Lecture-4

Python Data Types:

Data types are the classification or categorization of data items. Data types represent a kind of value which determines what operations can be performed on that data. Numeric, non-numeric and Boolean (true/false) data are the most used data types. However, each programming language has its own classification largely reflecting its programming philosophy.

Add your content here

Python has the following standard or built-in data types:

Numeric

A numeric value is any representation of data which has a numeric value. Python identifies three types of numbers:

Integer: Positive or negative whole numbers (without a fractional part)

Float: Any real number with a floating point representation in which a fractional component is denoted by a decimal symbol or scientific notation

Complex number: A number with a real and imaginary component represented as x+yj. x and y are floats and j is -1(square root of -1 called an imaginary number)

Boolean:

Data with one of two built-in values True or False. Notice that 'T' and 'F' are capital. true and false are not valid booleans and Python will throw an error for them.

Boolean logic:

Booleans are used in your code to make it behave differently based on current conditions within your program. You can use boolean values and comparisons in conjunction with the if, elif, and else keyoards as one means to achieve this.

```
my_age = 10
 if my_age >= 100:
   print("One hundred years old! Very impressive.")
elif my_age <= 3:
   print("Awwww. Just a baby.")
else:
    print("Ah - a very fine age indeed")
```

OUTPUT:

```
Ah - a very fine age indeed…..
```

Boolean and operator returns true if both operands return true.

```
>>> a=50
>>> b=25
>>> a>40 and b>40
False
>>> a>100 and b<50
False
>>> a==0 and b==0
False
>>> a>0 and b>0
True
```

Boolean or operator returns true if any one operand is true

```
>>> a=50
>>> b=25
>>> a>40 or b>40
True>>>
 a>100 or b<50
True
>>> a==0 or b==0
False
>>> a>0 or b>0
True
```

The not operator returns true if its operand is a false expression and returns false if it is true.

```
>>> a=10
>>> a>10
False
>>> not(a>10)
True
```

Python None Keyword:

The None keyword is used to define a null value, or no value at all.

None is not the same as 0, False, or an empty string. None is a datatype of its own (NoneType) and only None can be None.

Example:

If you do a boolean if test, what will happen? Is None True or False:

```
x = None
if x:
  print("Do you think None is True")
else:
  print("None is not True...")
```

OUTPUT

```
None is not True...
```

- **Sequence Type:**

  A sequence is an ordered collection of similar or different data types. Python has the following built-in sequence data types:

- **String**: A string value is a collection of one or more characters put in single, double or triple quotes.
- **List** : A list object is an ordered collection of one or more data items, not necessarily of the same type, put in square brackets.
- **Tuple**: A Tuple object is an ordered collection of one or more data items, not necessarily of the same type, put in parentheses.

Dictionary:

A dictionary object is an unordered collection of data in a key:value pair form. A collection of such pairs is enclosed in curly brackets. For example: {1:"Steve", 2:"Bill", 3:"Ram", 4: "Farha"}

type() function:

Python has an in-built function **type()** to ascertain the data type of a certain value. For example, enter type(1234) in Python shell and it will return <class 'int'>, which means 1234 is an integer value. Try and verify the data type of different values in Python shell, as shown below.

```
>>> type(1234)
<class 'int'>
>>> type(55.50)
<class 'float'>
>>> type(6+4j)
<class 'complex'>
>>> type("hello")
<class 'str'>
>>> type([1,2,3,4])
<class 'list'>
>>> type((1,2,3,4))
<class 'tuple'>
>>> type({1:"one", 2:"two", 3:"three"})
<class 'dict'>
```

Python - Number Types:

Python includes three numeric types to represent numbers: integer, float, and complex.

Integer:

Zero, positive and negative whole numbers without a fractional part and having unlimited precision, e.g. 1234, 0, -456.

A number having **0o** or **0O** as prefix represents an **octal** number.

For example: 0O12: equivalent to 10 (ten) in the decimal number system.

A number with **0x** or **0X** as prefix represents **hexadecimal** number.

For example: 0x12: equivalent to 18 (Eighteen) in the decimal number system.

Float:

Positive and negative real numbers with a fractional part denoted by the decimal symbol or the scientific notation using E or e, e.g. 1234.56, 3.142, -1.55, 0.23.

Scientific notation is used as a short representation to express floats having many digits.

For example:

345600000000 is represented as 3.456e11 or 3.456E11

345.56789 is represented as 3.4556789e2 or 3.4556789E2

Data Type Conversion in Python:

Sometimes, you may need to perform conversions between the built-in types. To convert between types, you simply use the type name as a function.

There are several built-in functions to perform conversion from one data type to another. These functions return a new object representing the converted value.

Add your content here

Sr.No.	Function & Description
1	**int(x [,base])** Converts x to an integer. base specifies the base if x is a string.
2	**long(x [,base])** Converts x to a long integer. base specifies the base if x is a string.
3	**float(x)** Converts x to a floating-point number.
4	**complex(real [,imag])** Creates a complex number.
5	**str(x)** Converts object x to a string representation.
6	**repr(x)** Converts object x to an expression string.
7	**eval(str)** Evaluates a string and returns an object.

Sr.No.	Function & Description
8	**tuple(s)** Converts s to a tuple.
9	**list(s)** Converts s to a list.
10	**set(s)** Converts s to a set.
11	**dict(d)** Creates a dictionary. d must be a sequence of (key,value) tuples.
12	**frozenset(s)** Converts s to a frozen set.
13	**chr(x)** Converts an integer to a character.
14	**unichr(x)** Converts an integer to a Unicode character.

Sr.No.	Function & Description
15	**ord(x)** Converts a single character to its integer value.
16	**hex(x)** Converts an integer to a hexadecimal string.
17	**oct(x)** Converts an integer to an octal string.

Numeric data conversion functions –

int() – converts a floating point number or a string with integer representation to integer object. When converting a string, parameter of base of number system to convert hexadecimal or octal number to integer

```
>>> int('11')
11
>>> int(11.15)
11
>>> int('20', 8)
16
>>> int('20', 16)
32
```

float() – attaches fractional part with 0 to integer, or converts a string with float representation to a floating point number object.

```
>>> float(11)

11.0

>>> float('11.11')

11.11
```

str() – converts object of any data type to string representation

```
>>> str(10) # int to str
'10‘
>>> str(11.11) # float to str
'11.11‘
>>> str([1,2,3]) #list to str
'[1, 2, 3]‘
>>> str((1,2,3)) # tuple to str
'(1, 2, 3)‘
>>> str({1:100,2:200})
'{1: 100, 2: 200}'
```

complex() – accepts two floats as parameters and returns a complex number object. First parameter is the real component and second parameter multiplied by j is the imaginary component.

```
>>> complex(2.5, 3.5)
(2.5+3.5j)
```

list() – converts a string and tuple to list object. It also returns a list object from keys of a dictionary

```
>>> list("TutorialsPoint")
['T', 'u', 't', 'o', 'r', 'i', 'a', 'l', 's', 'P', 'o', 'i', 'n', 't']
>>> list((1,2,3))
[1, 2, 3]
>>> list({'a':11,'b':22,'c':33})
['a', 'b', 'c']
```

tuple() – converts a string and list to tuple object. It also returns a tuple object from dictionary keys

```
>>> tuple('TutorialsPoint')
('T', 'u', 't', 'o', 'r', 'i', 'a', 'l', 's', 'P', 'o', 'i', 'n', 't')
>>> tuple([1,2,3])
(1, 2, 3)
>>> tuple({'a':11,'b':22,'c':33})
('a', 'b', 'c')
```

dict() – returns a dictionary object from list of two tuples with equal number of elements.

```
>>> dict([(1,1),(2,2)])
{1: 1, 2: 2}
```

References:

1.Introduction to Computation and Programming using Python, by John Guttag, PHI Publisher

2.Python Programming using problem solving Approach by ReemaThareja, Oxford University, Higher Education Oxford University Press; First edition (10 June 2017), ISBN-10: 0199480173

3. Fundamentals of Python first Programmes by Kenneth A Lambert, Copyrighted material Course Technology Inc. 1 st edition (6th February 2009)

4. https://www.tutorialspoint.com/python/index.htm

5. https://www.geeksforgeeks.org/python-programming-language

6. https://www.w3schools.com/python/

****END OF THE LECTURE***

THANK YOU