Installation of Python on Linux

Using the Package Manager

1.  If you are using Ubuntu Linux, at the command prompt type:

$ sudo apt-get install python3

2.  If you are using Fedora, at the command prompt type:

$ sudo yum install python3

3.  The most recent version of Python 3 will be downloaded and installed. To verify the installation, type:

$ Python

The Python prompt (>>>) will appear.

From source

You can install Python from its source code if you want to really customize the binary by setting certain options or flags during the build process.

1. Download the source from <https://www.python.org/ftp/python/3.6.2/Python-3.6.2rc1.tgz>

2.  Type the following commands to extract and install Python from its source:

$sudo tar xzf python-3.6.2.tar.xz

$./configure

$sudo make install

3. To verify the installation, in the terminal type:

$ Python3

4. The Python prompt (>>>) will appear.

Installation of Python on Mac OS X

Using a wizard-based installer

1. Go to the Python website, https://www.python.org/ftp/python/3.6.2/python-3.6.2-macosx10.6.pkg and download the required version.

2. Run the downloaded file and follow the instructions in the installation wizard.

Using Homebrew

Homebrew is a package manager that lets you install, update, and uninstall packages from the command line on the Mac OS.

1. Homebrew depends on Apple’s Xcode package, so run the following command to install Xcode first:

$ xcode-select --install

2. Next, install Homebrew by following the instructions on their website, <https://brew.sh/>

3. After installing Homebrew, from the prompt in the terminal type the following command to install Python:

$ brew install python3

4. To verify the installation, type:

$ Python3

The Python prompt (>>>) will appear.

Installation of Python on other platforms

You can install Python for other platforms such as MS-DOS, OS2 or Solaris from the Python website,<https://www.python.org/download/other/>

Alternative Python implementations

The standard distribution of Python (available on python.org) is developed in C Language and is often calledCPython. This recommended standard is tweaked to optimize its performance for specific applications. These implementations combine ease of development of Python and rich libraries of other platforms such as .net or Java. Some such alternative implementations are:

 IronPython (Python running on .NET framework. Developed in C#)

 Jython (Developed in Java. Python running on the Java Virtual Machine. Capable of using rich Java library in Python program)

 PyPy (A [fast](http://speed.pypy.org/) python implementation with a JIT compiler)

 Stackless Python (Branch of CPython supporting microthreads)

 MicroPython (Python running on micro controllers)

Commercial distributions of Python

While community versions are open source and intended to use on as is basis, commercial versions of Python ensure guarantee of service and maintenance of the products. Also, when modules and packages are installed from repositories, it may lead to version conflict in a community version. However, commercial versions ensure that the distribution is self-contained and without compatibility issues. Some of the commercial distributions of Python include:

 ActiveState ActivePython (commercial and community versions, including scientific computing modules)

 pythonxy (Scientific-oriented Python Distribution based on Qt and Spyder)

 winpython (WinPython is a portable scientific Python distribution for Windows)

 Enthought Canopy (a commercial distribution for scientific computing)

 PyIMSL Studio (a commercial distribution for numerical analysis )

 [Anaconda Python](https://store.continuum.io/cshop/anaconda) (a full Python distribution for data management, analysis and visualization of large data sets)

Keywords in Python

|  |  |  |
| --- | --- | --- |
| 'False' | 'elif' | 'lambda' |
| 'None' | 'else' | 'nonlocal' |
| 'True' | 'except' | 'not' |
| 'and' | 'finally' | 'or' |
| 'as' | 'for' | 'pass' |
| 'assert' | 'from' | 'raise' |
| 'break' | 'global' | 'return' |
| 'class' | 'if' | 'try' |
| 'continue' | 'import' | 'while' |
| 'def' | 'in' | 'with' |
| 'del' | 'is' | 'yield' |

1. Which syntax element is used to define a code block? (A code block is a group of statements that, together, perform a task.)

(1 Mark)

1. Comment
2. Triple Quotes, """ """
3. Indent
4. Curly brackets, { }

Correct Answer : c

2. Is the following statement true or false?  
  
Python is a compiler-based language.

(2 Marks)

1. True
2. False

Correct Answer : b

3. Which of the following statements is False?

(3 marks)

1. Parentheses are mandatory with the print function in Python 3.x but not in Python 2.x.
2. The raw\_input() function has been deprecated (discontinued) in Python 3.x.
3. Integer objects are long by default in Python 3.x but not in Python 2.x in which a trailing L is required.
4. In Python 3.x, 5/2 will result in an output of 2.

Correct Answer : d

4. Given the following code in IDLE, what will you see after you type print(a,b,b) and press Enter?   
  
>>> a=2  
>>>b=3  
>>>print(a,b,b) 

(1 Mark)

1. 18
2. a b b
3. 2 3 3
4. Error

5. What will you see if you type 19-4\*5 and press Enter in IDLE?

(2 Marks)

1. 759375
2. -1
3. 75
4. Error

Correct Answer : b

6. What is the correct command for displaying all the Python keywords?

(3 marks)

1. >>> import keyword  
   >>> keyword.kwlist
2. >>> keyword.python
3. >>> import keyword.kwlist  
   >>> keyword.python
4. >>> import .keyword.kwlist

7. The result of 36%5 is:

(1 Mark)

1. 1
2. 7
3. 7.2
4. 1.75

8. What will you see if you type a = 12 \* 2 and press Enter in IDLE?

(2 Marks)

1. 24
2. 144
3. Nothing
4. Error

9. There are some syntax errors in the displayed program. Of the options displayed, select the option which is the correct version of this program.  
  
#The purpose of this  
program is to display some text. #  
  
print ("Hello World")   
print ("Welcome to Internshala") #Author Internshala

(3 marks)

1. #The purpose of this  
   program is to display some text. #  
     
   print (Hello World)  
   print (Welcome to Internshala) #Author Internshala
2. #The purpose of this  
   #program is to display some text.  
     
   print ("Hello World")  
   print ("Welcome to Internshala") #Author Internshala
3. #  
   The purpose of this  
   program is to display some text.   
   #  
     
   print ("Hello World")  
   print ("Welcome to Internshala") #Author Internshala

10. Is the following statement true or false?  
  
Dynamic Typing is a feature of Python.

(1 Mark)

1. True
2. False

1. Which syntax element is used to define a code block? (A code block is a group of statements that, together, perform a task.)

(1 Mark)

1. Comment
2. Triple Quotes, """ """
3. Indent
4. Curly brackets, { }

Correct Answer : c

2. Is the following statement true or false?  
  
Python is a compiler-based language.

(2 Marks)

1. True
2. False

Correct Answer : b

3. Which of the following statements is False?

(3 marks)

1. Parentheses are mandatory with the print function in Python 3.x but not in Python 2.x.
2. The raw\_input() function has been deprecated (discontinued) in Python 3.x.
3. Integer objects are long by default in Python 3.x but not in Python 2.x in which a trailing L is required.
4. In Python 3.x, 5/2 will result in an output of 2.

Correct Answer : d

4. Given the following code in IDLE, what will you see after you type print(a,b,b) and press Enter?   
  
>>> a=2  
>>>b=3  
>>>print(a,b,b) 

(1 Mark)

1. 18
2. a b b
3. 2 3 3
4. Error

5. What will you see if you type 19-4\*5 and press Enter in IDLE?

(2 Marks)

1. 759375
2. -1
3. 75
4. Error

Correct Answer : b

6. What is the correct command for displaying all the Python keywords?

(3 marks)

1. >>> import keyword  
   >>> keyword.kwlist
2. >>> keyword.python
3. >>> import keyword.kwlist  
   >>> keyword.python
4. >>> import .keyword.kwlist

7. The result of 36%5 is:

(1 Mark)

1. 1
2. 7
3. 7.2
4. 1.75

8. What will you see if you type a = 12 \* 2 and press Enter in IDLE?

(2 Marks)

1. 24
2. 144
3. Nothing
4. Error

9. There are some syntax errors in the displayed program. Of the options displayed, select the option which is the correct version of this program.  
  
#The purpose of this  
program is to display some text. #  
  
print ("Hello World")   
print ("Welcome to Internshala") #Author Internshala

(3 marks)

1. #The purpose of this  
   program is to display some text. #  
     
   print (Hello World)  
   print (Welcome to Internshala) #Author Internshala
2. #The purpose of this  
   #program is to display some text.  
     
   print ("Hello World")  
   print ("Welcome to Internshala") #Author Internshala
3. #  
   The purpose of this  
   program is to display some text.   
   #  
     
   print ("Hello World")  
   print ("Welcome to Internshala") #Author Internshala

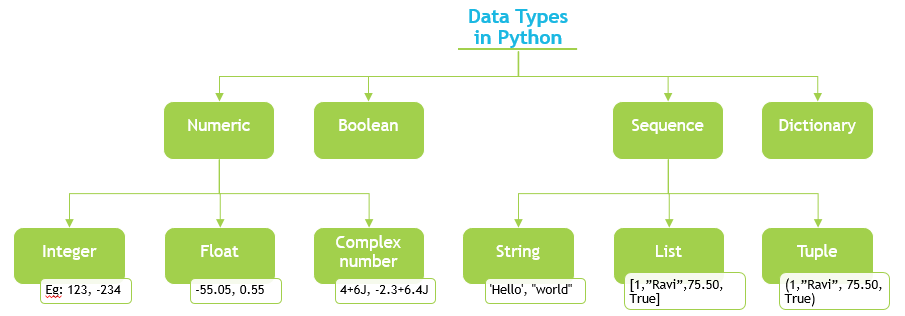
10. Is the following statement true or false?  
  
Dynamic Typing is a feature of Python.

(1 Mark)

1. True
2. False

Data Types in Python

The built-in data types in Python are displayed below.



|  |  |  |
| --- | --- | --- |
| Numeric |  | Any representation of data which has numeric value. Python identifies three types of numbers – integer, float and complex number. |
|  | Integer | Positive and negative whole numbers.  Examples: 1234, -234, 0x46 (hexadecimal number), 0O123 (octal number))  Note: In C and related programming languages such as Python, a hexadecimal number is prefixed with 0x and an octal number is prefixed with 0O. |
| Float | Real numbers with a floating point representation in which the fractional component is denoted by a decimal or scientific notation  Examples: -55.550, 0.005, 1.32E10 (scientific notation)) |
| Complex number | A number with a real and imaginary component is represented as a + bj inPython where a and b are floats and  j = √-1  ​​​Examples: 4+6j, -2.3+6.4j  Note: The common mathematical representation of a complex number uses a +bi with i being the imaginary part. But in electronics j is used because i already represent current and the next letter after i is j. |
| Boolean |  | Any representation of data which has two values denoted by True and False. |
| Sequence |  | An ordered collection of similar or different data types. The built-in Sequence data types in Python are – String, List and Tuple. |
|  | String | A collection of one or more characters put in single, double or triple quotes.  Examples: ‘Hello’, "Hello", "'Hello'", """Hello""" |
|  | List | An ordered collection of one or more data items, not necessarily of same type, put in square brackets.  Examples:  [1,"Ravi",75.50, True] |
|  | Tuple | An ordered collection of one or more data items, not necessarily of same type put in parentheses. The contents of a tuple cannot be modified – it is immutable - after the tuple is created.  Examples: (1,"Ravi", 75.50, True)  Note: Refer to the Helper Text to learn more about mutability. |
| Dictionary |  | An unordered collection of data in key:value pair form. Collection of such pairs is enclosed in curly brackets.  Example:  {1:"Superman", 2:"Wonder Woman", 3:"Thor", 4: "Hulk", 5:"Black Widow"} |

PREV

NEXT

* Helper Text
* Forum

Mutable and Immutable Objects

When a program is run, data objects in the program are stored in the computer’s memory for processing. While some of these objects can be modified at that memory location, other data objects can’t be modified once they are stored in the memory. *The property of whether or not data objects can be modified in the same memory location where they are stored is called*mutability*.*

We can check the mutability of an object by checking its memory location before and after it is modified. If the memory location remains the same when the data object is modified, it means it is mutable.

To check the memory location of where a data object is stored, we use the function, id(). Consider the following example (you can try this yourself in IDLE):

|  |  |
| --- | --- |
| >>> a=[5, 10, 15] | Assigning values to the list a. |
|  |  |
| >>> id(a) | Using the function id() to get the memory location of a. |
|  |  |
| 1906292064 | The ID of the memory location where a is stored. |
|  |  |
| >>> a[1]=20 | Replacing the second item in the list,10 with a new item, 20. |
|  |  |
| >>> print(a) | Using the print() function to verify the new value of a. |
|  |  |
| [5, 20, 15] | Verified that the value of a has changed. |
|  |  |
| >>> id(a) | Using the function id() to get the memory location of a. |
|  |  |
| 1906292064 | The ID of the memory location where a is stored. |

Notice that the memory location has not changed as the ID remains (1906292064) remains the same before and after the variable is modified. This indicates that the list is mutable, i.e., it can be modified at the same memory location where it is stored. Now, let us check if a tuple is mutable in Python.

|  |  |
| --- | --- |
| >>> b=(5, 10, 15) | Assigning values to the tuple b. |
|  |  |
| >>> b[1]=20 | Replacing the second item in the list, 10 with a new item, 20. |
|  |  |
| Traceback (most recent call last):    File "<pyshell#1>", line 1, in <module>      b[1]=20  TypeError: 'tuple' object does not support item assignment | Error explaining that a tuple does not support a modification in the items – i.e, it is immutable. |

You can verify the mutability of each of the data types in IDLE.

Immutable: numeric, string, and tuple  
Mutable: list, dictionary

Next live chat: Mon (10:00 - 11:00 PM)

Formatting Strings using Escape Sequences

You can use two or more specially designated characters within a string to format a string or perform a command. These characters are called escape sequences. An escape sequence in Python starts with a backslash (\). For example, \n is an escape sequence in which the common meaning of the letter n is literally escaped and given an alternative meaning - a new line.

Displayed here are a few common escape sequences available in python. You can try these out in IDLE or the Python prompt from the windows command prompt.

|  |  |  |  |
| --- | --- | --- | --- |
| Escape sequence | Description | Example | Result |
| \n | Breaks the string into a new line | >>> print('I designed this rhyme to explain in due time\nAll I know') | I designed this rhyme to explain in due time All I know |
| \t | Adds a horizontal tab | >>> print('Time is a \tvaluable thing') | Time is a valuable thing |
| \\ | Prints a backslash | >>> print('Watch it fly by\\ as the pendulum swings') | Watch it fly by\ as the pendulum swings |
| \' | Prints a single quote | >>> print('It doesn\'t even matter how hard you try') | It doesn't even matter how hard you try |
| \" | Prints a double quote | >>> print('It is so \"unreal\"') | It is so "unreal" |
| \a | makes a sound like a bell | >>> print('\a') | Sound effect is heard |

[ : ] Fetches characters in the range specified by two index operands separated by a colon.  
If the first operand is omitted, the range starts at zero index.  
If the second operand is omitted, the range goes up to the end of the string.  
Note: The slice starts at the first index. The slice ends one index before the second index, that is at the value of the index - 1.   
  
Example

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| T | h | e |  | B | u | r | r | o | w |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

>>> a='The Burrow' #String a

>>> a[2:7] #Starting index = 2 = e, Ending index = 7-1 = r

'e Bur'

>>> a[:6] #Starting index = 0 = T, Ending index = 6-1 = u

'The Bu'

>>> a[5:] #Starting index = 5 = u, Ending index = end of string = w

'urrow'

>>>

[in] Returns true if a character exists in the given string.  
  
Example

>>> a='Harry watched Dumbledore striding up and down in front of him, and thought. He thought of his mother, his father and Sirius. He thought of Cedric Diggory.' #String a

>>> 'v' in a #Checks if the character 'v' is present in the string, a

False

>>> 'dig' in a #Checks if the characters 'dig' are present in the string, a

False

>>> 'Dig' in a #Note that this is case-sensitive. As 'Dig' is present in 'Diggory', this returns True.

True

>>>

[not in] Returns true if a character does not exist in the given string.  
  
Example

>>> a=''' “For HIM?” shouted Snape. “Expecto Patronum!"

From the tip of his wand burst the silver doe: She landed on the office floor, bounded once across the office, and soared out of the window. Dumbledore watched her fly away, and as her silvery glow faded he turned back to Snape, and his eyes were full of tears.

“After all this time?”

“Always,” said Snape.''' #Multi-line string, a

>>> 'v' not in a #Checks that the character 'v' is not present in the string, a

False

>>> 'Red' not in a #Checks that the characters 'Red' iare not present in the string, a

True

>>> 'red' notin a #This is case sensitive. Since 'red' is present in 'soared', this returns False.

False

>>>

Format specification symbols

|  |  |  |
| --- | --- | --- |
| Format Symbol | Conversion |  |
| %c | character |  |
| %s | string conversion via str() prior to formatting | Example 1  >>> balltype='basketball' >>> result='hit' >>> print('I wondered why the %s was getting bigger. Then it %s me.' % (balltype, result)) I wondered why the basketball was getting bigger. Then it hit me. >>>   Example 2  >>> print("%20s" % ('Internshala', ))          Internshala >>>   Example 3  >>> print("%-20s" % ('Internshala', )) Internshala          >>>   Example 4  >>> print("%.5s" % ('Internshala', )) Inter >>> |
| %i | signed decimal integer |  |
| %d | signed decimal integer | >>> match=12553 >>> site='eBay' >>> print("%s is so useless. I tried to look up lighters and all they had was %d matches." % (site, match)) eBay is so useless. I tried to look up lighters and all they had was 12553 matches. >>> |
| %u | unsigned decimal integer |  |
| %o | octal integer |  |
| %x / %X | hexadecimal integer (lowercase letters) |  |

The format() Method

With Python 3.0, the format() method has been introduced for handling complex string formatting more efficiently. This method of the built-in string class provides functionality for complex variable substitutions and value formatting. This new formatting technique is regarded as more elegant. The general syntax of format() method is:

string.format(var1, var2,...)

The string itself contains placeholders {} in which values of variables are successively inserted.

>>> name="Malhar"

>>> age=23

>>> percentage=55.5

>>> "my name is {} and my age is {} years".format(name, age)

'my name is Malhar and my age is 23 years'

>>>

You can also specify formatting symbols. Only change is using colon (:) instead of %. For example, instead of %s use {:s} and instead of %d use (:d}

>>> "my name is {:s} and my age is {:d} years".format(name, age)

'my name is Malhar and my age is 23 years'

>>>

Precision formatting of numbers can be accordingly done.

>>> "my name is {:s}, age {:d} and I have scored {:6.3f} percent  
 marks".format(name, age, percentage)

'my name is Malhar, age 23 and I have scored 55.500 percent marks'

>>>

**Other Methods for String Processing**

To learn about other methods that can be used to process strings, click each on the methods below.

* [capitalize()](https://trainings.internshala.com/content/python#item1)
* [upper()](https://trainings.internshala.com/content/python#item2)
* [lower()](https://trainings.internshala.com/content/python#item3)
* [title()](https://trainings.internshala.com/content/python#item4)
* [find()](https://trainings.internshala.com/content/python#item5)
* [index()](https://trainings.internshala.com/content/python#item6)
* [count()](https://trainings.internshala.com/content/python#item7)
* [isalpha()](https://trainings.internshala.com/content/python#item8)
* [isdigit()](https://trainings.internshala.com/content/python#item9)
* [islower()](https://trainings.internshala.com/content/python#item10)
* [isupper()](https://trainings.internshala.com/content/python#item11)
* capitalize(): This method converts the first character of a string to uppercase letter.
* >>> var='internshala'
* >>> var.capitalize()
* 'Internshala'
* >>>
* upper(): This method returns a string with lowercase characters replaced by corresponding uppercase characters.
* >>> var='internshala'
* >>> var.upper()
* 'INTERNSHALA'
* >>>
* lower(): This method results in a string with uppercase characters replaced by corresponding lowercase characters.
* >>> var='INTERNSHALA'
* >>> var.lower()
* 'internshala'
* >>>
* title(): This method results in a string with the first character of each word converted to uppercase.
* >>> var='python training from internshala'
* >>> var.title()
* 'Python Training From Internshala'
* >>>
* find(): This method finds the first occurrence of a substring in another string. If not found, the method returns -1.
* >>> var='python training from internshala'
* >>> var.find('in')
* 10
* >>> var.find('on')
* 4
* >>> var.find('run')
* -1
* >>>
* The substring 'in' first occurs at the 10th position (count starts from 0), 'on' is found at the 4th position, but 'run' is not found hence returns -1.
* index(): This method is similar to find() but throws a ValueError if the substring is not found.
* >>> var='python training from internshala'
* >>> var.index('in')
* 10
* >>> var.index('run')
* Traceback (most recent call last):
* File "<pyshell#19>", line 1, in <module>
* var.index('run')
* ValueError: substring not found
* >>>
* count(): This method returns the number of occurrences of a substring in given string.
* >>> var='python training from internshala'
* >>> var.count('in')
* 3
* >>>
* isalpha(): This method returns true if all the characters in a string are alphabetic (a-z or A-Z), otherwise returns false.
* >>> var='Internshala'
* >>> var.isalpha()
* True
* >>> var='Intern shala'
* >>> var.isalpha()
* False
* >>>
* isdigit(): This method returns true if all characters in a string are digits( 0-9), if not returns false.
* >>> var='2000'
* >>> var.isdigit()
* True
* >>> var='2,000'
* >>> var.isdigit()
* False
* >>>
* islower(): This method returns true if all characters in a string are lowercase characters else returns false.
* >>> var='internshala'
* >>> var.islower()
* True
* >>> var='Internshala'
* >>> var.islower()
* False
* >>> var='intern shala'
* >>> var.islower()
* True
* >>>
* isupper(): This method returns true if all characters in a string are uppercase characters else returns false.
* >>> var='INTERN\_SHALA'
* >>> var.isupper()
* True
* >>> var='INTERNshala'
* >>> var.isupper()
* False
* >>> var='INTERN+SHALA'
* >>> var.isupper()
* True
* >>> var='1234'
* >>> var.isupper()
* False
* >>>

#### **Set Data type**

Set is also a collection data type in Python. However, it is not an ordered collection of objects, like list or tuple. Hence, indexing and slicing operations cannot be done on a set object. A set also doesn’t allow duplicate objects to be stored, where as in list and tuple, the same object can appear more than once. Even if an object is put more than once in a set, only one copy is held. Set is a Python implementation of a set as defined in Mathematics. The set object has suitable methods to perform mathematical set operations like union, intersection, difference etc. A set object contains one or more items, not necessarily of the same type which are separated by comma and enclosed in curly brackets {}.

>>> S1={1, "Ravi", 75.50}

>>> S1

{1, 75.5, 'Ravi'}

>>> type(S1)

<class 'set'>

>>> S2={10,23,40,23,50,10}

>>> S2

{40, 10, 50, 23}

>>>

##### **set() function**

Python has an in-built function set() using which set object can be constructed out of any sequence like string, list or tuple object.

>>> S1=set("Internshala")

>>> S1

{'t', 'n', 's', 'h', 'e', 'a', 'l', 'I', 'r'}

>>> S2=set([45,67,87,36,55])

>>> S2

{55, 67, 36, 45, 87}

>>> S3=set((10,25,15))

>>> S3

{25, 10, 15}

>>>

Order of elements in the set is not necessarily the same that is given at the time of assignment. Python optimizes the structure for performing operations over set as defined in mathematics. Only immutable (and hashable) objects can be a part of set object. Numbers (integer, float as well as complex), strings, and tuple objects are accepted but list and dictionary objects are not.

>>> S1={(10,10), 10,20}

>>> S1

{10, 20, (10, 10)}

>>> S2={[10,10], 10,20}

Traceback (most recent call last):

File "<pyshell#2>", line 1, in <module>

S2={[10,10], 10,20}

TypeError: unhashable type: 'list'

>>>

In first case, (10,10) is a tuple, hence it becomes part of set. In second example though, since [10,10] is a list, error message is displayed saying list is unhashable. (Hashing is a mechanism in computer science which enables quicker searching of objects in computer’s memory. <https://en.wikipedia.org/wiki/Hash_function>) Even though mutable objects are not stored in a set, set itself is a mutable object. A set object can be modified by add(), update(), remove() and discard() methods. 

* [add()](https://trainings.internshala.com/content/python#item1)
* [update()](https://trainings.internshala.com/content/python#item2)
* [clear()](https://trainings.internshala.com/content/python#item3)
* [copy()](https://trainings.internshala.com/content/python#item4)
* [discard()](https://trainings.internshala.com/content/python#item5)
* [remove()](https://trainings.internshala.com/content/python#item6)

##### **Set Operations**

As mentioned earlier, set data type in Python implements set as defined in mathematics. Various Set operations can be performed using Python’s set obect. The operators |, &, - and ^ perform union, intersection, difference and symmetric difference operations respectively. Each of these operators have a corresponding method associated with built-in set class.

* [Union](https://trainings.internshala.com/content/python#item7)
* [Intersection](https://trainings.internshala.com/content/python#item8)
* [Difference](https://trainings.internshala.com/content/python#item9)
* [Symmetric Difference](https://trainings.internshala.com/content/python#item10)

Set is a specialized data type. One of the major applications of Python is in area of mathematical computing and data analysis in which set operations are important. We may drop this discussion considering it as not for beginner (and also to curtail the size), but learner (especially who intends to go in mathematical and scientific computing) should be encouraged to explore this section. We should emphasize this and provide this as a text for further reading.