# 05_Control_Flow_Statements

April 11, 2020

## 1 Control Flow Statements

### 1.1 1. If

if some_condition:

`statement(s)`

[1]:
```
x = 12
if x >10:
    print ("Hello")
```

```
Hello
```

### 1.2 2. If-else

if some_condition:

`statement(s)`

else:

`statement(s)`

[2]:
```
x = 5
if x > 10:
    print ("hello")
else:
    print ("world")
```

```
world
```

### 1.3 3. if-elif

if some_condition:

`statement(s)`

elif some_condition:

`statement(s)`

else:

```
statement(s)
```

[3]:
```
x = 5
y = 10
if x > y:
    print ("x>y")
elif x < y:
    print ("x<y")
else:
    print ("x=y")
```

```
x<y
```

if statement inside a if statement or if-elif or if-else are called as nested if statements.

[4]:
```
x = 12
y = 15
if x > y:
    print ("x>y")
elif x < y:
    print ("x<y")
    if x==10:
        print ("x=10")
    else:
        print ("invalid")
else:
    print ("x=y")
```

```
x<y
invalid
```

## 1.4 4. Loops

### 1.4.1 4.1 For

for variable in something:

```
statement(s)
```

[5]:
```
for a in range(5):
    print (a)
```

```
0
1
2
3
4
```

In the above example, i iterates over the 0,1,2,3,4. Every time it takes each value and executes the statement(s) inside the loop. It is also possible to iterate over a nested list illustrated below.

[6]:
```
for a in range(1,5):
    print (a)
```

```
1
2
3
4
```

[7]:
```
# displaying the elements of a list
a=[1,2,4,5]
for i in a:
    print(i)
```

```
1
2
4
5
```

[8]:
```
list_of_lists = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
for list1 in list_of_lists:
        print (list1)
```

```
[1, 2, 3]
[4, 5, 6]
[7, 8, 9]
```

A use case of a nested for loop in this case would be,

[9]:
```
list_of_lists = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
for list1 in list_of_lists:
    print(list1)
    for x in list1:
        print (x)
```

```
[1, 2, 3]
1
2
3
[4, 5, 6]
4
5
6
[7, 8, 9]
7
8
9
```

### 1.4.2 4.2 While

while some_condition:

`statement(s)`

[10]:
```python
i = 1
while i<5:
    print(i ** 2)
    i = i+1
print('Bye')
```

```
1
4
9
16
Bye
```

### 1.4.3 Example: Finding factorial

[11]:
```python
n=5
f=1
if n<0:
    print("does not exist")
elif n==0:
    print("factorial of 0 is 1")
else:
    for i in range(1, n+1):
        f=f*i
    print("factorial is:",f)

    #1*2*3*4*5
```

```
factorial is: 120
```

## 1.5 5. Break

As the name says. It is used to break out of a loop when a condition becomes true when executing the loop.

[12]:
```python
for i in range(10):
    print (i)
    if i>5:
        break
```

```
0
1
2
3
4
```

```
5
6
```

### 1.6 6. Continue

This continues the rest of the loop. Sometimes when a condition is satisfied there are chances of the loop getting terminated. This can be avoided using continue statement.

[13]:
```python
for i in range(5):
    if(i==3):
        continue
    print(i)
```

```
0
1
2
4
```

[14]:
```python
for i in range(10):
    if i>4:
        print ("The end.")
        continue
    elif i<7:
        print (i)
```

```
0
1
2
3
4
The end.
The end.
The end.
The end.
The end.
```

### 1.7 7. The pass Statement

The body of a Python compound statement cannot be empty—it must contain at least one statement. The pass statement, which performs no action, can be used as a placeholder when a statement is syntactically required but you have nothing specific to do.

[15]:
```python
for char in 'Python':
    if (char == 'h'):
        continue
    print("Current character: ", char)
```

```
Current character:  P
Current character:  y
Current character:  t
Current character:  o
Current character:  n
```

[16]:
```python
for char in 'Python':
    if (char == 'h'):
        pass
    print("Current character: ", char)
```

```
Current character:  P
Current character:  y
Current character:  t
Current character:  h
Current character:  o
Current character:  n
```

[ ]:
```

```