output:-



```
~/RichBiodegradableLicenses$ python3 lecture11_notes.py
*
**
***
****
*****
~/RichBiodegradableLicenses$
```

Printing pyramid using for loop:-

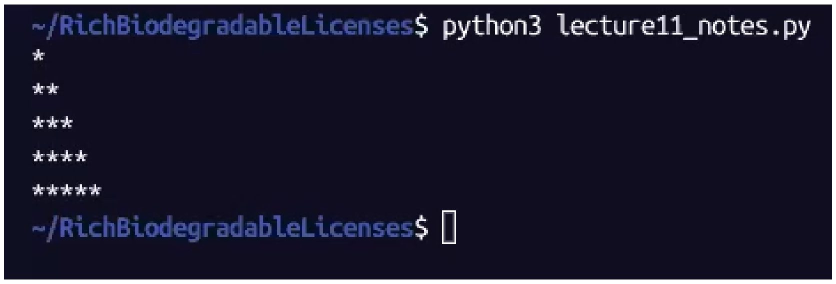
Example 1



```
for i in range(1, 6):
    print("*"*i)
```

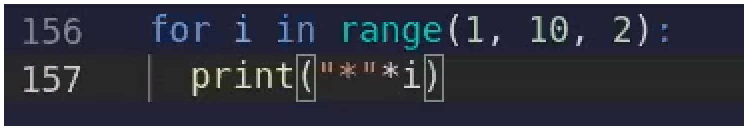
Like in the above example we have used print statements 5 times to make the pyramid. In this we are instead of using print statement 5 times we have put the print statement in the loop and run the loop for the 5 times “range(1,6)” that will print the statement 5 times. This will reduce the number of lines of code also the time.

Output:-



```
~/RichBiodegradableLicenses$ python3 lecture11_notes.py
*
**
***
****
*****
~/RichBiodegradableLicenses$
```

Example 2

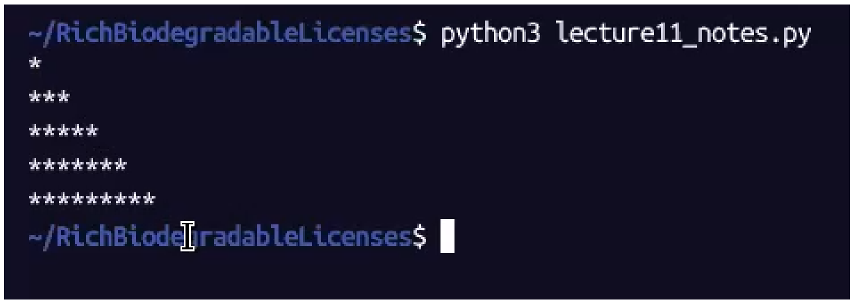


```python
for i in range(1, 10, 2):
  print("*"*i)
```

______

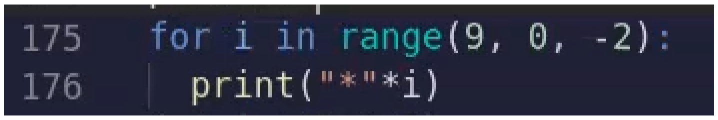
______Like in the above example we print the pyramid in the series (1*,2*,3*,4*,5*) but in this we are printing the pyramid in the series like (1*,3*,5*,7*,9*). So range(1,10,**2**) this will run the loop from 1 to 9 with the jump of 2.

______Output



```
~/RichBiodegradableLicenses$ python3 lecture11_notes.py
*
***
*****
*******
*********
~/RichBiodegradableLicenses$
```

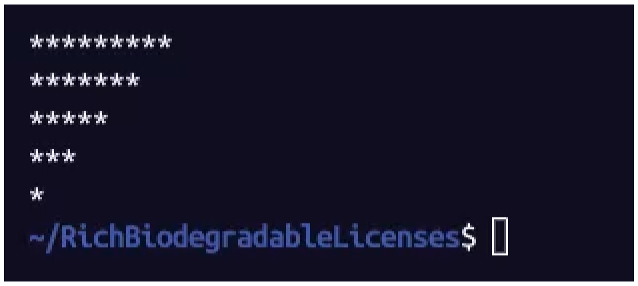
Printing pyramid using for loop:-



```python
for i in range(9, 0, -2):
  print("*"*i)
```

The for loop goes from 9 to 0 with the jump of -2 and print the statement when the range goes from 9 to 0.

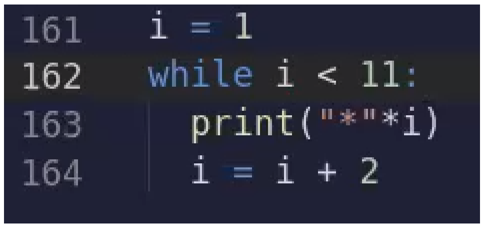
Output:-



```
*********
*******
*****
***
*
~/RichBiodegradableLicenses$
```

Printing pyramid using while loop:-

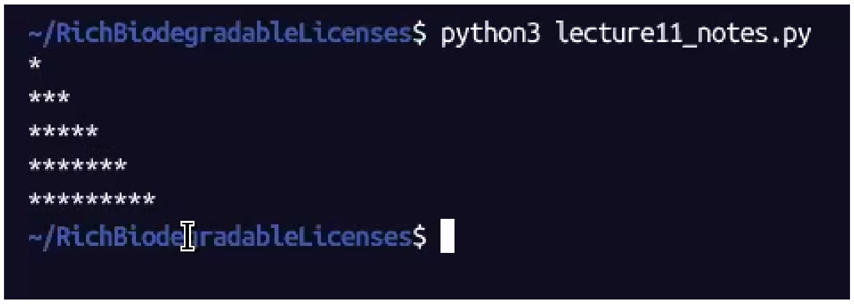
Example:-



```
i = 1
while i < 11:
    print("*"*i)
    i = i + 2
```

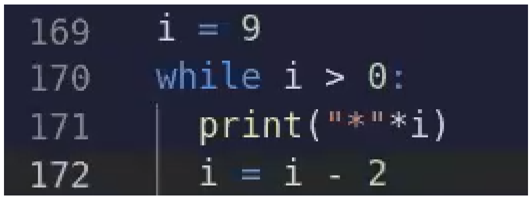
Same logic as for loop. We initialized the variable “i=1”, then the condition “while i<11” to check everytime if the condition is true or not, then if the condition is true it will move to the print statement. Then “i” is incremented by 2, again it will check if the condition is true or not and this process goes on until the condition is false.

Output:-



```
~/RichBiodegradableLicenses$ python3 lecture11_notes.py
*
***
*****
*******
*********
~/RichBiodegradableLicenses$
```

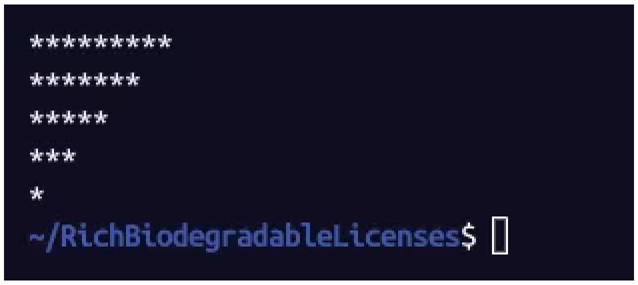
Printing reverse pyramid using while loop:-



```
i = 9
while i > 0:
    print("*"*i)
    i = i - 2
```

The variable “i”starts from 9 it will check the condition if it is greater than 0 if the condition is true move to the while statement and print. After print it will move down to the next line which will decrement the value by 2 again it goes up and checks the condition if it is correct or not and goes on until the condition is false.

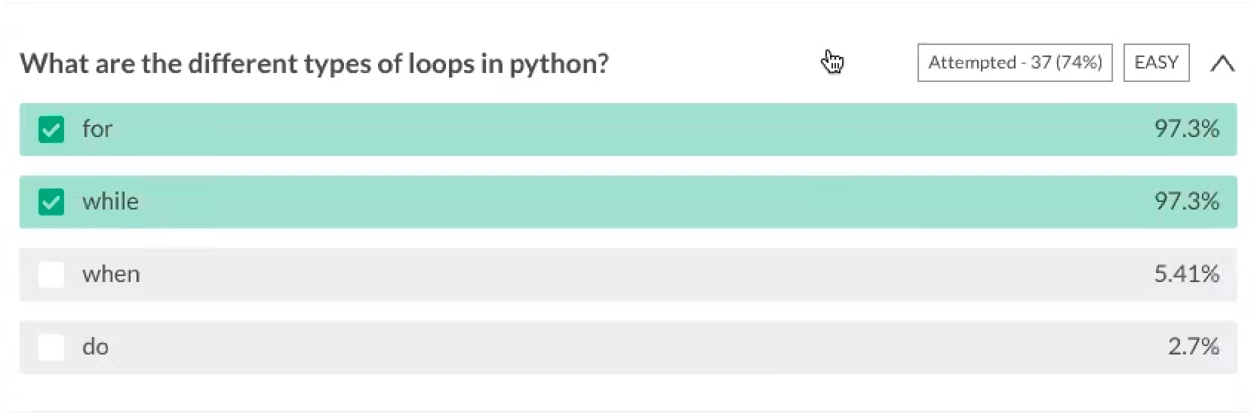
Output:-



```
*********
*******
*****
***
*
~/RichBiodegradableLicenses$ 
```

<u>Note:-</u> Ctrl + / is used to comment or uncomment multiple lines.

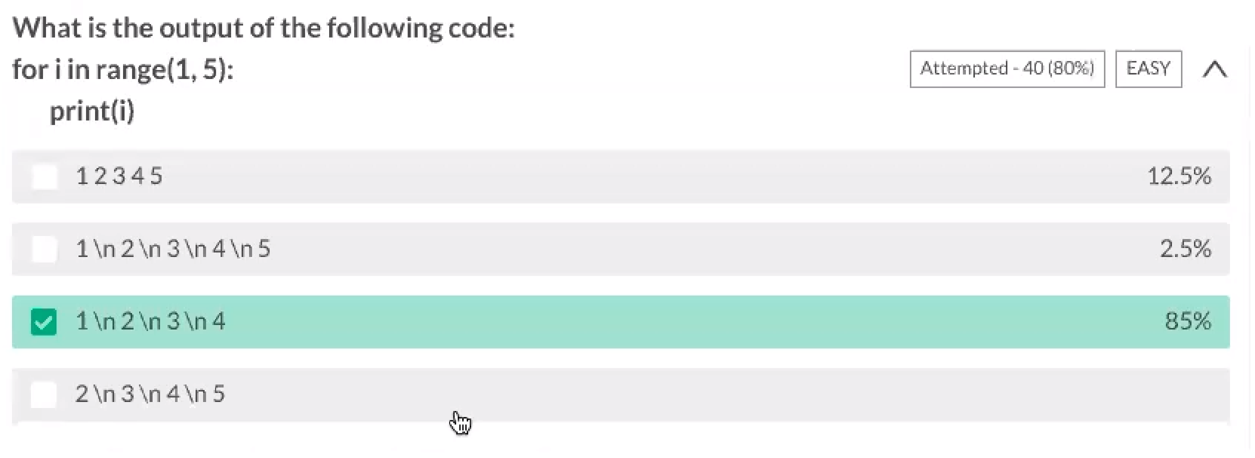
<u>MCQs</u>:-



What are the different types of loops in python?

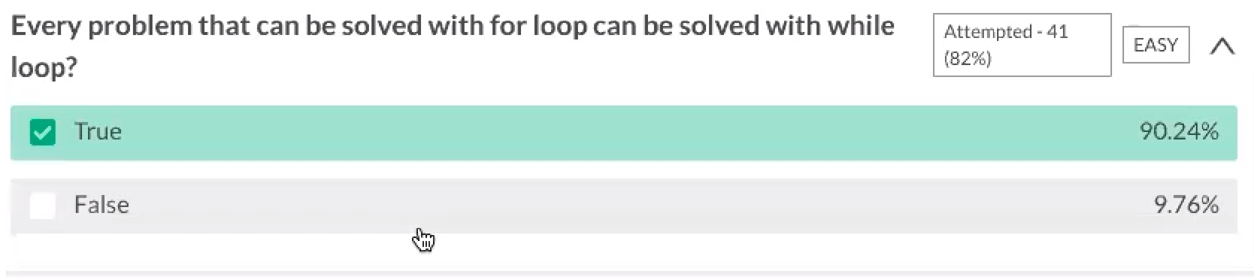
Attempted - 37 (74%) EASY

✓	for	97.3%
✓	while	97.3%
	when	5.41%
	do	2.7%


What is the output of the following code:
for i in range(1, 5):
    print(i)

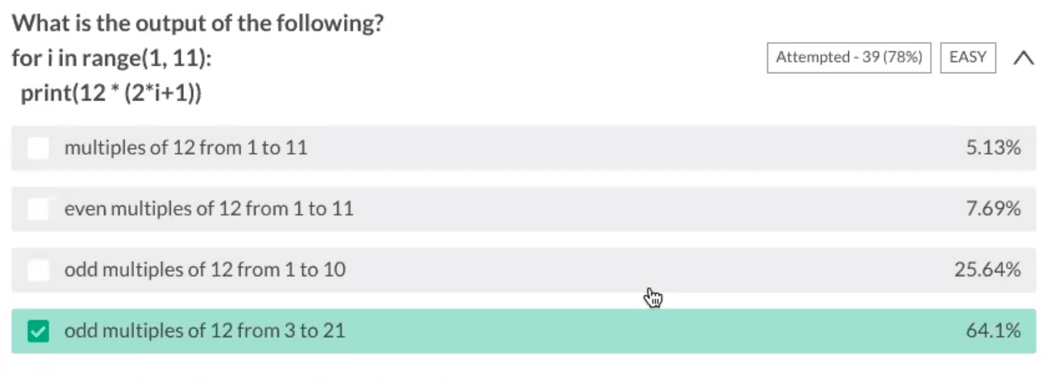
Attempted - 40 (80%) EASY

- 1 2 3 4 5 — 12.5%
- 1 \n 2 \n 3 \n 4 \n 5 — 2.5%
- [✓ 1 \n 2 \n 3 \n 4] — 85%
- 2 \n 3 \n 4 \n 5


Every problem that can be solved with for loop can be solved with while loop?

Attempted - 41 (82%) EASY

- [✓ True] — 90.24%
- False — 9.76%


What is the output of the following?
for i in range(1, 11):
 print(12 * (2*i+1))

Attempted - 39 (78%) EASY

- multiples of 12 from 1 to 11 — 5.13%
- even multiples of 12 from 1 to 11 — 7.69%
- odd multiples of 12 from 1 to 10 — 25.64%
- [✓ odd multiples of 12 from 3 to 21] — 64.1%