

# Django for Web Development & Artificial Intelligence

## Lecture - 02

### **Instructor:**

Md. Abu Noman Basar  
Django Developer

## ☐ Download & Install python.

- visit the official website of Python: <http://www.python.org/>

## ☐ Download & Install PyCharm.

- visit the official website of PyCharm: <https://www.jetbrains.com/pycharm/>

- ❑ We use the **print()** function to output data to the standard output device (screen).

Ex: `print('StudyMart')`

- ❑ This **input()** function is used to read a line of input entered by the user at the console and returns it as a string.

Ex: `name = input('Enter Your Organization Name: ')`

Variables are containers for storing data values.

## Rules of Variable:

- ☐ The first character of the variable must be an alphabet or underscore ( \_ ).

Ex: Studymart / \_studymart

- ☐ A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and \_ )
- ☐ Identifier name must not contain any white-space, or special character (!, @, #, %, ^, &, \*).
- ☐ Identifier name must not be similar to any keyword defined in the language.
- ☐ Identifier names are case sensitive

Ex: Aquest and aquest is not the same.

Variable names with more than one word can be difficult to read.

## ❑ Camel Case:

Each word, except the first, starts with a capital letter.

Ex: camelCaseVariable = “Aiquest”

## ❑ Pascal Case:

Each word starts with a capital letter.

Ex: PascalCaseVariable = “studymart”

## ❑ Snake Case:

Each word is separated by an underscore character.

Ex: snake\_case\_variable = “Django”

Python allows us to assign a value to multiple variables in a single statement, which is also known as multiple assignments.

## ❑ Assigning a single value to multiple variables.

```
a = b = c = 34  
print(a)  
print(b)  
print(c)
```

## ❑ Assigning multiple values to multiple variables.

```
a, b , c = 34,30,32  
print(a)  
print(b)  
print(c)
```

- ❑ Strings in python are surrounded by either single quotation marks or double quotation marks.

Ex: “Noman”

‘Basar’

- ❑ In Python, we use the input() function to take input from the user. **Whatever you enter as input, the input function converts it into a string.** If you enter an integer value still input() function converts it into a string.

## ❏ Slicing:

You can return a range of characters by using the slice syntax.

**Ex:** `s = "Welcome to python with Django"`

- **Slice From the Start:**

```
print(s[0:7])
```

-**Slice To the End**

```
print(s[5:])
```

- **Range**

```
print(s[5:12])
```

***\*\*\*The first character has an index of 0.***



## ❑ Upper Case:

The **upper()** method returns the string in upper case.

**Ex:** `s = "Welcome to python with Django"`  
`print(s.upper())`

## ❑ Lower Case:

The **lower()** method returns the string in lower case.

**Ex:** `print(s.lower())`

## ❑ Remove Whitespace:

The **strip()** method removes any whitespace from the beginning or the end.

**Ex:** `s = " Welcome to python with Django "`  
`print(s.strip())`

## ❑ Replace String:

The `replace()` method replaces a string with another string.

**Ex:** `s = "Welcome to python with Django"`  
`print(s.replace("o", "a"))`

## ❑ Split String:

The `split()` method returns a list where the text between the specified separator becomes the list items.

**Ex:** `print(s.split())`

## ❑ String Concatenation:

To concatenate, or combine, two strings you can use the `+` operator.

**Ex:** `sm = "AI"`  
`print(s+sm)`

## ❑ Count() method:

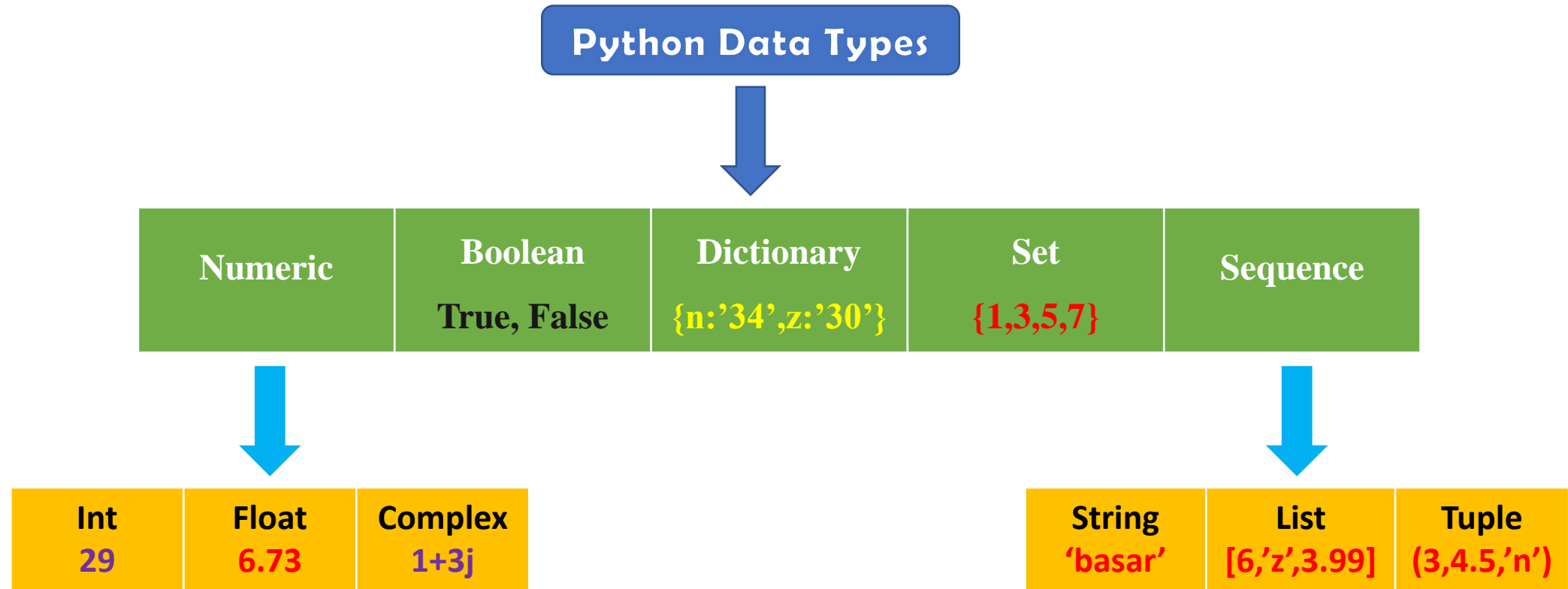
The `count()` method returns the number of times a specified value appears in the string.

**Ex:** `s = "Welcome to python with Django"`  
`print(s.count("o"))`

## ❑ len():

The `len()` returns the number of characters in the string.

**Ex:** `print(len(s))`



- **Integer:** Int, or integer, is a whole number, positive or negative, without decimals, of unlimited length.

**Ex:**  $n = 10$

$p = -12367632$

$y = 7823487325873273824$

- **Float:** Float is a number, positive or negative, containing one or more decimals.

**Ex:**  $z = 9.99$

- **Complex:** Complex numbers are written with a "j" as the imaginary part:

**Ex:**  $u = 8+10j$

# Data Type Conversion

## Numeric

```
z = 30      #int
u = 34.33   #float
m = 7j      #complex
```

### ❑ Int to float:

Ex: `l = float(z)`

### ❑ Int to complex:

Ex: `o = complex (z)`

### ❑ float to int:

Ex: `q = int(u)`

### ❑ float to complex:

Ex: `r = complex(u)`

**\*\*\*You cannot convert complex numbers into another number type.**

- **Boolean:** Boolean type provides two built-in values, True and False. These values are used to determine the given statement true or false. It denotes by the class bool.

**Ex 1:** `print(type(True))`  
`print(type(False))`

**Ex 2:** `x = 90>10`  
`print(x)`  
`print(type(x))`

**Ex 3:** `x = 90>100`  
`print(x)`  
`print(type(x))`

**Dictionary:** Dictionaries are used to store data values in key: value pairs.

- A dictionary is a collection that is **ordered, changeable, and does not allow duplicates**.
- Dictionaries are written with curly brackets and have keys and values.

**Ex:**

```
firstdict = {  
    'name': 'Noman',  
    'id': 107536,  
    'year': 2022  
}  
  
print(firstdict)  
print('firstdict type : ', type(firstdict))
```



**Set:** Sets are used to store multiple items in a single variable. Python Set is the unordered collection of the data type. It is iterable, mutable(can modify after creation), and has unique elements.

- Sets are written with curly brackets.

**Ex:**

```
firstset= {'noman',34,True,'basar'}
```

```
print(firstset)
```

```
print('firstdict type : ', type(firstset))
```

**List:** Lists are used to store multiple items in a single variable. The list can contain data of different types.

- The items stored in the list are separated with a comma (,) and enclosed within square brackets [].

**Ex:**

```
firstlist= ['noman',34,True, 'basar']
```

```
print(firstlist)
```

```
print('firstdict type : ', type(firstlist))
```

**Tuple:** Tuples are used to store multiple items in a single variable.

- A tuple is a collection that is ordered and **unchangeable**.
- Tuples are written with round brackets.

**Ex:**

```
firsttuple= ('noman',3.4,True, 'basar')
```

```
print(firsttuple)
```

```
print('firstdict type : ', type(firsttuple))
```