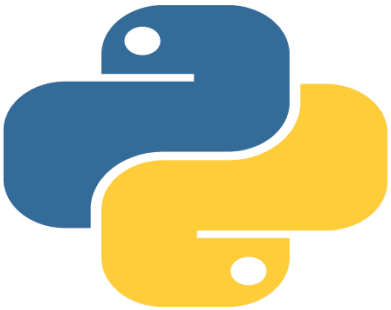

Python

Part 3

Python-array

Python does not have built-in support for arrays.An **array** is a container which can hold a fix number of items and these items should be of the same type. Each item stored in an array is called an **element** and they can be of any type including integers, floats, strings, etc.

Creating Array in Python

To create an array in Python, import the array module and use its **array() function**. We can create an array of three basic types namely integer, float and Unicode characters using this function.

Syntax:

The syntax for creating an array in Python is −

```
# importing
import array as array_name

# creating array
obj = array_name.array(typecode[, initializer])
```

Where,

typecode − The typecode character used to specify the type of elements in the array.

initializer − It is an optional value from which an array is initialized. It must be a list, a bytes-like object, or iterable elements of the appropriate type.

Example:

```
import array as arr

# creating array
Array = arr.array('i', [200, 234, 258, 400, 567])

#indexing
print (numericArray[0])
print (numericArray[1])
print (numericArray[2])
```

Output:

```
200
234
258
```

Python - Add Array Items

Python array is a mutable sequence which means they can be changed or modified whenever required. However, items of same data type can be added to an array.

Using append() method

To add a new element to an array, use the **append()** method. It accepts a single item as an argument and append it at the end of given array.

Syntax

Syntax of the append() method is as follows −

```
append(v)
```

Where

v − new value is added at the end of the array. The new value must be of the same type as datatype argument used while declaring array object.

Example:

```
import array as arr
a = arr.array('i', [1, 2, 3])
a.append(10)
print (a)
```

Output:

```
array('i', [1, 2, 3, 10])
```

Using insert() method

It is possible to add a new element at the specified index using the **insert()** method.It accepts two parameters which are index and value and returns a new array after adding the specified value.

Syntax

Syntax of this method is shown below −

```
insert(i, v)
```

Where,

i − The index at which new value is to be inserted.

v − The value to be inserted. Must be of the arraytype.

Example:

```
import array as arr
a = arr.array('i', [1, 2, 3])
a.insert(1,20)
print (a)
```

Output:

```
array('i', [1, 20, 2, 3])
```

Using extend() method

The **extend()** method belongs to Python array module. It is used to add all elements from an iterable or array of same data type.

Syntax

This method has the following syntax −

```
extend(x)
```

Where,

x − This parameter specifies an array or iterable.

Example:

```
import array as arr
a = arr.array('i', [1, 2, 3, 4, 5])
b = arr.array('i', [6,7,8,9,10])
a.extend(b)
print (a)
```

Output:

```
array('i', [1, 2, 3, 4, 5, 6, 7, 8, 9, 10])
```

Removing array items in Python

Python arrays are a mutable sequence which means operation like adding new elements and removing existing elements can be performed with ease. We can remove an element from an array by specifying its value or position within the given array.

Syntax

```
array.remove(v)
```

Example:

```
import array as arr

# creating array
Array = arr.array('i', [100, 223, 389, 401, 518])

# before removing array
print ("Before removing:",Array)
```

```
# removing array
Array.remove(389)
# after removing array
print ("After removing:",Array)
```

Output:

```
Before removing: array('i', [100, 223, 389, 401, 518])
After removing: array('i', [100, 223, 401, 518])
```

Remove Items from Specific Indices

To remove an array element from specific index, use the **pop()** method. This method removes an element at the specified index from the array and returns the element at ith position after removal.

Syntax

```
array.pop(i)
```

Where, **i** is the index for the element to be remove

Example

```
import array as arr

# creating array
numericArray = arr.array('i', [100, 223, 389, 401, 518])

# before removing array
print ("Before removing:", numericArray)
# removing array
numericArray.pop(3)
# after removing array
print ("After removing:", numericArray)
```

```
```

Output:

```
Before removing: array('i', [100, 223, 389, 401, 518])
After removing: array('i', [100, 223, 389, 518])
```

Python - Loop Arrays

Loops are used to repeatedly execute a block of code. In Python, there are two types of loops named for loop and while loop. Since the array object behaves like a sequence, you can iterate through its elements with the help of loops

Python for Loop with Array

Example:

```
import array as arr
Array = arr.array('i', [100, 402, 236, 815, 458])
for iterated_array  in Array:
  print (iterated_array)
```

Output:

```
100
402
236
815
458
```

Python while Loop with Array

In the while loop, the iteration continues as long as the specified condition is true. When you are using this loop with arrays, initialize a loop variable before entering the loop. This variable often

represents an index for accessing elements in the array. Inside the while loop, iterate over the array elements and manually update the loop variable.

Example

The following example shows how you can loop through an array using a while loop −

```
import array as arr

# creating array
a = arr.array('i', [961, 226, 156, 376, 246])

# checking the length
l = len(a)

# loop variable
idx = 0

# while loop
while idx < l:
   print (a[idx])
   # incrementing the while loop
   idx+=1
```

Output

```
961
226
156
376
246
```

Python for Loop with Array Index

We can find the length of array with built-in len() function. Use it to create a range object to get the series of indices and then access the array elements in a **for** loop.

Example:

```
import array as arr
array = arr.array('d', [5, 2, 3, 5, 4])
l = len(array)
for x in range(l):
   print (a[x])
```

Output:

```
5
2
3
5
4
```

Python - Reverse Arrays

Reversing an array is the operation of rearranging the array elements in the opposite order. There are various methods and approaches to reverse an array in Python including reverse() and reversed() method

Example

```
import array as arr

# creating array
Array = arr.array('i', [8, 9, 7, 5, 6])

print("Original array:", numericArray)
revArray =Array[::-1]
```

```
print("Reversed array:",revArray)
```

Output:

```
Original array: array('i', [8, 9, 7, 5, 6])
Reversed array: array('i', [6, 5, 7, 9, 8])
```

Python - Sort Arrays

Sort Arrays Using sorted() Method

The third technique to sort an array is with the **sorted()** function, which is a built-in function.

Example:

```
import array as arr
a = arr.array('i', [4, 5, 6, 9, 10, 15, 20])
sorted(a)
print(a)
```

Output:

```
array('i', [4, 5, 6, 9, 10, 15, 20])
```

Import in python

In Python, modules help organize code into reusable files. They allow you to import and use functions, classes and variables from other scripts. The import statement is the most common way to bring external functionality into your Python program.

Example:

```
Import math
x=math.pi
print(x)
```

Output:

```
3.14
```

Example:

```
from math import *
x=pi
print(x)
```

Output:

```
3.14
```

Example:

```
From math import pi
x=pi
print(x)
```

Output:

```
3.14
```

User defined module

Create a Python Module

To create a Python module, write the desired code and save that in a file with .py extension. Let's understand it better with an example:

Example:

Let's create a simple mathop.py in which we define two functions, one **add** and another is subtraction.

```
# A simple module, calc.py
def add(x, y):
    return (x+y)

def subtract(x, y):
    return (x-y)
```

Output:

```
<<<
```

Import module in Python

We can import the functions, and classes defined in a module to another module using the **import statement** in some other Python source file.

For example, to import the module mathop.py, we need to put the following command at the top of the script.

Syntax to Import Module in Python

```
import module
```

Importing modules in Python Example

Now, we are importing the calc that we created earlier to perform add operation.

Example:

```
# importing  module mathop.py
import mathop
print(calc.add(1, 2))
```

Output:

```
3
```

FILE HANDLING IN PYTHON

File handling in Python involves interacting with files on your computer to read data from them or write data to them. Python provides several built-in functions and methods for creating, opening, reading, writing, and closing files.

Opening a File in Python

To perform any file operation, the first step is to open the file. Python's built-in open() function is used to open files in various modes, such as reading, writing, and appending. The syntax for opening a file in Python is −

```
file = open("filename", "mode")
```

Steps for open a txt format file on python ide.

Step1:

Create a txt file on desktop(Right click mouse point on desktop then select text document give a file name)

Step2:

Open that file then write some content to it then close it

Step3:

Next step is open ide then write below program on it this is for open that created txt file on our program

Example:

```
f=open(“C://Users/lenovo/Desktop/python.txt”,r)
```

Output:

```
>>>
```

Reading file on program

Reading a file in Python involves opening the file in a mode that allows for reading, and then using various methods to extract the data from the file. Python provides several methods to read data from a file −

read() − Reads the entire file.

readline() − Reads one line at a time.

readlines − Reads all lines into a list.

Example:

```
f=open("C://Users/lenovo/Desktop/python.txt",'r')
print(f.read())
```

Output:

```
Python is an interpreter language
Python is a oops language
Easy to learn
```

Example:

```
f=open("C://Users/lenovo/Desktop/python.txt",'r')
print(f.readline())
```

Output:

```
Python is an interpreter language
```

Example:

```
f=open(“C://Users/lenovo/Desktop/python.txt”,r)
print(f.readline(12))
```

Output:

```
Python is an
```

Writing to a File in Python

Writing to a file in Python involves opening the file in a mode that allows writing, and then using various methods to add content to the file.

Example: Using append method to write content to file

```
f=open(“C://Users/lenovo/Desktop/python.txt”,’r’)
print(f.readline(12))
```

Append method for write content fo file

Example: Using the write() method

Opens a file for appending. The file pointer is at the end of the file if the file exists. That is, the file is in the append mode. If the file does not exist, it creates a new file for writing.

```
f=open("C://Users/lenovo/Desktop/python.txt",'a')
f.write("\npython is general purpose language")
f.close()
```

Output:

```
>>>
```

Example: Using the Write() method

Open a file for writing only. Overwrites the file if the file exists. If the file does not exist, create a new file for writing.

```
f=open("C://Users/lenovo/Desktop/python.txt",'w')
f.write("\npython is general purpose language")
f.close()
```

Output:

```
>>>
```

Remove file in python

You can delete a file in Python using the **os.remove()** function. This function deletes a file specified by its filename.

Syntax

Following is the basic syntax of the remove() function in Python −

```
os.remove(file_name)
```

Example:

```
import os
os.remove("C://Users/lenovo/Desktop/python.txt")
```

Output:

```
>>>
```