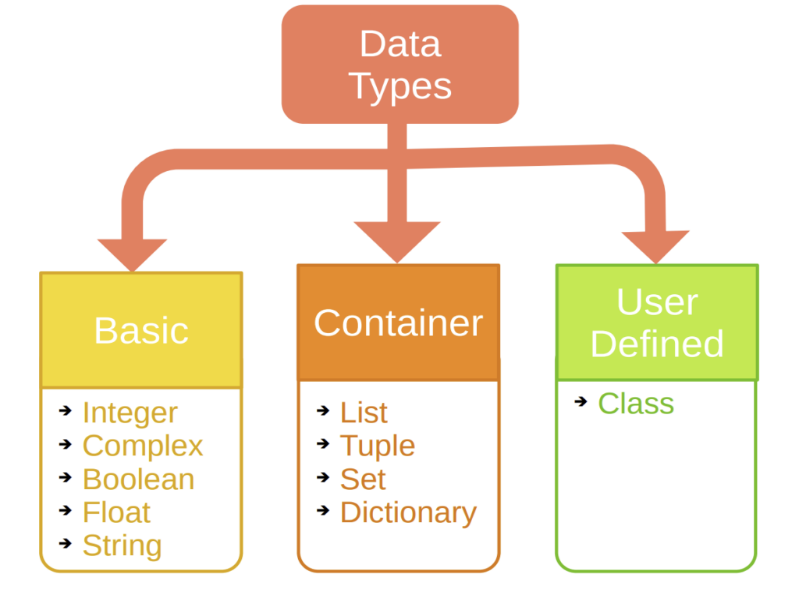
**Python:-**

* It is object oriented and high level Language.
* It is a scripting language and It is simplicity and readable .
* It used to web developing ,software developing , system scripting.

**Variables:-**

A memory location that stores data. Variables allow you to store, modify, and retrieve values throughout your program.

**Datatypes:-**



* The Basic datatypes are represents a single value.
* The container datatypes are represents a multiple values.

**Basic Example:-**

Int:-5,10,-4………………..

Float:-5.34,23.56………

String:-“hello”,’hi’……………..

Boolean:-true or false…………………

**Container Example:-**

List:-[4,5,6,]……….

Tuple:-(2,45,345,)……………..

Dictionary:-{“kallu”,”lp”…..}

Set:-{1,2,3}…….

**Type Casting:-**

Typecasting, also known as type conversion, is the process of converting a value from one data type to another datatype.

It has two types:-1. Implicit type:-Reference type to value type.

Ex:- a=10;

B=a;

2.Explicity type:-value type to reference type.

Ex:-a=10

B=10

**Operators:-**

Operators are used to perform operations on variables and values.

Different types of operators:-

* Arithmetic operators
* Assignment operators
* Comparison operators
* Logical operators
* Identity operators
* Membership operators
* Bitwise operators

**Arithmetic operators** are used with numeric values to perform common mathematical operations:

|  |  |  |
| --- | --- | --- |
| **Operator** | **Name** | **Example** |
| + | Addition | a + b = 30 |
| - | Subtraction | a – b = -10 |
| \* | Multiplication | a \* b = 200 |
| / | Division | b / a = 2 |
| % | Modulus | b % a = 0 |
| \*\* | Exponent | a\*\*b =10\*\*20 |
| // | Floor Division | 9//2 = 4 |

Ex:-

a = 21

b = 10

c = 0

c = a + b

print ("a: b: a+b: ".format(a,b,c))

c = a - b

print ("a: b: a-b: ".format(a,b,c) )

c = a \* b

print ("a: b: a\*b: ".format(a,b,c))

c = a / b

print ("a: b: a/b: ".format(a,b,c))

c = a % b

print ("a: b: a%b: ".format(a,b,c))

a = 2

b = 3

c = a\*\*b

print ("a: b: a\*\*b: ".format(a,b,c))

a = 10

b = 5

c = a//b

print ("a: b: a//b: ".format(a,b,c))

**Assignment operators** are used to assign values to variables:

|  |  |  |
| --- | --- | --- |
| **Operator** | **Example** | **Same As** |
| = | x = 5 | x = 5 |
| += | x += 3 | x = x + 3 |
| -= | x -= 3 | x = x - 3 |
| \*= | x \*= 3 | x = x \* 3 |
| /= | x /= 3 | x = x / 3 |
| %= | x %= 3 | x = x % 3 |
| //= | x //= 3 |  |
| \*\*= | x \*\*= 3 | x = x \*\* 3 |
| &= | x &= 3 | x = x & 3 |
| |= | x |= 3 | x = x | 3 |
| ^= | x ^= 3 | x = x ^ 3 |
| >>= | x >>= 3 | x = x >> 3 |
| <<= | x <<= 3 | x = x << 3 |

**Comparison operators** are used to compare two values:

|  |  |  |
| --- | --- | --- |
| **operator** | **Name** | **Example** |
| == | Equal | x == y |
| != | Not equal | x != y |
| > | Greater than | x > y |
| < | Less than | x < y |
| >= | Greater than or equal to | x >= y |
| <= | Less than or equal to | x <= y |

**Logical operators** are used to combine conditional statements:

|  |  |  |
| --- | --- | --- |
| **Operator** | **Name** | **Example** |
| and | AND | a and b |
| or | OR | a or b |
| not | NOT | not(a) |

**Identity operators** are used to compare the objects, not if they are equal, but if they are actually the same object, with the same memory location:

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| is | Returns True if both variables are the same object | x is y |
| is not | Returns True if both variables are not the same object | x is not y |

**Bitwise operators** are used to compare (binary) numbers:

|  |  |  |
| --- | --- | --- |
| **Operator** | **Name** | **Example** |
| & | AND | a & b |
| | | OR | a | b |
| ^ | XOR | a ^ b |
| ~ | NOT | ~a |
| << | Zero fill left shift | a << 3 |
| >> | Signed right shift | a >> 3 |

Three types of statements:-

Conditional :-if ,if else,…….

Unconditional:-break ,continue

Looping :-for,while…

**Conditions:-**

These conditions can be used in several ways, most commonly in "if statements" and loops.

Statement.

* If
* If else
* elif ladder
* nested if else

**if:** if is used to decide whether a certain statement or block or statements will be executed or not i.e if a certain condition is true then a block of statement is executed otherwise not

**syntax:**

if condition:

statement 1

statement 2

statement 3

**Ex1:**

a=20

b=10

if a>b:

print(“a is greater”)

output: a is greater

**if else:**

In the case of simple if it is possible to represent what is the action needs to be done if the condition is true. But there is no option to specify what action needs to be done. If the condition is false, in if-else text expression is true if block will get execute otherwise else part will get execute.

**Syntax:**

If test expression:

Body of if

else:

body of else

Program:

**Ex1:**

a=int(input())

b=int(input())

if a==b:

print(“both are equal”)

else:

print(“both are different”)

input: 2,5

output: 5 both are different.

**elseif ladder:**

To test the sequence of condition elif ladder is useful. The test expression is false in the case of it, then only it moves to the elif statement otherwise body of if block will get executed. If in case all the text expressions get failed then the by default else part will get execute.

**Syntax:**

If test expression:

Body of if

elif test expression:

body of elif

else:

body of else

Program:

**Ex1:**

a=int(input())

b=int(input())

if a==b:

print(“both are equal”)

elif a>b:

print(“a is greater”)

else:

print(“b is greater”)

input: 5,5

output: Both are equal

input: 4,2

output: a is greater

**Nested if:** writing if inside of other if statements. If the condition is true inside we will test other condition also

**Syntax:**

If condition:

If condition:

Statements

Else:

Statements

else:

if condition:

statements

else:

statements

Program: to storing of three numbers:

**Ex:**

a=int(input())

b=int(input())

c=int(input())

if a>b and a>c:

if b>c:

print(c,b,a)

else:

print(c,a,b)

elif b>c:

if a>c:

print(c,a,b)

else:

print(a,c,b)

else:

if a>b:

print(b,a,c)

else:

print(a,b,c)

input: 5,3,7

output: 3,5,7

**Control Statements:**

**break statement**: break statement is used to terminate the loop before the ending the sequence . it brings control out of the loop.

**Ex:**

for i in range(1,10):

if i==5:

break

print( i , end=’ ‘)

output: 1, 2, 3, 4

**continue statement:** if returns the control to the beginning of the loop.

For I in range(1,10):

If i%2 == 0 or i%3==0:

continue

print( i , end=’ ‘)

output: 5, 3, 7

**Set:**

* In python is set of unordered collection of Unique items no duplicates are allowed in set .
* The set has itself is inmutable. We can add or remove items from
* Set can be used to perform mathematical set operation like union, Intersection, symmetric difference etc.

EX:s1={4,7,’s’,”kalyani”,7.8}

EX2: s2=set([‘s’,’mango’,4,6,’apple’])----set constructor.

**Set function:**

* Add():add an element to set
* Clear():remove all elements from a set.
* Copy(): return show copy of set.
* Difference():return difference two or more sets as new set.
* Update():to insert elements into a set elements are immutable.
* Discard():It is remove a specific element from the set if the element is not available it does not effect on the set.
* Intersection():return intersection of two sets as a new set.
* Pop():it remove the element which placed as 1st element.
* Remove():it remove a specific element from the set if the element is not available it raises an exception key error.
* Symmetric\_difference():return the symmetric difference of two sets as a new set.
* Union():it will combine set one or more set and it will return the unique element.
* Update():update a set with the union of itself and others.
* Len():return the length of set.
* max():return largest element in the set.
* sorted():it return new sorted element in set.
* #set example  
  p={1,2,23,4,2}  
  k=set(p)  
  print(k)  
  # output{1,2,23,4}
* # Add()  
  my = {1, 2, 3}  
  my.add(4)  
  print(my)  
  #output: {1, 2, 3, 4}
* # Clear()  
  my1 = {1, 2, 3}  
  my1.clear()  
  print(my1)  
  # output: set()  
    
  # Copy()  
  a = {1, 2, 3}  
  new = a.copy()  
  print(new) # output: {1, 2, 3}  
    
  # difference  
  set1 = {1, 2, 3}  
  set2 = {3, 4, 5}  
  print(set1.difference(set2)) # output: {1, 2}  
    
  # Discard()  
  my1 = {1, 2, 3}  
  my1.discard(2)  
  my1.discard(4)  
  print(my1)  
  # Output: {1, 3}  
    
  # Intersection()  
  set1 = {1, 2, 3}  
  set2 = {2, 3, 4}  
  print(set1.intersection(set2))  
  # 0utput: {2, 3}  
    
  # pop()  
  k = {1, 2, 3}  
  element = k.pop()  
  print(element)  
  # Output: 1  
  print(k)  
  # Output: {2, 3}  
  # remove()  
  my\_set = {1, 2, 3}  
  my\_set.remove(2)  
  print(my\_set)  
  # Output: {1, 3}  
    
  # Symmetric difference()  
  set1 = {1, 2, 3}  
  set2 = {3, 4, 5}  
  print(set1.symmetric\_difference(set2))  
  # Output: {1, 2, 4, 5}  
  # union  
  set1 = {1, 2, 3}  
  set2 = {3, 4, 5}  
  print(set1.union(set2))  
  # Output: {1, 2, 3, 4, 5}  
    
  # Update()  
  set1 = {1, 2, 3}  
  set2 = {3,4, 5}  
  set1.update(set2)  
  print(set1)  
  # Output: {1, 2, 3, 4, 5}  
  # length  
  b = {1, 2, 3, 4}  
  print(len(b))  
  # Output: 4  
  # max()  
  krishna = {10, 20, 5, 7}  
  print(max(krishna))  
  # Output: 20  
    
  # sorted()  
  j = {3, 1, 4, 2}  
  print(sorted(j))  
  # Output: [1, 2, 3, 4]

**Dictionary:**

* Dictionary is data type data can store in the form of key value pair.
* Dictionary items are unordered, changeable, and do not allow duplicates.
* Key should be immutable.
* Value should be mutable it has duplicate values and different type of data.
* Key will act as index .No slicing because of it has no index.
* Key are unique.

EX: {}

Ex: a={‘a’:123,1:’abc’};

**Dictionary functions:**

Get():to retrieve the value of a particular key, we will use get(),if there is no key available get() will return none.

Update():we can insert new pair of key and value or update the existing key with a new value.

Pop(): In dictionary deal with key.

popItem(): deletes the last insert element in the dictionary.

Values():All the key are retrieved from the dictionary as addict.values object. We can type cast list also.

Keys():All the key are retrieved from the dictionary as addict.key object. We can type cast list also.

Items():All the keys and values are retrieved from the dictionary as items object we can typecast to list also.

# dictionary  
a= {'a':123 , 1:'abc'}  
print(a)  
  
  
# get()  
my = {'a': 1, 'b': 2, 'c': 3}  
print(my.get('b'))  
# Output: 2  
print(my.get('d'))  
# Output: None  
  
# update()  
k = {'a': 1, 'b': 2}  
k.update({'b': 10, 'c': 3})  
print(k)  
# Output: {'a': 1, 'b': 10, 'c': 3}

# pop()  
k= {'a': 1, 'b': 2, 'c': 3}  
value = k.pop('b')  
print(value)  
# Output: 2  
print(k)  
# Output: {'a': 1, 'c': 3}  
  
#popitem()  
k= {'a': 1, 'b': 2, 'c': 3}  
key\_value\_pair = h.popitem()  
print(key\_value\_pair)  
# Output: ('c', 3)  
print(k)  
# Output: {'a': 1, 'b': 2}  
  
#values()  
my\_dict = {'a': 1, 'b': 2, 'c': 3}  
values = my\_dict.values()  
print(list(values))  
# Output: [1, 2, 3]  
  
#keys()  
my\_dict = {'a': 1, 'b': 2, 'c': 3}  
keys = my\_dict.keys()  
print(list(keys))  
# Output: ['a', 'b', 'c']  
  
#items()  
my\_dict = {'a': 1, 'b': 2, 'c': 3}  
items = my\_dict.items()  
print(list(items))  
# Output: [('a', 1), ('b', 2), ('c', 3)]

**List:-**

List is one of the built-in data types in Python. A Python list is a sequence of comma separated items, enclosed in square brackets [ ]. The items in a Python list need not be of the same data type.

Ex:-

list1 = ["Rohan", "Physics", 21, 69.75]

list2 = [1, 2, 3, 4, 5]

Accessing Values in Lists:-

To acess the values in list use the square brackets for slicing along with the index or indices to obtain value available at that index.

Ex:-

list1 = ['physics', 'chemistry', 1997, 2000];

list2 = [1, 2, 3, 4, 5, 6, 7 ];

print ("list1[0]: ", list1[0])

print ("list2[1:5]: ", list2[1:5])

Updating Lists:-

You can update single or multiple elements of lists by giving the slice on the left-hand side of the assignment operator, and you can add to elements in a list with the append() method.

Ex:-

list = ['physics', 'chemistry', 1997, 2000];

print ("Value available at index 2 : ")

print (list[2])

list[2] = 2001;

print ("New value available at index 2 : ")

print (list[2])

Delete :-

To remove a list element you can use the del statement .

Ex:-

list1 = ['physics', 'chemistry', 1997, 2000];

print (list1)

del list1[2];

print ("After deleting value at index 2 : ")

print (list1)

|  |
| --- |
| List. append(obj):- Appends object obj to list.  List. Extend(seq):- Appends the contents of seq to list  List. clear():-Clears the contents of list.  List. insert(index, obj):-Inserts object obj into list at offset index  List. remove(obj):- Removes object obj from list  **Tuple:-**  Tuple is one of the built-in data types in Python. A Python tuple is a sequence of comma separated items, enclosed in parentheses (). The items in a Python tuple need not be of same data type.  Ex:-  tup1 = ("Rohan", "Physics", 21, 69.75)  tup2 = (1, 2, 3, 4, 5)  Access Tuple :-  The most common way to access values within a Python tuple is using indexing, We just need to specify the index of the elements we want to retrieve to the square bracket **[]** notation.  tuple1 = ("Rohan", "Physics", 21, 69.75)  tuple2 = (1, 2, 3, 4, 5)  print ("Item at 0th index in tuple1: ", tuple1[0])  print ("Item at index 2 in tuple2: ", tuple2[2])  Updating Tuples:-  In Python, tuple is an immutable sequence, meaning once a tuple is created, its elements cannot be changed, added, or removed.  Ex:-  T1 = (10, 20, 30, 40)  T2 = ('one', 'two', 'three', 'four')  #updating  T1 = T1 + T2  print(T1)  **Loops:-**  Loops allow us to execute a statement or group of statements multiple times.  **While Loop:-**  Repeats a statement or group of statements while a given condition is TRUE. It tests the condition before executing the loop body.  EX:-  count=0  while count<5:  count+=1  print ("Iteration no. {}".format(count))  print ("End of while loop")  **For Loop:-**  Executes a sequence of statements multiple times and abbreviates the code that manages the loop variable.  Ex:-  for num in range(5):  print (num, end=' ')  print()  for num in range(10, 20):  print (num, end=' ')  print()  for num in range(1, 10, 2):  print (num, end=' ')  Modules:-  You can define more than one related functions together and load required functions. A module is a file containing definition of functions, classes ,variables,  constants or any other Python object. Contents of this file can be made available to any other program.  Built-in Modules:-  Python's standard library comes bundled with a large number of modules. They are called built-in modules.  1.Math  2. Random  3. Datetime  Ex:-  import math  print ("Square root of 100:", math.sqrt(100))  output:10.0  User-defined Modules:-  Any text file with **.py** extension and containing Python code is basically a module. It can contain definitions of one or more functions, variables, constants as well as classes.  Ex:- def sum(x,y):  return x+y  def average(x,y):  return (x+y)/2  def power(x,y):  return x\*\*y  Packages:-  The module is a Python script with a **.py** extension and contains objects such as classes, functions, etc..   Packages in pythonextend the concept of the modular approach further. The package is a folder containing one or more module files; additionally, a special file **"\_\_init\_\_.py"** file may be empty but may contain the package list.  Create package:-  # mathfunctions.py  def sum(x,y):  val = x+y  return val  def average(x,y):  val = (x+y)/2  return val  def power(x,y):  val = x\*\*y  return val |
|  |  |