VALUES AND DATA TYPES

Values

A value is one of the fundamental things; a program works with like a word or number. Some example values are 5, 83.0, and 'Hello, World!'. These values belong to different **types**: 5 is an **integer**, 83.0 is a **floating-point number**, and 'Hello, World!' is a **string.** If you are not sure what type a value has, the interpreter can tell you:

```
>>>type(5)
<class 'int'>
>>>type(83.0)
<class 'float'>
>>>type('Hello, World!')
<class 'str'>
```

In these results, the word ―class‖ is used in the sense of a category; a type is a category of values. Integers belong to the type int, strings belong to str and floating-point numbers belong to float. The values like '5' and '83.0' look like numbers, but they are in quotation marks like strings.

```
>>>type('5')
<class 'str'>
>>>type('83.0')
<class 'str'>
```

Standard Datatypes

A datatype is a category for values and each value can be of different types. There are 7 data types mainly used in python interpreter.

a) Integer
b) Float
c) Boolean
d) String
e) List
f) Tuple
g) Dictionary

a) Integer

Let Integer be positive values, negative values and zero.

Example:

```
>>>2+2
    4
>>>a=-20
```

print() → 20

>>> type(a) → <type 'int'>

b) Float

A floating point value indicates number with decimal point.

Example:

>>> a=20.14

>>>type(a) → <type 'float'>

c) Boolean

A Boolean variable can take only two values which are **True or False.** True and False are simply set of integer values of 1 and 0.The type of this object is bool.

Example:

>>>bool(1)

True

>>>bool(0)

False

>>>a=True

>>>type(a) → <type 'bool'>

>>>b=false #Prints error

>>>c='True'

>>>type(c) → <type 'str'>

The boolean type is a subclass of the int class so that arithmetic using a boolean works.

>>>True + 1

2

>>>False * 85

0

 # A Boolean variable should use Capital T in true & F in False and shouldn't be enclosed within the quotes.

>>>d=10>45 → #Which returns False

Boolean Operators

Boolean Operations are performed by 'AND', 'OR', 'NOT'.

Example:

True and True → True

True and False → False

True or True → True

False or True → False

not False → True

d) String

A String is an ordered sequence of characters which can be created by enclosing characters in single quotes or double quotes.

Example:

```
>>>a=”Hello”
>>>type(a)
 <type ‘str’>
```

Subsets of strings can be taken using the slice operator ([ ] and [:]) with indexes starting at 0 in the beginning of the string and working their way from -1 at the end. The plus (+) sign is the string concatenation operator and the asterisk (*) is the repetition operator.

Example:

```
>>>str = 'Python Programming'
>>>print(str)                          # Prints complete string
>>>print(str[0])                       # Prints first character of the string
>>>print(str[-1])                      # Prints last character of the string
>>>print(str[2:5])                     # Prints characters starting from 3rd to 5th
>>>print(str[2:])                      # Prints string starting from 3rd character
>>>print(str * 2)                      # Prints string two times
>>>print(str + " Course")              # Prints concatenated string
```

Output

```
Python Programming
P
g
tho
thon Programming
Python ProgrammingPython Programming
Python Programming Course
```

String Functions:

For the following string functions the value of ***str1*** and ***str2*** are as follows:

```
>>>str1=”Hello”
>>>str2=”World”
```

S.No	Method	Syntax	Description	Example
1.	+	String1 + String2	It Concatenates two Strings	print(str1+str2)→ HelloWorld

2.	*	String*3	It multiples the string	str1*3 → HelloHelloHello
3.	len()	len(String)	Returns the length of the String	len(str1) →5
4.	centre()	centre(width,fullchar)	The String will be centred along with the width specified and the charecters will fill the space	str1.centre(20,+) → ++++Hello++++
5.	lower()	String.lower()	Converts all upper case into lower case	str1.lower() → hello
6.	upper()	String.upper()	Converts all lower case into upper case	str1.upper() → HELLO
7.	split()	String.split("Char")	splits according to the character which is present inside the function	str1.split("+") → H+E+L+L+O
8.	ord()	ord(String)	It converts a string in to its corresponding value	ord('a')→ 96
9.	chr()	chr(Number)	It converts a number in to its corresponding String	chr(100)-->'d'
10.	rstrip()	rstrip()	It removes all the spaces at the end	rstrip(a) → it returns -1
11.	\n	print("String\n")	New Line Character	print("Hello\n")
12.	\t	print("String\t")	It provides Space	print("Hello\t")
13.	\'	print("String\'String")	Escape Character (/) is used to print single quote or double quote in a String	print("Hello I\'m Fine")
14.	\"	print("String\"String")		print("Hello I\"m Fine")

e) List

A list is an ordered set of values, where each value is identified by an index. The values that make up a list are called its elements. A list contains items separated by commas and enclosed within square brackets ([]). Lists are mutable which means the items in the list can be add or removed later.

The values stored in a list can be accessed using the slice operator ([ ] and [:]) with indexes starting at 0 in the beginning of the list and working their way to end -1. The plus (+) sign is the list concatenation operator, and the asterisk (*) is the repetition operator.

Example:

```
>>>list = [ 'Hai', 123 , 1.75, 'vinu', 100.25 ]
>>>smalllist = [251, 'vinu']
>>>print(list)                    # Prints complete list
>>>print(list[0])                 # Prints first element of the list
>>>print(list[-1])                # Prints last 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(smalllist * 2)      # Prints list two times
>>>print(list + smalllist)   # Prints concatenated lists
```

Output

```
['Hai', 123, 1.75, 'vinu', 100.25]
Hai
100.25
[123, 1.75]
[1.75, 'vinu', 100.25]
[251, 'vinu', 251, 'vinu']
['Hai', 123, 1.75, 'vinu', 100.25, 251, 'vinu']
```

f) Tuple

Tuple are sequence of values much like the list. The values stored in the tuple can be of any type and they are indexed by integers. A tuple consists of a sequence of elements separated by commas. The main difference between list and tuples are:" List is enclosed in square bracket ([ ]) and their elements and size can be changed while tuples are enclosed in parenthesis (( )) and cannot be updated.

Syntax:

```
tuple_name=(items)
```

Example:

```
>>> tuple1=('1','2','3','5')
>>>tuple2=('a','b','c')
>>>tuple3='3','apple','100'
>>>print(tuple2)                #print tuple2 elements
>>>print(tuple2[0])             #print the first element of tuple2
>>>print(tuple2 + tuple3)       #print the concatenation of tuple2 and tuple3
>>>print(tuple3[2])             #print the second element of tuple3
```

Output:

```
('a','b','c')
('a')
('1','2','3','5','a','b','c')
('3')
```

g) Dictionary

Dictionaries are an unordered collection of items. Dictionaries are enclosed by curly braces '{ }' .The element in dictionary is a comma separated list of keys: value pairs where keys are usually numbers and strings and values can be any arbitrary python data types. The value of a dictionary can be accessed by a key. and values can be accessed using square braces '[ ]'

Syntax:

```
dict_name= {key: value}
```

Example:

```
>>>dict1={}
>>>dict2={1:10,2:20,3:30}
>>>dict3={'A':'apple','B':'200'}
>>>Dict={'Name':'john','SSN':4576,'Designation':'Manager'}
```