Revision Tour

Control Flow Statements in Python

VINOD KUMAR VERMA, PGT(CS), KV OEF KANPUR &
SACHIN BHARDWAJ, PGT(CS), KV NO.1 TEZPUR

Types of Statement in Python

- Statements are the instructions given to computer to perform any task. Task may be simple calculation, checking the condition or repeating action.
- Python supports 3 types of statement:
  - Empty statement
  - Simple statement
  - Compound statement

Empty Statement

- It is the simplest statement i.e. a statement which does nothing. It is written by using keyword – pass
- Whenever python encountered pass it does nothing and moves to next statement in flow of control
- Required where syntax of python required presence of a statement but where the logic of program does. More detail will be explored with loop.

Simple Statement

- Any single executable statement in Python is simple statement. For e.g.
- Name = input(“enter your name “)
- print(name)

Compound Statement

- It represent group of statement executed as unit. The compound statement of python are written in a specific pattern:

```
Compound_Statement_Header :
        indented_body containing multiple
        simple or compound statement
```

Compound Statement has:

- Header which begins with keyword/function and ends with colon(:)
- A body contains of one or more python statements each indented inside the header line. All statement in the body or under any header must be at the same level of indentation.

Statement Flow Control

- In python program statement may execute in a sequence, selectively or iteratively. Python programming support 3 Control Flow statements:

1. Sequence
2. Selection
3. Iteration

if Statement of Python

- ‘if’ statement of python is used to execute statements based on condition. It tests the condition and if the condition is true it perform certain action, we can also provide action for false situation.
- if statement in Python is of many forms:
  - if without false statement
  - if with else
  - if with elif
  - Nested if

Simple "if"

- In the simplest form if statement in Python checks the condition and execute the statement if the condition is true and do nothing if the condition is false.
- **Syntax:**

```
if condition:
        Statement1
        Statements ....
```

All statement belonging to if must have same indentation level

*** if statement is <u>compound statement</u> having <u>header</u> and a body containing <u>intended statement</u>.*

Points to remember with “if”

- It must contain valid condition which evaluates to either True or False
- Condition must followed by Colon (:) , it is mandatory
- Statement inside if must be at same indentation level.

Example - 1

Input monthly sale of employee and give bonus of 10% if sale is more than 50000 otherwise bonus will be 0

```
bonus = 0
sale = int(input("Enter Monthly Sales :"))
if sale>50000:
        bonus=sale * 10 /100
print("Bonus = " + str(bonus))
```

if with else

- **if with else** is used to test the condition and if the condition is **True it perform certain action** and **alternate course of action if the condition is false**.
- **Syntax:**

```
if condition:
        Statements
else:
        Statements
```

Example - 2

Input Age of person and print whether the person is eligible for voting or not

```
age = int(input("Enter your age "))
if age>=18:
        print("Congratulation! you are eligible for voting ")
else:
        print("Sorry! You are not eligible for voting")
```

if with elif

- if with elif is used where multiple chain of condition is to be checked. Each elif must be followed by condition: and then statement for it. After every elif we can give else which will be executed if all the condition evaluates to false
- Syntax:

```
if condition:
        Statements
elif condition:
        Statements
elif condition:
        Statements
else:
        Statement
```

Example - 3

Input temperature of water and print its physical state

```
temp = int(input("Enter temperature of water "))
if temp>100:
        print("Gaseous State")
elif temp<0:
        print("Solid State")
else:
        print("Liquid State")
```

Nested if

- In this type of “if” we put if within another if as a statement of it. Mostly used in a situation where we want different else for each condition. Syntax:

```
if condition1:
        if condition2:
                statements
        else:
                statements
elif condition3:
        statements
else:
        statements
```

Storing Condition

- In a program if our condition is complex and it is repetitive then we can store the condition in a name and then use the named condition in if statement. It makes program more readable.
- For e.g.
- x_is_less=y>=x<=z
- y_is_less=x>=y<=z
- Even = num%2==0

Python Loop Statements

- To carry out repetition of statements Python provide 2 loop statements
  - Conditional loop (while)
  - Counting loop (for)

range() function

- Before we proceed to for loop let us understand **range()** function which we will use in for loop to repeat the statement to **n number** of times.
- Syntax:
  - range(lower_limit, upper_limit)
- The range function generate set of values from **lower_limit** to **upper_limit-1**

range() function

- For e.g.
- **range(1,10)** will generate set of values from **1-9**
- range(0,7) will generate [0-6]
- Default step value will be +1 i.e.
- range(1,10) means (1,2,3,4,5,6,7,8,9)

range() function

- To change the step value we can use third parameter in range() which is step value
- For e.g.
- **range(1,10,2)** now this will generate **value [1,3,5,7,9]**
- Step value can be in –ve also to generate set of numbers in reverse order.
- range(10,0) will generate number as [10,9,8,7,6,5,4,3,2,1]

range() function

- To create list from ZERO(0) we can use
- **range(10)** it will generate **[0,1,2,3,4,5,6,7,8,9]**

Operators in and not in

- The operator *in* and *not in* is used in *for* loop to check whether the value is in the range / list or not
- For e.g.

```
>>> 5 in [1,2,3,4,5]
True
>>> 5 in [1,2,3,4]
False
```

```
>>>'a' in 'apple'
True
>>>'national' in 'international'
True
```

for loop

- for loop in python is used to create a loop to process items of any sequence like List, Tuple, Dictionary, String
- It can also be used to create loop of fixed number of steps like 5 times, 10 times, n times etc using **range()** function.

Example – for loop with List

```
School=["Principal","PGT","TGT","PRT"]
for sm in School:
        print(sm)
```

Example – for loop with Tuple

```
Code=(10,20,30,40,50,60)
for cd in Code:
        print(cd)
```

Let us understand how for loop works

```
Code=(10,20,30,40,50,60)
for cd in Code:
        print(cd)
```

for loop with string

```
for ch in 'Plan':
        print(ch)
```

The above loop product output

```
P
l
a
n
```

for with range()

Let us create a loop to print all the natural number from 1 to 100

```
for i in range(1,101):
        print(i,end='\t')
```

** here end='\t' will cause output to appear without changing line and give one tab space between next output.

while loop

- While loop in python is conditional loop which repeat the instruction as long as condition remains true.
- It is entry-controlled loop i.e. it first check the condition and if it is true then allows to enter in loop.
- while loop contains various loop elements: **initialization**, **test condition**, **body of loop** and **update statement**

while loop elements

1. **Initialization :** it is used to give starting value in a loop variable from where to start the loop
2. **Test condition :** it is the condition or last value up to which loop will be executed.
3. **Body of loop :** it specifies the action/statement to repeat in the loop
4. **Update statement :** it is the increase or decrease in loop variable to reach the test condition.

Example of simple while loop

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

Jump Statements – break & continue

break statement in python is used to **terminate the containing loop** for any given condition. Program resumes from the statement immediately after the loop

Continue statement in python is used to **skip the statements below continue statement** inside loop and **forces the loop to continue with next value**.

Example – break

```
for i in range(1,20):
        if i % 6 == 0:
                break
        print(i)
print("Loop Over")
```

The above code produces output

```
1
2
3
4
5
Loop Over
```

when the value of i reaches to 6 condition will becomes True and loop will end and message "Loop Over" will be printed

Example – continue

```
for i in range(1,20):
        if i % 6 == 0:
                continue
        print(i, end = ‘ ‘)
print(“Loop Over”)
```

The above code produces output

```
1 2 3 4 5 7 8 9 10 11 13 14 15 16 17 19
Loop Over
```

when the value of i becomes divisible from 6, condition will becomes True and loop will skip all the statement below continue and continues in loop with next value of iteration.

Loop .. **else** .. Statement

- Loop in python provides else clause with loop also which will execute when the loop terminates normally i.e. when the test condition fails in while loop or when last value is executed in for loop **but not when break terminates the loop**

Example ("else" with while)

```
i=1
while i<=10:
        print(i)
        i+=1
else:
        print("Loop Over")
```

Output

```
1
2
3
4
5
6
7
8
9
10
Loop Over
```

Example ("else" with for)

```
names=["allahabad","lucknow","varanasi","kanpur","agra","ghaziabad"
,"mathura","meerut"]
city = input("Enter city to search: ")
for c in names:
        if c == city:
                print("City Found")
                break
else:
        print("Not found")
```

Output

```
Enter city to search : varanasi
City Found
Enter city to search : unnao
Not found
```

Generating Random numbers

- Python allows us to generate random number between any range
- To generate random number we need to import **random** library
- Syntax: (to generate random number)
  - random.randint(x,y)
  - it will generate random number between x to y
  - For e.g. : **random.randint(1,10)**
  - **Refer to code page no. 156**

STRING MANIPULATION

VINOD KUMAR VERMA, PGT(CS), KV OEF KANPUR &
SACHIN BHARDWAJ, PGT(CS), KV NO.1 TEZPUR

Program to read string and print in reverse order

```
string1 = input("Enter any string ")
print("The Reverse of ", string1 , " is :")
length=len(string1)
for ch in range(-1,(-length-1),-1):
        print(string1[ch])
```

The above code will print

```
Enter any string: karan
n
a
r
a
k
```

Program to input string and print short form

```
string=input("Enter any string ")
print(string[0],".",end='')
for ch in range(1,len(string)):
  if string[ch]==' ':
    print(string[ch+1],".",end='')
```

String operators

Two basic operators + and * are allowed

+ is used for concatenation (joining)

e.g. **"shakti" + "man"** OUTPUT: **shaktiman**

* Is used for replication (repetition)

e.g. **"Bla"** * 3 OUTPUT: **BlaBlaBla**

*Note: you cannot multiply string and string using **
*Only number*number or string*number is allowed*

MEMBERSHIP OPERATORS

- Membership operators (in and not in) are used to check the presence of character(s) in any string.

Example	Output
‘a’ in ‘python’	False
‘a’ in ‘java’	True
‘per’ in ‘operators’	True
‘men’ in ‘membership’	False
‘Man’ in ‘manipulation’	False
‘Pre’ not in ‘presence’	True

Comparison operators

- We can apply comparison operators (==, !=,>,<,>=,<=) on string. Comparison will be character by character.

str1='program'

str2='python'

str3='Python'

Comparison of string will be based on ASCII code of the characters

Example	Output
str1==str2	False
str1!=str2	True
str2=='python'	True
str2>str3	True
str3<str1	True

Characters	Ordinal/ ASCII code
A-Z	65-90
a-z	97-122
0-9	48-57

DETERMINING ORDINAL / UNICODE OF A SINGLE CHARACTER

Python allows us to find out the ordinal position single character using ord() function.

>>>ord('A') **output will be 65**

We can also find out the character based on the ordinal value using chr() function

>>>chr(66) **output will be 'B'**

String slicing

```
>>> str1="wonderful"
>>> str1[0:6]
'wonder'
>>> str1[0:3]
'won'
>>> str1[3:6]
'der'
>>> str1[-1:-3]
''
>>> str1[-3:-1]
'fu'
>>> str1[-3:0]
''


'Wnefl'
```

```
>>> str1[3:3]
''
>>> str1[3:4]
'd'
>>> str1[-5:-2]
'erf'
>>> str1[:-2]
'wonderf'
>>> str1[:4]
'wond'
>>> str1[-3:]
'ful'
>>>str1[::-1]
lufrednow
```

Reverse of string

Interesting string slicing

For any index position n: str1[:n]+str1[n:] will give you the original string

>>>str1="wonderful"

>>>str1[:n]+str[n:] ***output will be wonderful***

String slicing will never return error even if you pass index which is not in the string . For e.g.

>>>str1[10] **will give error, but**

>>>str1[10:15] **will not give error but return empty string**

Python string manipulation function

Function name	Purpose	Example
String.capitalize()	Return a copy of string with first character as capital	>>>'computer'.capitalize() Computer

String.find(str[,start[,end]])	Return lowest index of str in given string , -1 if not found	Str="johny johny yes papa"
		Sub="johny"
		
		Str.find(Sub)
		0
		Str.find(Sub,1)
		6
		Str.find('y',6,11)
		10
		Str.find('pinky')
		-1

String.isalnum()	Return True if the string is alphanumeric character	'hello123'.isalnum() True

Python string manipulation function

Function name	Purpose	Example

String.isalnum()		S="ravi@gmail.com
		S.Isalnum()
		False

String.isalpha()	Return True if string contains only alphabets characters	Str1="hello"
		Str2="hello123"
		Str1.isalpha()
		True
		Str2.isalpha()
		False

String.isdigit()	Return True if string contains only digits	s1="123"
		s1.isdigit()
		True
		Str2.isdigit()
		False

String.islower()	Return True if all character in string is in lower case	Str1.islower() True

Python string manipulation function

Function name	Purpose	Example

String.isspace()	Return True if only whitespace characters in the string	S=" "
		S.isspace()
		True
		S2="'
		S2.isspace()
		False

String.upper()	Return True if all characters in the string are in upper case	S="HELLO"
		S.isupper()
		True

String.lower()	Return copy of string converted to lower case characters	S="INDIA"
		S.lower()
		india

String.upper()	Return copy of string converted to upper case characters	S="india"
		S.lower()
		INDIA

Python string manipulation function

Function name	Purpose	Example

String.lstrip([chars])	Returns a copy of string with leading characters removed.
	If no characters are passed then it removes whitespace
	It removes all combination of given characters i.e. if we pass “The” characters i.e. if we pass “The” then its combination – The, Teh, heT, ehT, he, eh, h, e, etc will be removed
		string=" hello"
		string.lstrip()
		'hello'
		string2="National"
		string2.lstrip('nat')
		National
		string2.lstrip('Nat')
		ional
		string2.lstrip('at')
		National
		string2.lstrip('Na')
		tional
		string2.lstrip('atN')
		ional

String.rstrip([chars])	Returns a copy of string with trailing characters removed. Rest is same as lstrip	“saregamapadhanisa”.rstrip(‘ania’) saregamapadha