TOPIC 6: Decision Making in Python

Introduction.

Decision making is anticipation of conditions occurring while execution of the program and specifying actions taken according to the conditions. **Decision** structures evaluate multiple expressions which produce TRUE or FALSE as outcome.

Objectives

Objectives by the end of this topic you should be able to demonstrate understanding in:

- if statement
- if..else statements
- nested if statements
- if-elif ladder

Learning activities

Learning Activity 6.1: Reading

Read further on decision making in python.

Learning Activity 6.2: Journal

Write a Python program to find numbers between 100 and 400 (both included) where each digit of a number is an even number. The numbers obtained should be printed in a comma-separated sequence.

Learning Activity 6.3: Discussion

Write a Python program to get the Fibonacci series between 0 to 50. Go to the editor Note : The Fibonacci Sequence is the series of numbers : 0, 1, 1, 2, 3, 5, 8, 13, 21, ... Every next number is found by adding up the two numbers before it

Assessment

Topic resources

1. The Python Tutorial¶. (n.d.). Retrieved from https://docs.python.org/3/tutorial/index.html
2. Mueller, J. P. (n.d.). *Beginning Programming with Python For Dummies*. S.l.: For Dummies.
3. (n.d.). Python 3.7.4 documentation. Retrieved from https://docs.python.org/3
4. (n.d.). Git Handbook. Retrieved from https://guides.github.com/introduction/git-handbook/
5. Shaw, Z. (2017). *Learn Python 3 the hard way: a very simple introduction to the terrifyingly beautiful world of computers and code*. Boston: Addison-Wesley.
6. Bader, D. (2018). *Python tricks: the book*. Vancouver, BC: Dan Bader.
7. Downey, A. B. (2015). *Think Python*. Sebastopol: OReilly.
8. Ramalho, L. (2016). *Fluent Python:*Beijing: OReilly.

URL Links

https://www.tutorialspoint.com/python/python_decision_making.htm

https://www.w3schools.in/python-tutorial/decision-making/

https://data-flair.training/blogs/python-decision-making-expressions/

TOPIC 6 NOTES

There comes situations in real life when we need to make some decisions and based on these decisions, we decide what should we do next. Similar situations arise in programming also where we need to make some decisions and based on these decisions we will execute the next block of code.

Decision making statements in programming languages decides the direction of flow of program execution. Decision making statements available in python are:

- if statement
- if..else statements
- nested if statements
- if-elif ladder

if statement

if statement is the most simple decision making statement. It is used to decide whether a certain statement or block of statements will be executed or not i.e if a certain condition is true then a block of statement is executed otherwise not.

Syntax:

```
if condition:
  # Statements to execute if
  # condition is true
```

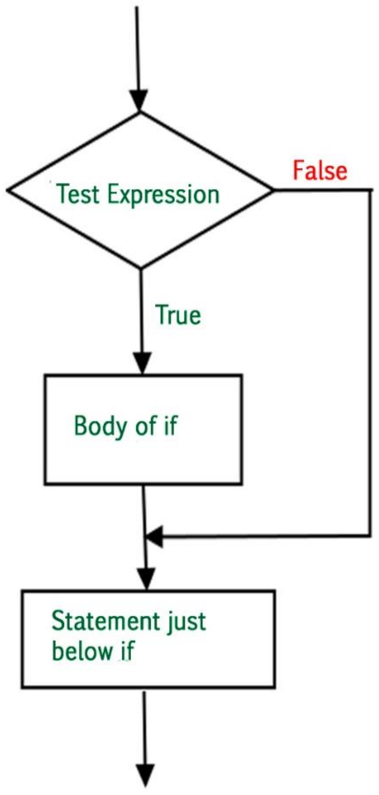
Here, condition after evaluation will be either true or false. if statement accepts boolean values – if the value is true then it will execute the block of statements below it otherwise not. We can use *condition* with bracket '(' ')' also.

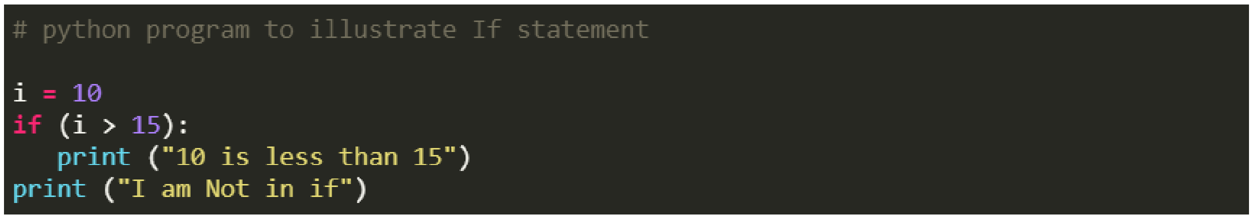
As we know, python uses indentation to identify a block. So the block under an if statement will be identified as shown in the below example:

```
if condition:

  statement1

statement2
```

```
# Here if the condition is true, if block

# will consider only statement1 to be inside

# its block.
```

Flowchart:-





```python
# python program to illustrate If statement

i = 10
if (i > 15):
   print ("10 is less than 15")
print ("I am Not in if")
```

Output:

```
I am Not in if
```

As the condition present in the if statement is false. So, the block below the if statement is not executed.

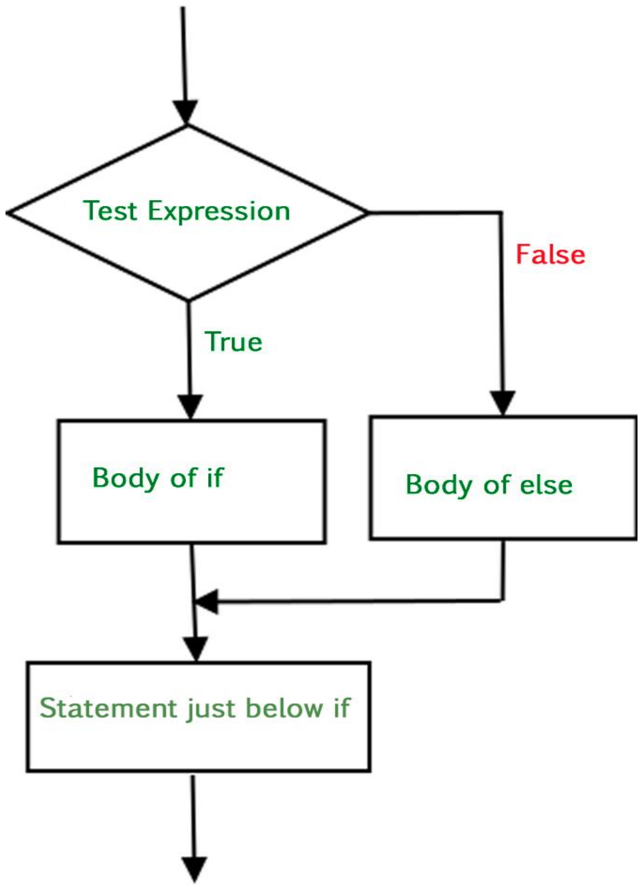
if- else

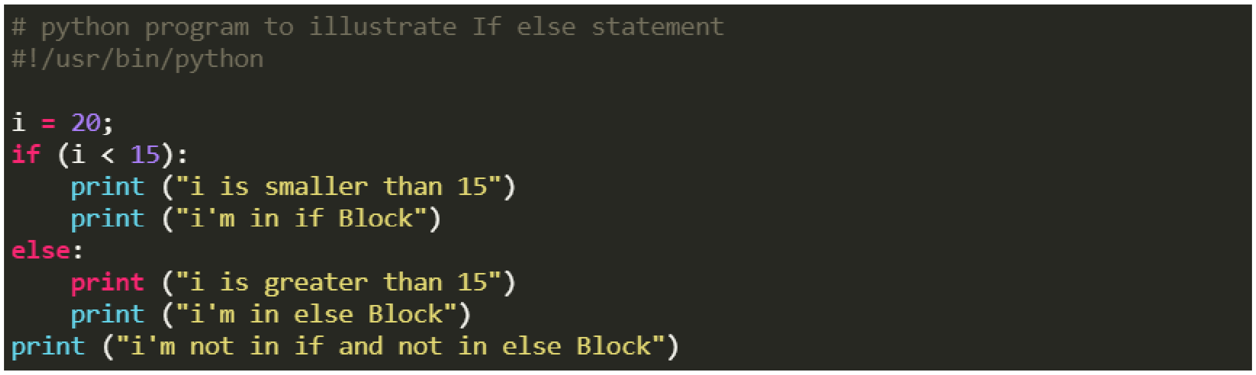
The if statement alone tells us that if a condition is true it will execute a block of statements and if the condition is false it won't. But what if we want to do something else if the condition is false. Here comes the *else* statement. We can use the *else* statement with *if* statement to execute a block of code when the condition is false.

Syntax:

```
if (condition):

    # Executes this block if

    # condition is true

else:

    # Executes this block if

    # condition is false
```

Flow Chart:-





```
# python program to illustrate If else statement
#!/usr/bin/python

i = 20;
if (i < 15):
    print ("i is smaller than 15")
    print ("i'm in if Block")
else:
    print ("i is greater than 15")
    print ("i'm in else Block")
print ("i'm not in if and not in else Block")
```

Output:

```
i is greater than 15

i'm in else Block

i'm not in if and not in else Block
```

The block of code following the else statement is executed as the condition present in the if statement is false after call the statement which is not in block(without spaces).

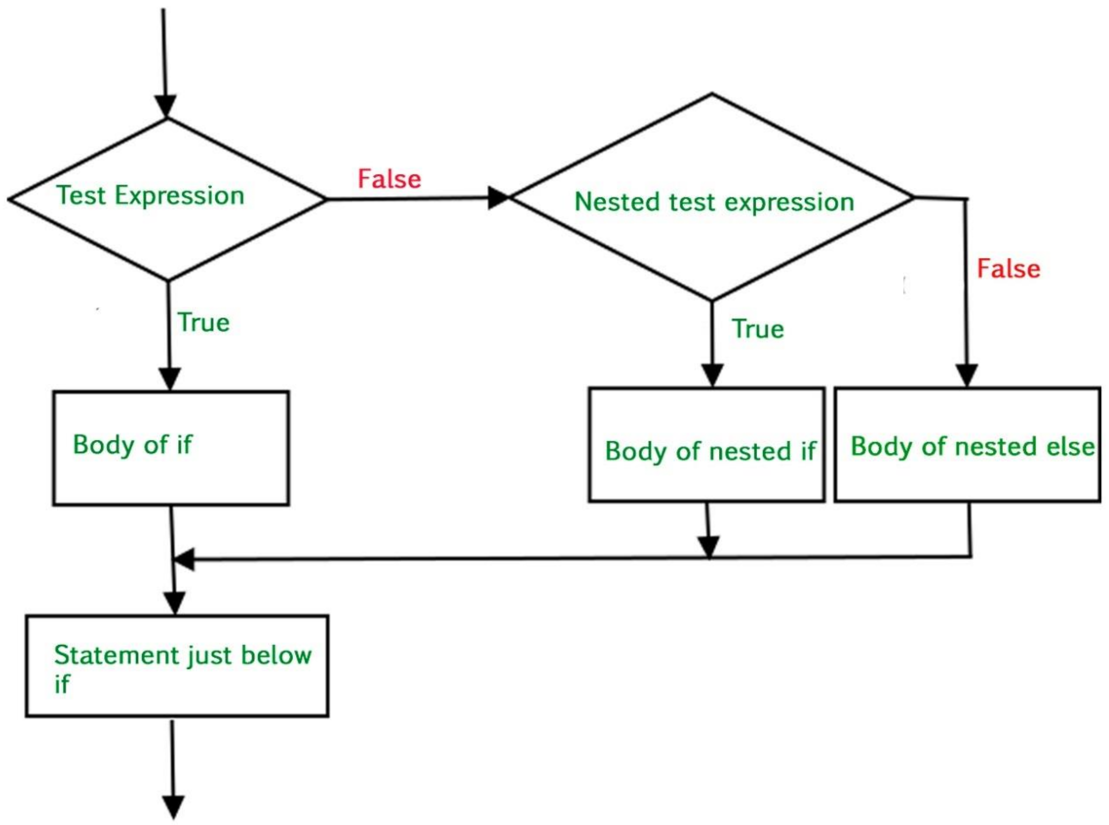
nested-if

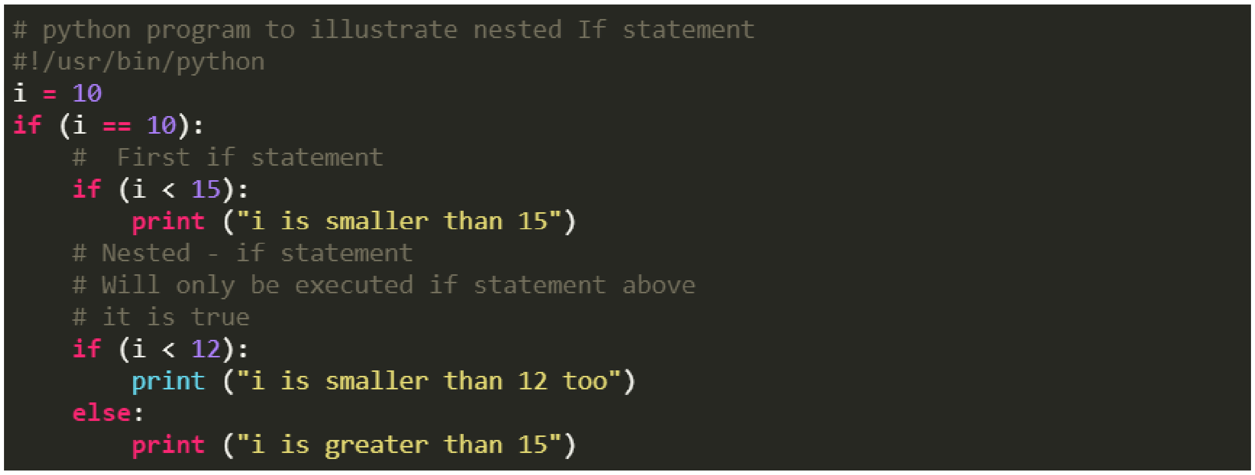
A nested if is an if statement that is the target of another if statement. Nested if statements means an if statement inside another if statement. Yes, Python allows us to nest if statements within if statements. i.e, we can place an if statement inside another if statement.

Syntax:

```
if (condition1):

  # Executes when condition1 is true

  if (condition2):

    # Executes when condition2 is true

  # if Block is end here

# if Block is end here
```

Flow chart:-





```
# python program to illustrate nested If statement
#!/usr/bin/python
i = 10
if (i == 10):
    #  First if statement
    if (i < 15):
        print ("i is smaller than 15")
    # Nested - if statement
    # Will only be executed if statement above
    # it is true
    if (i < 12):
        print ("i is smaller than 12 too")
    else:
        print ("i is greater than 15")
```

Output:

```
i is smaller than 15

i is smaller than 12 too
```

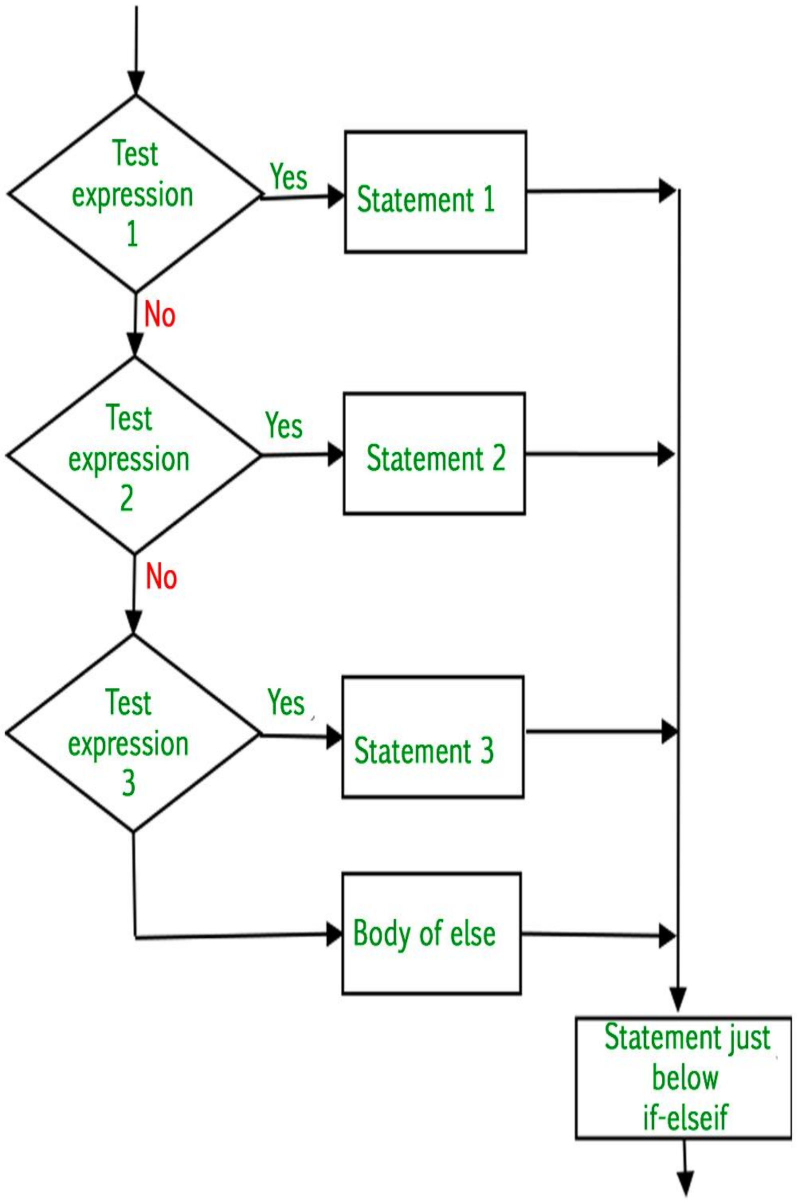
if-elif-else ladder

Here, a user can decide among multiple options. The if statements are executed from the top down. As soon as one of the conditions controlling the if is true, the statement associated with that if is executed, and the rest of the ladder is bypassed. If none of the conditions is true, then the final else statement will be executed.

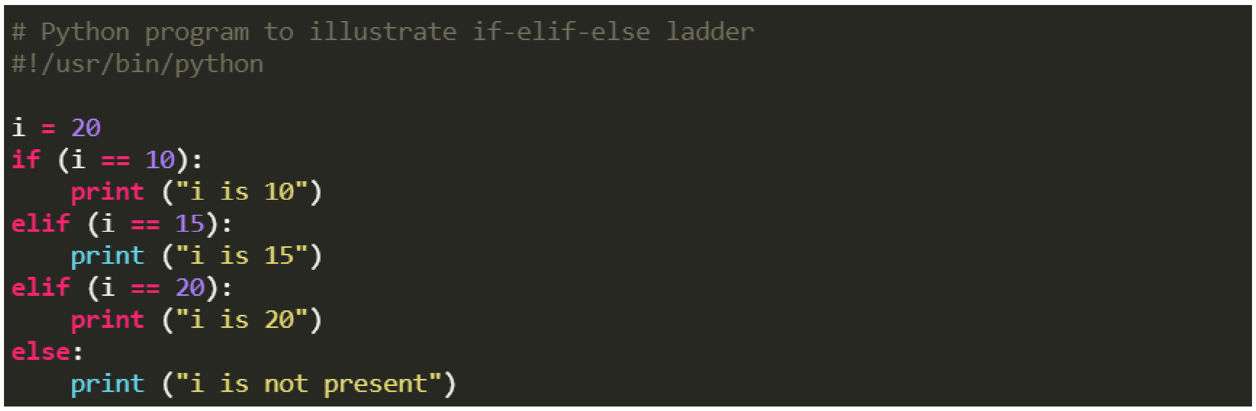
Syntax:-

```
if (condition):
    statement
elif (condition):
    statement
.
.
else:
    statement
```

Flow Chart:-



Example:-



```
# Python program to illustrate if-elif-else ladder
#!/usr/bin/python

i = 20
if (i == 10):
    print ("i is 10")
elif (i == 15):
    print ("i is 15")
elif (i == 20):
    print ("i is 20")
else:
    print ("i is not present")
```

Output:

```
i is 20
```

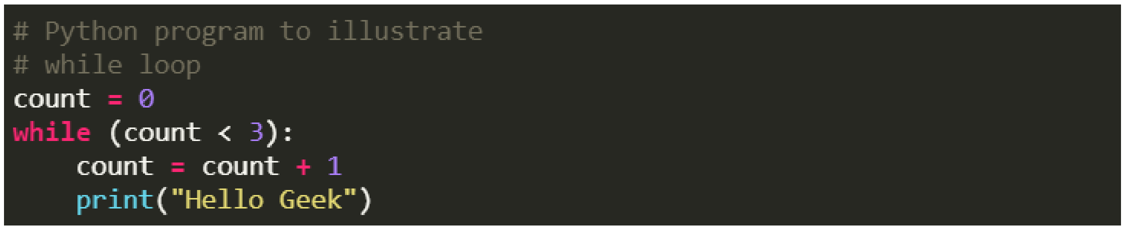
Loops in python

Python programming language provides following types of loops to handle looping requirements. Python provides three ways for executing the loops. While all the ways provide similar basic functionality, they differ in their syntax and condition checking time.

1. **While Loop:**
   In python, while loop is used to execute a block of statements repeatedly until a given a condition is satisfied. And when the condition becomes false, the line immediately after the loop in program is executed.
   Syntax :

   ```
   while expression:
      statement(s)
   ```

   All the statements indented by the same number of character spaces after a programming construct are considered to be part of a single block of code. Python uses indentation as its method of grouping statements.
   Example:



```python
# Python program to illustrate
# while loop
count = 0
while (count < 3):
    count = count + 1
    print("Hello Geek")
```

Output:

```
Hello Geek
Hello Geek
Hello Geek
```

- **Using else statement with while loops:** As discussed above, while loop executes the block until a condition is satisfied. When the condition becomes false, the statement immediately after the loop is executed.
  The else clause is only executed when your while condition becomes false. If you break out of

the loop, or if an exception is raised, it won't be executed.
If else like this:

```
if condition:
    # execute these statements
else:
    # execute these statements
```

and while loop like this are similar

```
while condition:
     # execute these statements
else:
     # execute these statements
```

```
#Python program to illustrate
# combining else with while
count = 0
while (count < 3):
    count = count + 1
    print("Hello Geek")
else:
    print("In Else Block")
```

- Output:

```
Hello Geek

Hello Geek

Hello Geek

In Else Block
```

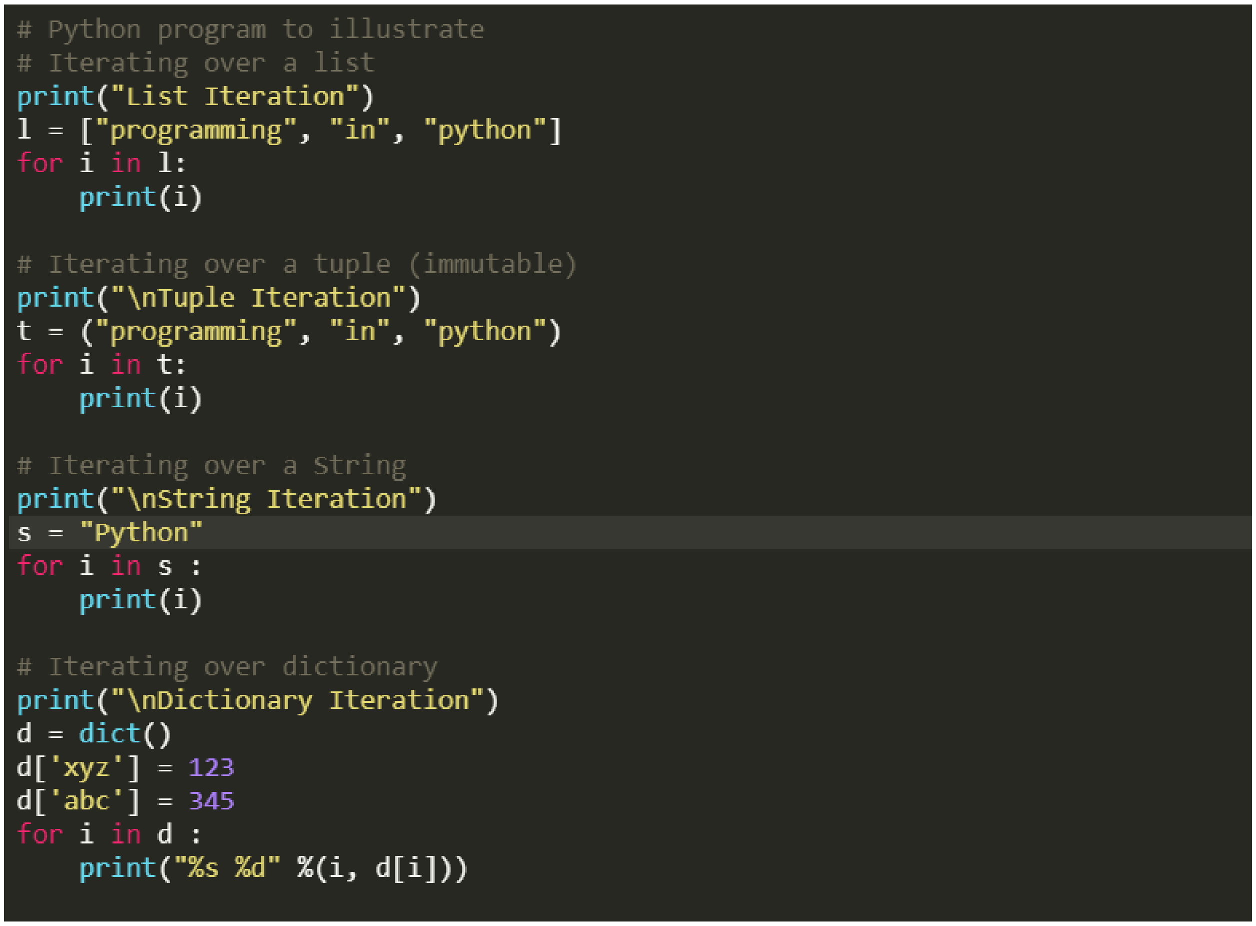
- **Single statement while block:** Just like the if block, if the while block consists of a single statement the we can declare the entire loop in a single line as shown below:

```
# Python program to illustrate
# Single statement while block
count = 0
while (count == 0): print("Hello Geek")
```

1.
    - **Note**: It is suggested **not to use** this type of loops as it is a never ending infinite loop where the condition is always true and you have to forcefully terminate the compiler.

See this for an example where while loop is used for iterators. As mentioned in the topic, it is not recommended to use while loop for iterators in python.

2. **for in Loop:** For loops are used for sequential traversal. For example: traversing a list or string or array etc. In Python, there is no C style for loop, i.e., for (i=0; i<n; i++). There is “for in” loop which is similar to for each loop in other languages. Let us learn how to use for in loop for sequential traversals.
   Syntax:
   ```
   for iterator_var in sequence:
      statements(s)
   ```
   It can be used to iterate over iterators and a range.



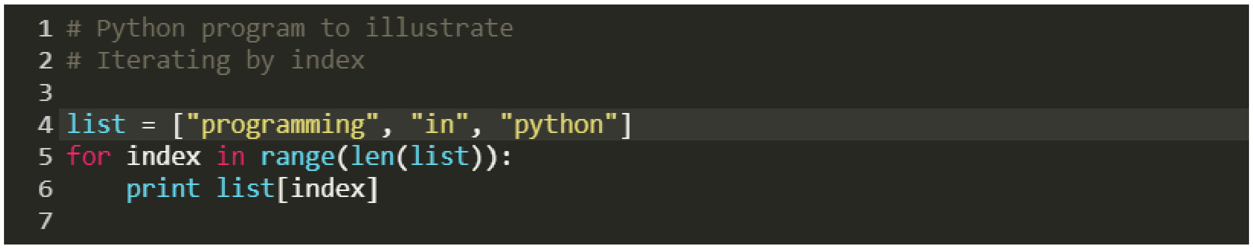
```python
# Python program to illustrate
# Iterating over a list
print("List Iteration")
l = ["programming", "in", "python"]
for i in l:
    print(i)

# Iterating over a tuple (immutable)
print("\nTuple Iteration")
t = ("programming", "in", "python")
for i in t:
    print(i)

# Iterating over a String
print("\nString Iteration")
s = "Python"
for i in s :
    print(i)

# Iterating over dictionary
print("\nDictionary Iteration")
d = dict()
d['xyz'] = 123
d['abc'] = 345
for i in d :
    print("%s %d" %(i, d[i]))
```

Output:

```
List Iteration

programming

in

python
```

```
Tuple Iteration
programming
in
python

String Iteration
p
y
t
h
o
n

Dictionary Iteration
xyz  123
abc  345
```

1. **Iterating by index of sequences**: We can also use the index of elements in the sequence to iterate. The key idea is to first calculate the length of the list and in iterate over the sequence within the range of this length.
   See the below example:



```python
# Python program to illustrate
# Iterating by index

list = ["programming", "in", "python"]
for index in range(len(list)):
    print list[index]

```

Output:

```
programming
in
```

```
python
```

2. **Using else statement with for loops:** We can also combine else statement with for loop like in while loop. But as there is no condition in for loop based on which the execution will terminate so the else block will be executed immediately after for block finishes execution. Below example explains how to do this:

```
# Python program to illustrate
# combining else with for

list = ["programming", "in", "python"]
for index in range(len(list)):
    print list[index]
else:
    print "Inside Else Block"
```

Output:

```
programming

in

python
```

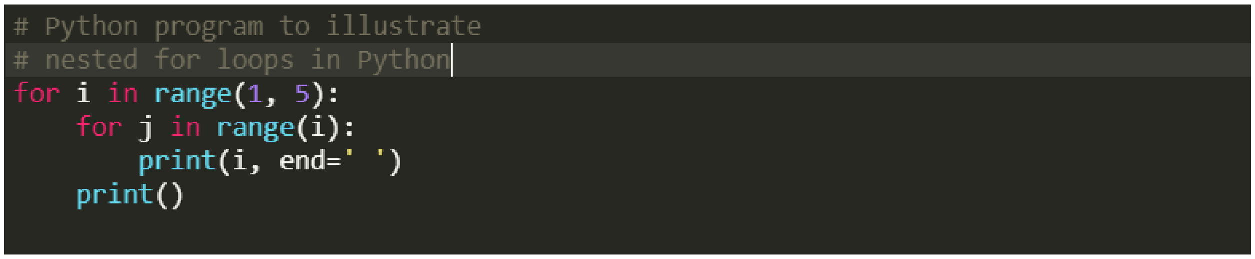
5. **Using else statement with for loops:** We can also combine else statement with for loop like in while loop. But as there is no condition in for loop based on which the execution will terminate so the else block will be executed immediately after for block finishes execution. Below example explains how to do this:

```
for iterator_var in sequence:
    for iterator_var in sequence:
        statements(s)
        statements(s)
```

The syntax for a nested while loop statement in Python programming language is as follows:

```
while expression:
    while expression:
        statement(s)
        statement(s)
```

A final note on loop nesting is that we can put any type of loop inside of any other type of loop. For example a for loop can be inside a while loop or vice versa.

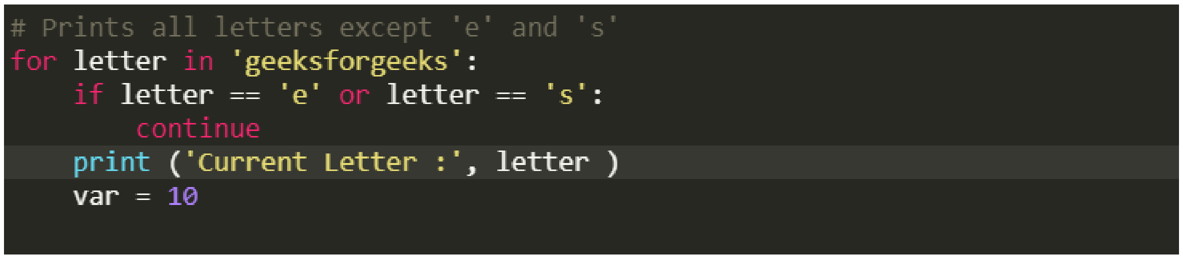


```python
# Python program to illustrate
# nested for loops in Python
for i in range(1, 5):
    for j in range(i):
        print(i, end=' ')
    print()
```

Output:

```
1

2 2

3 3 3

4 4 4 4
```

6. **Loop Control Statements:** Loop control statements change execution from its normal sequence. When execution leaves a scope, all automatic objects that were created in that scope are destroyed. Python supports the following control statements.
    - **Continue Statement:** It returns the control to the beginning of the loop.

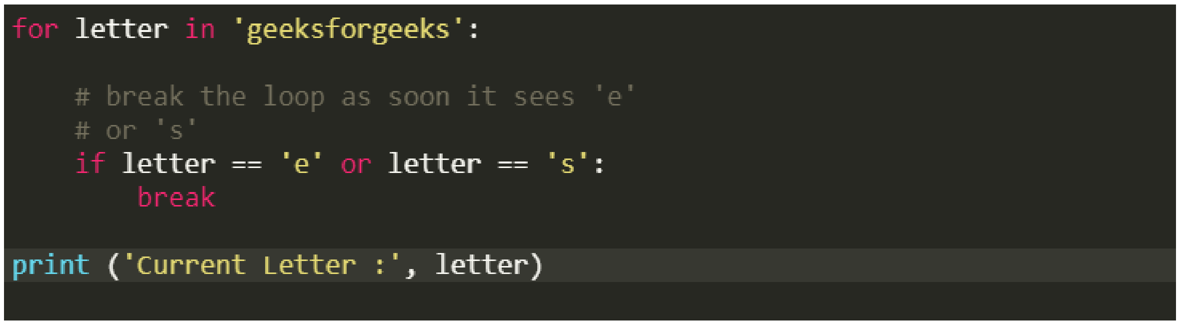


```python
# Prints all letters except 'e' and 's'
for letter in 'geeksforgeeks':
    if letter == 'e' or letter == 's':
        continue
    print ('Current Letter :', letter )
    var = 10
```

Output:

```
Current Letter : g

Current Letter : k

Current Letter : f

Current Letter : o

Current Letter : r

Current Letter : g

Current Letter : k
```

- **Break Statement:** It brings control out of the loop

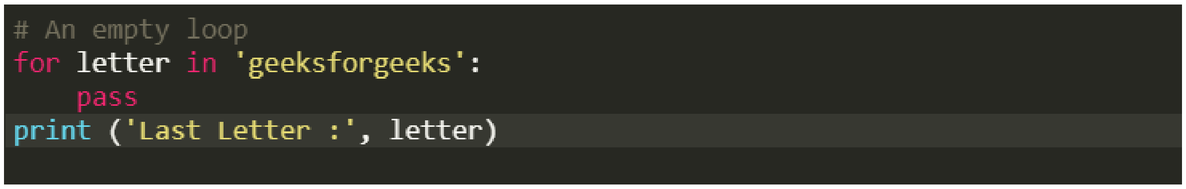


```
for letter in 'geeksforgeeks':

    # break the loop as soon it sees 'e'
    # or 's'
    if letter == 'e' or letter == 's':
        break

print ('Current Letter :', letter)
```

Output:

```
Current Letter : e
```

- **Pass Statement:** We use pass statement to write empty loops. Pass is also used for empty control statement, function and classes.



```
# An empty loop
for letter in 'geeksforgeeks':
    pass
print ('Last Letter :', letter)
```

Output:

```
Last Letter : s
```

Revision questions

1. Print a list in reverse order (from last to first item) using while and for in loops.
2. Write a Python program to find those numbers which are divisible by 7 and multiple of 5, between 1500 and 2700 (both included). Go to the editor
Click me to see the sample solution

3. Write a Python program to convert temperatures to and from celsius, fahrenheit. Go to the editor
[Formula : c/5 = f-32/9 [where c = temperature in celsius and f = temperature in fahrenheit]
Expected Output :
60°C is 140 in Fahrenheit
45°F is 7 in Celsius
Click me to see the sample solution

4. Write a Python program to guess a number between 1 to 9. Go to the editor
Note : User is prompted to enter a guess. If the user guesses wrong then the prompt appears again until the guess is correct, on successful guess, user will get a "Well guessed!" message, and the program will exit.
Click me to see the sample solution

5. Write a Python program to construct the following pattern, using a nested for loop.

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

Click me to see the sample solution

6. Write a Python program that accepts a word from the user and reverse it. Go to the editor
Click me to see the sample solution

7. Write a Python program to count the number of even and odd numbers from a series of numbers. Go to the editor
Sample numbers : numbers = (1, 2, 3, 4, 5, 6, 7, 8, 9)
Expected Output :

```
Number of even numbers : 5
Number of odd numbers : 4
```

Click me to see the sample solution

8. Write a Python program that prints each item and its corresponding type from the following list.
Sample List : datalist = [1452, 11.23, 1+2j, True, 'w3resource', (0, -1), [5, 12], {"class":'V', "section":'A'}]
Click me to see the sample solution

9. Write a Python program that prints all the numbers from 0 to 6 except 3 and 6.
Note : Use 'continue' statement.
Expected Output : 0 1 2 4 5
Click me to see the sample solution