Python Programming

by Narendra Allam

Copyright 2019

Chapter 3

Control Structures

Topics Covering

Control structures:

- if statement
- if - else statement
- if - elif statement
- Nested if-else
- Multiple if
- Which control structure to choose?
- Looping statements
- Exercise Programs

Control structures are used to control the flow of execution. Sometimes we may require to skip the execution of some statements. Sometimes we may want to execute two blocks of statements exclusively. Sometimes may need to execute a block of statements repeatedly. Below are the control statements we are going to discuss now.

1. if
2. if-else
3. if-elif ladder
4. multiple if
5. nested if-else
6. while loop
7. for loop

if statement:

Syntax:

```
if condition:
    statements
```

Program: Read a number from keyboard, and if the number is even, print 'EVEN'.

In [1]:

```
x = 20
y = -30
z = x + y
print('y is {1} x is {0} and sum is {2} and {1} is -ve'.format(x, y, z))
```

```
y is -30 x is 20 and sum is -10 and -30 is -ve
```

Program: Check the given number is multiple of 3 and ends with the digit 5.

In [2]:

```
x = int(input("Enter x:"))

if x%3 == 0 and x%10 == 5:
    print("YES")

print('The End')
```

```
Enter x:45
YES
The End
```

Indentation: Python identifies code-blocks using indentation. All the statements which needs to be exceuted when condition is True, should be placed after 'if condition:' preceeded by one tab(generally 4 spaces). All the statements after 'if' statement with one tab indentation, are considered as if block.

---

if-else statement: When we want to execute two blocks of code exclusively we use if-else statement. Only one block of code is executed all the time.

Syntax:

```
if condition:
    # statements...
else:
    #statements...
```

Program: Read a number from keyboard, and if the number is even, print 'EVEN' else print 'ODD'.

In [3]:

```
x = int(input("Enter x:"))

if x%2 == 0:
    print("Even")
else:
    print("Odd")

print('The End')
```

```
Enter x:24
Even
The End
```

if-elif ladder - more exclusive cases

When we have mulitple exclusive cases, we use if-elif ladder.

Syntax:

```
if condition:
    # statements...
elif condition:
    # statements...
elif condition:
    # statements...

else:
    # statements
```

Program: Read a character from keyboard. If the given input is,

'a' - print 'Apple'
'b' - print 'Ball'
'c' - print 'Cat'
'd' - print 'Dog'
Any other character, print 'Unknown'

In [4]:

```
char = input("Enter character:")

if char == 'a':
    print ("Apple")
elif char == 'b':
    print ("Ball")
elif char == 'c':
    print ("Cat")
elif char == 'd':
    print ("Dog")
else: # if non of the if's are true, else gets executed
    print ("Unknown")

print ('The End') # This is just to know that program ended.
```

```
Enter character:b
Ball
The End
```

Multiple if: a special case

Even though it is not a seperate control statement, there is a special use case for this. When the given input falls under multiple categories we use mutiple if's.

Program: Which of the following numbers [3, 7, 11] are the factors for a given number

In [5]:

```
x = int(input("Enter x:"))

if x%3 == 0 :
    print("3 is a factor")

if x%7 == 0 :
    print("7 is a factor")

if x%11 == 0 :
    print("11 is a factor")

print("The End")
```

```
Enter x:33
3 is a factor
11 is a factor
The End
```

Nested if-else: one more special case

Like multiple if, nested if-else is also a basic control structure it has a special use-case. When we have multiple sub-categories under a condition we use nesed if-else.

Program: Read a number from keyboard,
if the number is even square it
if it is odd multiple of 5 cube it
if it is odd non-multiple of 5 mutiply with 10

In [6]:

```
x = int(input("Enter x:"))

if x%2 == 0:
    print (x**3)
else:
    if x%5 == 0:
        print (x**2)
    else:
        print (x*10)

```

```
Enter x:17
170
```

Looping statements

When a block of statements is required to be executed, repeatedly, we use looping staments. There are three looping statements in python.

1. while and while-else
2. for and for-else

In [7]:

```
i = 1
```

In [8]:

```
print(i%7)
i += 1
```

```
1
```

In [9]:

```
i
```

Out[9]:

```
2
```

While loop:

As log as condition is true, block of statements under while are executed repeatedly. Once condition is false, loop stops exection and control goes immediate next statement after while block.

Syntax:

```
while condition:
    statements
```

Using while-else: *else* block is executed after successful completion of while loop. While loop should complete all of its iterations to reach else. If break is executed , else will not be executed.

Program: Print first 5 natural numbers.

In [10]:

```
i = 1
while i <= 5:
    print(i)
    i += 1
```

```
1
2
3
4
5
```

In [11]:

```
i = 1
while i <= 100:
    if i%3 == 0 and i%10 == 5:
        print(i)
    i += 1
```

```
15
45
75
```

What is the value of i, After loop exit?

In [12]:

```
print('Value of i, after loop exit:', i)
```

```
Value of i, after loop exit: 101
```

In [13]:

```
i = 0
s = 'Apple'
while i < len(s):
    print(s[i])
    i += 1
```

```
A
p
p
l
e
```

Program: Reverse the given number.

In [14]:

```
n = int(input("Enter x:"))
b = n
rev = 0

while n > 0:
    r = n%10
    rev = rev*10 + r
    n //= 10

print ('Reverse of {} is {}'.format(b, rev))
```

```
Enter x:123456
Reverse of 123456 is 654321
```

Program: Magic number of the given number. Magic number means sum all the digits, until you get single digit sum.

In [15]:

```
n = int(input("Enter x:"))
b = n
s = n

while s > 9:
    n = s
    s = 0
    while n > 0:
        r = n%10
        s += r
        n //= 10

print ('Magic number of {} is {}'.format(b, s))
```

```
Enter x:98765
Magic number of 98765 is 8
```

Program: Find the multiples of 17 below 100 and their Count.

In [16]:

```
i = 1
c = 0
print('Multiples of 17 below 100: ')
while i <= 100:
    if i%17 == 0:
        print(i)
        c += 1
    i += 1
print('Multiples count:', c )
```

```
Multiples of 17 below 100:
17
34
51
68
85
Multiples count: 5
```

In [17]:

```
n = int(input('Enter n:'))
i = 2
rt = int(n ** 0.5)

while i <= rt:
    if n%i == 0:
        print("Not Prime")
        break
    i += 1
else:
    print("Prime")
```

```
Enter n:35
Not Prime
```

Program: Print prime numbers between 501 to 600.

In [18]:

```
n = 501
while n <= 600:
    i = 2
    rt = n**0.5
    while i <= rt:
        if n%i ==0:
            break
        i+=1
    else:
        print (n, end =', ')
    n += 1
```

```
503, 509, 521, 523, 541, 547, 557, 563, 569, 571, 577, 587, 593, 599,
```

Program: Check the given number is perfect or not. A Perfect number is a number whose sum of the factors(excluding iteslef) should be equal to itself.

In [19]:

```
n = int(input("Enter n:"))
i = 1
s = 0
while i <= n//2:
    if n%i ==0:
        s += i
    i+=1

if s == n:
    print ("Perfect")
else:
    print ("Not perfect")
```

```
Enter n:35
Not perfect
```

Program: Count numbers which are multiples of 3 and ending with 5 below 100.

In [20]:

```
i = 1
c = 0
print("Multiples of 3 and ending with 5 below 100:")
while i <= 100:
    if i%3 == 0 and i%10 == 5:
        print(i)
        c += 1
    i += 1
print("Multiples count: ",c)
```

```
Multiples of 3 and ending with 5 below 100:
15
45
75
Multiples count:  3
```

Program: Print all the numbers which end with 3 below 50.

In [21]:

```
i = 1
while i <= 50:
    if i%10 == 3:
        print(i)
    i += 1
```

```
3
13
23
33
43
```

Program: Factors of given number and their number

In [22]:

```
n = int(input('Enter n value:'))
c = 0
i = 1
print('Factors of {}: '.format(n))
while i <= n:
    if n%i == 0:
        print(i)
        c += 1
    i += 1
print('Factors count: ',c)
```

```
Enter n value:35
Factors of 35:
1
5
7
35
Factors count:  4
```

With the help of the above program we can check the given number is Prime or Not Prime

In [23]:

```
n = int(input('Enter n:'))
c = 0
i = 1
while i <= n:
    if n%i == 0:
        c += 1
    i += 1

if c == 2:
    print("Prime")
else:
    print("Not Prime")
```

```
Enter n:35
Not Prime
```

There will not be any factors for a number after n//2. Right?

In [24]:

```
n = int(input('Enter n:'))
c = 0
i = 2
while i <= n//2:
    if n%i == 0:
        c += 1
    i += 1

if c == 0:
    print("Prime")
else:
    print("Not Prime")
```

```
Enter n:35
Not Prime
```

Theorem: A prime number doesn't have any factors from 2 to square root of itself.

In [25]:

```
n = int(input('Enter n:'))
c = 0
i = 2
rt = int(n ** 0.5)
while i <= rt:
    if n%i == 0:
        c += 1
    i += 1

if c == 0:
    print("Prime")
else:
    print("Not Prime")
```

```
Enter n:35
Not Prime
```

Using break: In the above program, if condition at line 6, is true, that means we can tell the number is not prime, and we do not need to continue remaining iterations of the loop. this is where we can use **break**. break statement breaks the execution of while or for loops, control transferred to immediate next statement after while.

In [26]:

```
# Using break
n = int(input("Enter Value:"))
rt = int(n ** 0.5)
i = 2
while i <= rt:
    if n%i == 0:
        print("Not Prime")
        break
    i += 1
else:
    print ("Prime")
```

```
Enter Value:35
Not Prime
```

In [27]:

```
# Using break
n = int(input("Enter Value:"))
rt = int(n ** 0.5)
flag = True
for i in range(2, rt+1):
    if n%i == 0:
        flag = False
        print ("Not Prime")
        break
if flag:
    print ("Prime")
```

```
Enter Value:35
Not Prime
```

for loop:

Unlike other programming languages, for loop in python completely works on iteration protocol. iterable in the below syntax should provide a next value in each iteration to iter_var.

Syntax:

```
for iter_var in iterable:
    statements
```

Note: for loop is also used to iterate on any sequence type like string etc

Using for-else: 'else' block gets executed on the successful completion of for loop

In [28]:

```
n = int(input("Enter Value: "))
rt = int(n**0.5)

for i in range(2, rt+1):
    if n%i == 0:
        print ("Not Prime")
        break
else:
    print ("Prime")
```

```
Enter Value: 35
Not Prime
```

In [29]:

```
l = [4, 2, 1, 5, 7]
for x in l:
    print(x)
```

```
4
2
1
5
7
```

Using range() function in for loops

range() is a function which produces a value in each iteration in the range given.

Syntax:

```
range(start, end, step)
```

Some examples of using range() function:

In [30]:

```
for x in range(5): # if we pass one value to range, it treats it as 'end'. defau
    print(x)
```

```
0
1
2
3
4
```

Program: Printing first 6 natural numbers using **for**

In [31]:

```
for x in range(1, 6): # if we pass two values to range, assumes and 'start' and
    print(x)
```

```
1
2
3
4
5
```

In [32]:

```
for x in range(1, 11, 2):
    print(x)
```

```
1
3
5
7
9
```

This is equivalent to below while loop.

In [33]:

```
i = 1
while i < 10:
    print(i)
    i += 2
```

```
1
3
5
7
9
```

In [34]:

```
for i in range(1, 1+1):
    print('*')
```

```
*
```

In [35]:

```
for i in range(4):
    for j in range(3):
        print('*', end=' ')
```

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

continue:- continue skips the execution to next iteration of the loop.

Program:- Sum all the elements in the range 1 to 10, which are non-multiples of 3

In [36]:

```
s = 0
for x in range(1, 11):
    if x%3 == 0:
        continue
    s += x
print ("sum of non-multiples of 3: ", s)
```

```
sum of non-multiples of 3:  37
```

Printing in the same line

Each print statement takes the control to the net line. If you to keep the control in the same line, use end='' as the last argument of the print statement line.

In [37]:

```
print('Apple')
print('Orange')
```

```
Apple
Orange
```

In [38]:

```
print('Apple', end=' ')
print('Orange')
```

```
Apple Orange
```

Program: Print the below trianlge using for loop.

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

In [39]:

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

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

In [40]:

```
for i in range(0, 5):
    for j in range(5, i, -1):
        print(j, end=' ')
    print()
```

```
5 4 3 2 1
5 4 3 2
5 4 3
5 4
5
```

In [41]:

```
for i in range(5, 0, -1):
    for j in range(1, i+1):
        print(i, end=' ')
    print()
```

```
5 5 5 5 5
4 4 4 4
3 3 3
2 2
1
```

Program: Print the below trianlge using for loop.

```
1
2 2
3 3 3
4 4 4 4
5 5 5 5 5
```

In [42]:

```
for i in range(1,6):
    for j in range(1, i+1):
        print(i, end=' ')
    print()
```

```
1
2 2
3 3 3
4 4 4 4
5 5 5 5 5
```

Program: Print the below trianlge using for loop.

```
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
```

In [43]:

```
for i in range(1,6):
    for j in range(1, i+1):
        print(j, end=' ')
    print()
```

```
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
```

Program: Print the below trianlge using for loop.

```
5
4 4
3 3 3
2 2 2 2
1 1 1 1 1
```

In [44]:

```
for i in range(5, 0, -1):
    for j in range(5, i-1, -1):
        print(i, end=' ')
    print()
```

```
5
4 4
3 3 3
2 2 2 2
1 1 1 1 1
```

Program: Print the below trianlge using for loop.

```
5
5 4
5 4 3
5 4 3 2
5 4 3 2 1
```

In [45]:

```
for i in range(5, 0, -1):
    for j in range(5, i-1, -1):
        print(j, end=' ')
    print()
```

```
5
5 4
5 4 3
5 4 3 2
5 4 3 2 1
```

Program: Print the below trianlge using for loop.

```
1
2 3
4 5 6
7 8 9 10
11 12 13 14 15
```

In [46]:

```
c = 1
for i in range(1, 6):
    for j in range(1, i+1):
        print(c, end=' ')
        c += 1
    print()
```

```
1
2 3
4 5 6
7 8 9 10
11 12 13 14 15
```

Practice programs

Program 1: A Company insures its drivers in the following cases

1. If the driver is married
2. If the driver is unmarried, male and above 30 years of age
3. If the driver is unmarried, female and above 25 years of age
   in all other cases, the driver is not insured. If the marital status, gender, age of the drivers are the inputs, write a program to determine whether the driver is insured or not.(use 'nested - if')

Solution:

In [47]:

```
marital_status = input('Enter marital status[m/u]:')
gender = input('Enter gender[m/f]:')
age    = int(input('Enter age:'))

if marital_status == 'm':
    print ('Insured')
else:
    if (gender == 'm' and age >= 30) \
    or (gender == 'f' and age >= 25):
        print ('Insured')
    else:
        print ('Not Insured')
```

```
Enter marital status[m/u]:u
Enter gender[m/f]:m
Enter age:750000
Insured
```

Program 2: Indian income tax calcuation:
0% tax - upto 5.0 lac

20% tax - till 10 lac
30% tax - for above 10 lac

In [48]:

```
actual_salary = float(input('Enter salary:'))

tax = 0.0
rem = actual_salary

if actual_salary > 1000000:
    slab = rem - 1000000
    tax = slab * 0.3
    rem -= slab

if actual_salary > 500000:
    slab = rem - 500000
    tax += slab * 0.2
    rem -= slab



print('tax for {} is {}'.format(actual_salary, tax))
```

```
Enter salary:750000
tax for 750000.0 is 50000.0
```

Program 3: Write a program to read a character and find out whether it is uppercase or lowercare

Solution:

In [49]:

```
ch = input('Enter a charcater:')

if ch.isdigit():
    print('Digit Character')
elif ch.islower():
    print('Lower case Character')
elif ch.isupper():
    print('Upper case Character')
else:
    print('Special Character')

print('The End')
```

```
Enter a charcater:f
Lower case Character
The End
```

Excercise

Program 1: Write a programe to read the characters continuosly until '$' is given and display the number of characters entered.
Program 2: Write a program to print the uppercase letter of a given lowercase
Program 3: Write a program to check whether the given input is digit or lowercase character or uppercase character or a special character(use 'if-else-if' ladder)

Program 4: Write a program to print all prime numbers between 100 to 200 using only for and for-else looping statements.

Program 5: Print all perfect numbers below 100