Python Programming

RGM College of Engineering & Technology (Autonomous)

Department of Computer Science & Engineering

Academic Year : 2020-2021

FLOW CONTROL STATEMENTS - 3

Guido Van Rossum

Learning Mantra

If you really strong in the basics, then remaining things will become so easy.

Agenda:

1. Nested Loops
2. Transfer Statements

Nested Loops

- Sometimes we can take a loop inside another loop, which are also known as nested loops.

Eg:

```
for i in range(3):
    for j in range(2):
        print('Hello')
```

Output:

Hello
Hello
Hello
Hello
Hello
Hello

Eg:

```
for i in range(4):
        for j in range(3):
                print('i = {} j = {}'.format(i,j))
```

Output:

```
i = 0 j = 0
i = 0 j = 1
i = 0 j = 2
i = 1 j = 0
i = 1 j = 1
i = 1 j = 2
i = 2 j = 0
i = 2 j = 1
i = 2 j = 2
i = 3 j = 0
i = 3 j = 1
i = 3 j = 2
```

Eg: Q. Write a program to display *'s in Right angled triangle form.

```
n = int(input("Enter number of rows:"))
for i in range(1,n+1):
    for j in range(1,i+1):
        print("*",end=" ")
    print()
```

Output:

```
Enter number of rows:6
*
* *
* * *
* * * *
* * * * *
* * * * * *
```

Alternative way:

```
n = int(input("Enter number of rows:"))
for i in range(1,n+1):
    print("* " * i)
```

Output:

```
Enter number of rows:7
*
* *
* * *
* * * *
* * * * *
* * * * * *
* * * * * * *
```

Eg: Q. Write a program to display *'s in pyramid style (also known as equivalent triangle).

```
n = int(input("Enter number of rows:"))
for i in range(1,n+1):
    print(" " * (n-i),end="")                # Equivalent Triangle form
    print("* "*i)
```

Output:

```
Enter number of rows:7
      *
     * *
    * * *
   * * * *
  * * * * *
 * * * * * *
* * * * * * *
```

Transfer Statements

i) break:

- We can use break statement inside loops to break loop execution based on some condition.

Eg:

```
for i in range(10):
        if i==7:
                print("processing is enough..plz break")
                break
        print(i)
```

Output:

```
0
1
2
3
4
5
6
processing is enough..plz break
```

Eg:

```
cart=[10,20,600,60,70]
for item in cart:
    if item>500:
        print("To place this order insurance must be required")
        break
    print(item)
```

Output:

```
10
20
To place this order insurance must be required
```

ii) continue:

- We can use continue statement to skip current iteration and continue next iteration.

Eg 1: To print odd numbers in the range 0 to 9.

```
for i in range(10):
    if i%2==0:
        continue
    print(i)
```

Output:

```
1
3
5
7
9
```

Eg:

```
cart=[10,20,500,700,50,60]
for item in cart:
if item >= 500:
        print("We cannot process this item :",item)
        continue
print(item)
```

Output:

```
10
20
We cannot process this item : 500
We cannot process this item : 700
50
60
```

Eg:

```
numbers=[10,20,0,5,0,30]
for n in numbers:
    if n==0:
        print("Hey how we can divide with zero..just skipping")
        continue
print("100/{} = {}".format(n,100/n))
```

Output:

```
100/10 = 10.0
100/20 = 5.0
Hey how we can divide with zero..just skipping
100/5 = 20.0
Hey how we can divide with zero..just skipping
100/30 = 3.3333333333333335
```

Loops with else block:

- Inside loop execution, if break statement not executed ,then only else part will be executed.
- else means loop without break.

Eg:

```
cart=[10,20,30,40,50]
for item in cart:
    if item>=500:
        print("We cannot process this order")
        break
    print(item)
else:
    print("Congrats ...all items processed successfully")
```

```
10
20
30
40
50
Congrats ...all items processed successfully
```

Eg:

```
cart=[10,20,600,30,40,50]
for item in cart:
    if item>=500:
        print("We cannot process this order")
        break
    print(item)
else:
    print("Congrats ...all items processed successfully")
```

```
10
20
We cannot process this order
```

Questions

Q 1. What is the difference between for loop and while loop in Python?

- We can use loops to repeat code execution
- Repeat code for every item in sequence ==>for loop
- Repeat code as long as condition is true ==>while loop

Q 2. How to exit from the loop?

- by using break statement

Q 3. How to skip some iterations inside loop?

- by using continue statement.

Q 4. When else part will be executed w.r.t loops?

- If loop executed without break

iii) pass statement:

- pass is a keyword in Python.
- In our programming syntactically if block is required which won't do anything then we can define that empty block with pass keyword.
- pass statement --
    - It is an empty statement
    - It is null statement
    - It won't do anything

Eg:

```
if True:                # It is invalid
```

SyntaxError: unexpected EOF while parsing

```
if True:
    pass                # It is valid

def m1():               # It is invalid

def m1():
    pass                # It is valid
```

use case of pass:

- Sometimes in the parent class we have to declare a function with empty body and child class responsible to provide proper implementation. Such type of empty body we can define by using pass keyword. (It is something like abstract method in java).

Eg:

```
for i in range(100):
    if i%9==0:
        print(i)
    else:
        pass
```

➔

```
0
9
18
27
36
45
54
63
72
81
90
99
```

iv) del statement:

- del is a keyword in Python.
- After using a variable, it is highly recommended to delete that variable if it is no longer required, so that the corresponding object is eligible for Garbage Collection.
- We can delete variable by using 'del' keyword.
- After deleting a variable we cannot access that variable otherwise we will get NameError.

Eg:

```
x = 10
del(x)
print(x)
```

NameError: name 'x' is not defined

Note:

- We can delete variables which are pointing to immutable objects. But we cannot delete the elements present inside immutable object.

Eg:

```
s="karthi"
print(s)
del s                    # valid
del s[0]                 # TypeError: 'str' object doesn't support item deletion
```

Difference between 'del' and 'None':

- In the case del, the variable will be removed and we cannot access that variable**(unbind operation**).

```
s="karthi"
del s
print(s)                    # NameError: name 's' is not defined.
```

- But in the case of None assignment the variable won't be removed but the corresponding object is eligible for Garbage Collection**(re bind operation**). Hence after assigning with None value, we can access that variable.

```
s="karthi"
s=None
print(s)                    ➔None
```

Any question?

If you try to practice programs yourself, then you will learn many things automatically

Spend few minutes and then enjoy the study

Thank You