Module 1:

Control Flow Statements

By Ninad Gaikwad

Reference - “Core Python Programming”
Dr. R. Nageshwara Rao
Dreamtech Press

Control Statements

- Needed to repeat a group of statements
- Directly jump from one statement to another
- Control statements available in python:
    1. if statement
    2. if…else statement
    3. if…elif…else statement
    4. while loop
    5. for loop
    6. else suite
    7. break statement
    8. continue statement
    9. pass statement
    10. assert statement
    11. return statement

if and if…else

6.1 if statement

- Execute one or more statements depending on whether a given statement is true or false.
- Eg.

```
str = ‘yes’
if str == ‘yes’:
    print(‘yes’)
    print(‘This is what you said’)
    print(‘Your response is good’)
```

- Indentation is very important in python. Statements with the same indentation belong to the same group called ‘suite’. By default python uses 4 spaces for indentation.

6.2 if…else statement

- Will execute one set of instructions (if block) if the condition is true. Will execute other set of statements (else block) if the condition is false.
- Eg.

```
str = ‘yes’
if str == ‘yes’:
    print(‘Yes’)
else:
    print(‘No’
```

if…elif….else statement

6.3 if…elif….else statement

- Used to check multiple conditions and execute statements based on those conditions.
- Eg.

```
# Print numbers between one and five
x = int( input(‘ Enter a digit: ’) )
if x == 0: print(“zero”)
elif x == 1: print(“Two”)
elif x == 2: print(“Three”)
elif x == 3: print(“Four”)
elif x == 4: print(“Five”)
else: print(“Not a number between one and five”)
```

6.4 while loop

- Useful for running a group of statements repeatedly based on whether a condition is true or false.
- Eg.

```
# To display numbers from 1 to 10
x = 1
while x<=10:
    print(x)
    x+=1
print(“End”)
```

for loop

6.5 for loop

- Useful to iterate over the elements of a sequence like string, list, tuple, range etc.
- Eg

```
# Display numbers 1 to 10  in descending order
for x in range(10, 0, -1):
        print(x)
```

- Infinite loops:
  - A loop that never terminates is called an infinite loop
  - while loop has a greater chance of turning into an infinite loop if the index variable is not incremented / decremented
  - To break the infinite loop ‘ctrl+c’ is used
- Nested Loops:
  - A loop inside another loop is called a nested loop
  - It could be for example: while inside for or a for inside for etc.
- Eg Display * in equilateral triangle form using nested loops:

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

- Eg

```
# Program to display numbers from 1 to 100 in proper format
for x in rang(1, 11):
        for y in range(1, 11):
                print(‘{:8}’.format(x*y), end=’’)
        print()
```

Else suit and break statement

6.6 The else suite

- else statement can be used along with loops in python
- else suite will always be executed irrespective of whether the statements in the loop are executed or not.
- To skip execution of else break should be used
- Eg

```
g = [1,2,3,4,5]
s = int(input('Enter the number to search: '))
for i in g:
    if i == s:
        print ('yes')
        break
else:
    print('No')
```

6.7 The break statement

- Used inside a loop to break out of the loop
- Eg is shown above

Continue and Pass statements

6.8 The continue statement

- Skips the rest of the statements and goes to the beginning of the loop
- Eg x=0

```
while x<10:
    x+=1
    print('x1= ',x)
    if x>5:
        continue
    print('x2= ',x)
print('Out of loop')
```

6.9 The pass statement

- Used inside an if…else statement to represent no operation
- Used when we need a statement syntactically but do not want to do any operation
- Eg num = [1, 2, 3, -4, -5, -6, -7, 8, 9]

```
for i in num:
    if(i>0):
        pass
    else:
        print(i)
```

Assert and Return statements

6.10 The assert statement

- Useful to check if a particular condition is fulfilled.
- Eg

```
# Exceptions can be handled using ‘try except’ statements
x = int(input(“Enter number greater than 0: ”))
assert x>0, “wrong input entered”
print(“U entered: ”, x)
Or
x = int(input(“Enter number greater than 0: ”))
try:
        assert(x>0)
        print(“U entered: ”, x)
except AssertionError:
        print(“Wrong input entered”)
```

6.11 The return statement

- Used in returning the results obtained from a function
- Eg

```
def sum(a, b):
        return a+b
```

Experiment 1:

Q: Write a python program accepting two numbers as command line arguments, to swap those numbers and check if the first number is positive or negative or zero.

A: Step 1: Create ArgumentParser class object

Step 2: Add two arguments

Step 3: Retrieve and convert arguments passed to the program

Step 4: Swap and display the numbers

Step 5: Use if…elif to check if the number is positive, negative or zero