RV College of Engineering®
Autonomous Institution Affiliated to Visvesvaraya Technological University, Belagavi
Approved by AICTE New Delhi

Introduction to Python Programming

UNIT-3

For Loops, Strings , Lists, Tuples, and Dictionaries

Prof. Narasimha Swamy S
Department of AIML
RV College of Engineering
Bengaluru-59

Go, Change the World

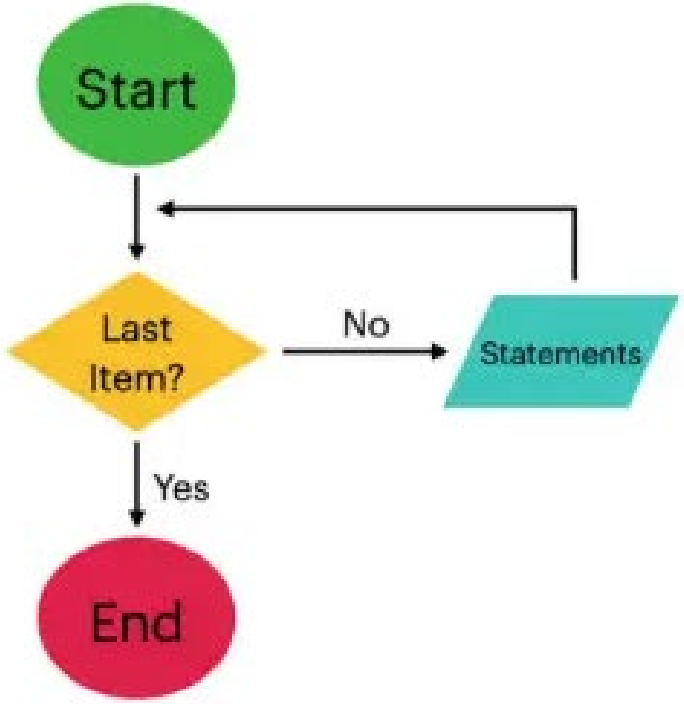
Outline

- → Using For Loops
- → Counting with the for loops
- → Indexing String
- → Understanding the String immutability
- → Building a New String
- → Slicing the Strings
- → Using Tuples
- → Using Compound Conditions
- → Using Lists
- → List Methods
- → Dictionaries

For Loop

- A for loop is used for iterating over a particular sequence (can be a list, a tuple, a dictionary, a set, or a string)
- This is less like the for keyword in other programming languages
- With the for loop a set of statements, once for each item in a list, tuple, set etc can be executed
- Used for sequential traversal
- Syntax:

```
for iterator_var in sequence:
    statements(s)
```



For Loop (Contd.)

Example 1

```
n = 4
for i in range(0, n):
    print(i)
```

Output

0 1 2 3

- range() function returns a sequence of numbers, starting from 0 by default, and increments by 1 (by default), and stops before a specified number

- Iterating the List elements Directly

Example 2

```
for i in [4, 3, 2, 1] :
    print(i)
print('stop!')
```

Output

4 3 2 1 stop!

For Loop (Contd.)

Example 3

```
friends = ['Joseph', 'Glenn', 'Sally']
for friend in friends :
        print(‘Hello:', friend)
        print('Done!’)
```

Output:

```
Hello: Joseph
Hello: Glenn
Hello: Sally
Done!
```

For Loop (Contd.)

Nested for Loop

- Loop inside the loop is called as Nested loops
- Inner loops complete the execution first then outer loops executes

Example 1

```
for i in [1, 2, 3]:
        print(i)

        for j in [4, 5, 6]:
                print(i*j)
```

Example 2

```
fruits = ["Apple", "Strawberry", "Cherry"]
features = ["Red", "Tasty", "Big"]
for i in fruits:
        print("-----" + i + "-----")
        for j in features:
                print(i + " is "+ j)
```

Control Statements in For Loop

- Break
- Continue
- Pass

Control Statements in For Loop (Contd.)

Break Statements

Example 1

```
Names = ["Amar", "Arjun", "Anil"]
for i in Names:
        print(i)
```

Example 2

```
Names = ["Amar", "Arjun", "Anil"]
for i in Names:
        print(i)
        if(i == "Arjun"):
                break
print(i)
```

Control Statements in For Loop (Contd.)

Continue Statements

Example 1

```
Names = ["Amar", "Arjun", "Anil"]
for i in Names:
        print(i)
```

Example 2

```
Names = ["Amar", "Arjun", "Anil"]
for i in Names:
                if(i == "Arjun"):
                continue
print(i)
```

Control Statements in For Loop (Contd.)

Pass Statements

- **pass** is a keyword used when the user doesn't want any code to execute
- So user can simply place **pass** where empty code is not allowed, like in loops, function definitions, class definitions, or in if statements

Example 1

```
for i in range(1, 10, 2):
        print(i)
```

Example 2

```
for i in range(1, 10, 2):
```

Example 3

```
for i in range(1, 10, 2):
        pass
```

Strings in Python

- Accessing Strings
- Basic Operations
- String slices
- Function and Methods

Strings in Python (Contd.)

- String Initialization
  - a = 'Welcome, to RVCE!'
  - b = "Welcome, to RVCE!"
  - c = """Welcome,

    to RVCE! """
- Accessing characters in a string
  - a[1]

Strings in Python (Contd.)

- Substring

```
a = 'welcome, to rvce !’
print(a)
print(a[2:5])
print(a[2:])
print(a[:3])
print(a[-1])
```

Strings in Python (Contd.)

String Manipulation Functions

- capitalize() → Capitalizes first letter of string
- center(width, fillchar) → Returns a space-padded string with the original string cantered to a total of width columns.
- count(str, beg= 0,end=len(string)) → Counts how many times str occurs in string or in a substring of string if starting index beg and ending index end are given
- endswith(suffix, beg=0, end=len(string)) → Determines if string or a substring of string (if starting index beg and ending index end are given) ends with suffix; returns true if so and false otherwise
- find (str, beg=0 end=len(string)) → Determine if str occurs in string
- isalnum() → Returns true if string has at least 1 character and all characters are alphanumeric and false otherwise

Strings in Python (Contd.)

String Manipulation Functions

- isdigit() → Returns true if string contains only digits and false otherwise
- islower() → Returns true if string has at least 1 cased character and all cased characters are in lowercase and false otherwise
- isspace() → Returns true if string contains only whitespace characters and false otherwise

Strings in Python (Contd.)

String Manipulation Functions

```
str = “rvce"
output=str.capitalize()
print("The resultant string is:", output)
```

```
str = "Welcome to RV College of Engineering.“
output=str.center(40, '.’)
print("The string after applying the center() function is:", output)
```

```
str = "Hello! Welcome to RV College of Engineering“
substr = "i"
print("The number of occurrences of the substring in the input string are: ", str.count(substr, 3, 30))
```

Strings in Python (Contd.)

String Manipulation Functions

```
str = "Hello! RV College of Engineering.“
suffix = “ring"
result=str.endswith(suffix, 27, 32)
print("The input string ends with the given suffix:",result)
```

```
str1 = "Hello! RV Colleg of Engineering"
str2 = " RV";
result= str1.find(str2)
print("The index where the substring is found:", result)
```

```
str = “RVCE"
result=str.isalnum()
print("Are all the characters of the string alphanumeric?", result)
```

Strings in Python (Contd.)

String Manipulation Functions

```
str = "1235"
result=str.isdigit()
print("Are all the characters of the string digits?", result)
```

```
str = "Hello! RVCE”
result=str.islower()
print("Are all the cased characters in the string lowercases?", result)
```

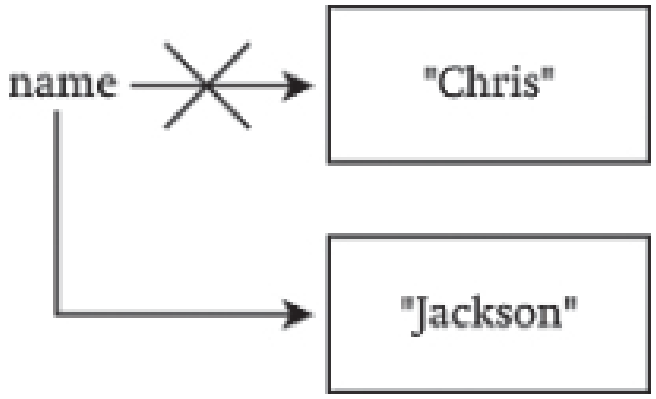
```
str = “R V C E\t\t\t"
result=str.isspace()
 print("Are all the characters of the string spaces?", result)
```

Strings in Python (Contd.)

Understanding the String Immutability

```
name = “chris”
print(name)

name = “jackson”
print(name)
```




```
name = “chris”
print(name)

name[0] = “C”
```

```
chris
Traceback (most recent call last):
  File "main.py", line 4, in <module>
    name[0] = "C"
TypeError: 'str' object does not support item assignment


...Program finished with exit code 1
Press ENTER to exit console.
```

List in Python

- Lists are used to store multiple items using a single variable.
- Lists is a built-in data types
- Lists are used to store collections of data
- List items are ordered, changeable, and allow duplicate values
- List items are indexed, the first item has index [0], the second item has index [1] and so on
- List items can be of any data type

List in Python (Contd.)

Example 1

```
list = [ 'abcd', 786 , 2.23, 'john', 70.2 ]
tinylist = [123, 'john']

print (list)                # Prints complete list
print (list[0])             # Prints first element of the list
print (list[1:3])           # Prints elements starting from 2nd till 3rd
print (list[2:])            # Prints elements starting from 3rd element
print (tinylist * 2)        # Prints list two times
print (list + tinylist)     # Prints concatenated lists
print(list[-1])             # Prints the last element in the list
```

List in Python (Contd.)

Method	Description	Example
append()	add an item to the end of the list	list.append(100)
extend()	Appends the list1 to list2 or vice-versa	list1.extend(list)
insert()	inserts an item at the specified index	list.insert(1, “RVCE”)
remove()	removes given item present in the list	list.remove(‘abcd’)
pop()	returns and removes item present at the given index	list.pop(1)
clear()	removes all items from the list	list.clear()
index()	returns the index of the first matched item	list.index(5)
count()	returns the count of the specified item in the list	list.count(786)
sort()	sort the list in ascending/ descending order	list.sort(reverse=True)
reverse()	reverses the item of the list	list.reverse()
copy()	returns the shallow copy of the list	list1 = list.copy()

Tuple in Python

- A tuple is a order sequence of values
- The values can be of any type, and they are indexed by integers, so in that respect tuples are a lot like lists
- The important difference is that the order of the tuples are immutable
- Syntactically, a tuple is a comma-separated values

```
t = 'a', 'b', 'c', 'd', 'e'
```

- Although it is not necessary, it is common to enclose tuples in parentheses

```
t = ('a', 'b', 'c', 'd', 'e')
```

Tuple in Python (Contd.)

Creating Tuples

 # Concatenating the Tuple

```
Tup1 = (‘Ajith’,  ’Amar’, ’Abhinav’)
Tup2 = (1, 2, 3, 4, 5)
Tup3 = (1.4, 12.3, 12.1, 15.01)
Tup4 = ‘(100, 100.1, 200.8, ‘RVCE’, ‘Bengaluru’)

# Printing all the Tuple Elements
print(Tup1)
print(Tup2)
print(Tup3)
print(Tup4)
```

1

Note: Tuples are immutable which means you cannot update or change the values of tuple elements

```
Tup1 = (‘Ajith’,  ’Amar’, ’Abhinav’)
print(Tup1[0])

Tup1 [0] = “RVCE”

# Printing all the Tuple Elements
print(Tup1)
```

2

```
Tup1 = (10, 20, 30, [100, 200, 300], 40, 50)

Tup1[3][1] = 1000
```

3

Tuple in Python (Contd.)

 # Concatenating the Tuple

```
Tup1 = (‘Ajith’,  ’Amar’, ’Abhinav’)
Tup2 = (1, 2, 3, 4, 5)
Tup3 = Tup1 + Tup2

# Printing all the Tuple Elements

print(Tup3)
```

 # Deleting the Tuple

```
Tup1 = (‘Ajith’,  ’Amar’, ’Abhinav’)
print(Tup1)

del Tup1
print(Tup1)
```

 # Tuple Operation

```
Tup1 = (‘Ajith’,  ’Amar’, ’Abhinav’)
Tup2 = (1, 2, 3, 4, 5)

print(len(Tup1))

print(Tup2*2)

print(‘Anu’ in Tup1)
print(‘Ajith’ in Tup1)
print(5 in Tup2)

For x in Tup3:
        print(x)
```

Tuple in Python (Contd.)

Indexing, Slicing & Built-in Functions

```
# Indexing the Tuple
Tup = (‘Apple’,  ’Doll’, ’Camera’, ‘Ball’, ‘Ice-cream’)

print(Tup[3])
print(Tup[-3])
print(Tup[1:])
print(Tup[:3])
```

```
# Indexing the Tuple
Tup1 = (‘Apple’,  ’Doll’, ’Camera’, ‘Ball’)
Tup2 = (1,2,3,4,5)

print(max(Tup1))
print(max(Tup2))
print(min(Tup1))
print(min(Tup2))
print(max(Tup1, key = len))

List = [1, 2, 3, 4, 5, ‘A’]
ex = tuple(list)
print(ex)
print(tuple(ex))
```

Dictionary in Python

- A dictionary is a collection of key-value pairs subject to the constraint that all the keys should be unique
- Keys and Values are separated using the symbol **':'**
- Duplicates are not allowed in dictionary
- Each key maps to a value
- The association of a key and a value is called a key-value pair
- Example

  { 'grapes' : 'green' , ' apple' : 'red' }

  # grapes and apple are keys with their corresponding values green and red respectively

  Personal Information

  {"Name":"John", "Age":18}

Dictionary in Python (Contd.)

Creating the Dictionary

```
D = {'Name': 'Raghu', 1:28, 'Designation': 'Manager'}
print("D[Name]: ", D['Name'])
print("D[1]: ", D[1])
print("D[Designation]: ", D['Designation'])
```

Accessing the Dictionary Elements

```
FruitColor = {'Apple': 'red', 'Banana': 'Yellow', 'Kiwi':
'Green''}
print(FruitColor)
print(FruitColor.get(Banana))


FruitColor.keys()          # List the Keys
FruitColor.values()        # List the Values
```

Updating the Dictionary

```
print("Before Updating")
print("D[Name]: ", D['Name'])
print("D[1]: ", D[1])
print("D[Designation]: ", D['Designation'])


D['Name'] = "Ranjan"
D[1] = 56
D[Designation] = 'Team Lead'

print("After Updating")
print("D[Name]: ", D['Name'])
print("D[1]: ", D[1])
print("D[Designation]: ", D['Designation'])
```

Thank you