# Built-in Data Types in Python

Vivek K. S., Deepak G.

Information Systems Decision Sciences (ISDS)
MUMA College of Business
University of South Florida
Tampa, Florida

2017

## Objectives

- To understand built-in data types such as integers, floats, strings and Boolean type.
- To learn their usage and specific behavior.
- To understand objects in Python.
- To understand dynamic typing in Python.
- To understand variables and references.

## Objects in Python

- In Python, everything is an object.
- Different programming languages have different definitions to objects.
- In some languages, it is a very strict definition that dictates that all objects must have methods and attributes associated with them.
- Such objects are subclassable.
- In Python, the definition is a little loose, which says that objects may or may not have attributes and methods.
- Everything that can be assigned to a variable and passed as an argument to a function qualifies to be an object in Python.

## Objects in Python

From Guido's Blog –
One of my goals for Python was to make it so that all objects were "first class." By this, I meant that I wanted all objects that could be named in the language (e.g., integers, strings, functions, classes, modules, methods, etc.) to have equal status. That is, they can be assigned to variables, placed in lists, stored in dictionaries, passed as arguments, and so forth.

## Objects in Python

- Integers, Strings, Lists, Functions, Modules are all implemented as objects.
- An object is in its simple definition a block(s) of memory that contains data in it.
- The data has a type associated with it and consequently a set of behaviors (methods) and what could be done with them to change its attributes.
- The type of data also determines its mutability and immutability.
- The advantage of this implementation is consistency.

## Dynamic Typing in Python

- Python is a dynamically typed language.
- Unlike statically-typed languages like Java or C++, Python does not attach the type of an object to its variable identifier.
- Instead, the assignment of an object to an identifier simply attaches a name to the block of memory containing the data and acts only as a reference to it.
- In python, we use the type() method to identify the type of the variable.

```
# Code to identify data type
a = 10
print(a)
type(a)
```

## Rules for Identifier Names

The following are some of the rules associated with identifier names in Python. The following characters are allowed in identifier names.

- Lowercase letters (a through z)
- Uppercase letters (A through Z)
- Digits (0 through 9)
- `Underscore (_)`

Their usage is as follows:

- Names cannot begin with a digit.
- Python treats names that begin with an underscore in special ways. For example __init__
- Reserved keywords in Python cannot be used for identifier names.
  `https://docs.python.org/2.5/ref/keywords.html`

## Built-in Types in Python

The following types are the basic building block of programming in Python. Dealing with numbers and textual data is the cornerstone of any programming language and it is no different in Python. Python has in-built support for,

- Numbers
  - Integers (whole numbers such as 5 and 1,000,000,000).
  - Floating point numbers (such as 3.1416, 14.99, and 1.87e4).
- Strings (The first Python Sequence Type. Python 3 supports Unicode standard).

## Unique Operations in Python

Python offers some unique flavors in the most common operations we use on a daily basis:
Python offer two types of division operation.

```
# '/' (slash) carries out decimal division.
7/2 = 3.5.
# '//' (double slash) carries out integer
# division also called as floor division.
7//2 = 3

x = 77
x //= 10
print(x)
7
```

## Arithmetic Operation and Assignment in Python

An arithmetic operation and assignment of the result could be done as easy as follows between two operands.

```
Adding two integers (numbers).
a = 95
a -= 3
print(a)
92
a *= 2
print(a)
184
```

## Common Operations in Python

The Modulo operator returns the remainder of a division operation.

```
Modulo operation 9%5 = 4

# Getting the reminder & quotient can also
be done by using divmod()

divmod(9,5) = (1,4)
```

## Type Conversions in Python

Python also offers ways to convert one type into another:

```
# Converting a Boolean to integer
>>>int(True)
1
>>>int(False)
0

# Converting a Float to integer

>>>int(55.5)
55
>>>int(1.0e4)
10000
```

## Continued..

```
# Converting a String to integer
>>>int('99')
99
>>>int('-55')
-55

# Converting an int to char
>>>chr(97)
'a'
Getting the ASCII code of a character.
>>>ord('a')
97

ord() and chr() are built-in functions in Python.
1
```

[1] Refer https://docs.python.org/3/library/functions.html for more details on built-in functions.

## Integer Overflow

- Python handles really long numbers with ease.
- This is a feature which most Programming languages have a problem with, commonly referred to a s "Integer overflow".
- For example 10**100 (10 raised to the power 100) will result in a huge number called the googol.
- Even if an multiplication is to be performed between two googols, Python can easily handle that math and support the operation.
- Lets try it.

```
googol = 10**100
print(googol*googol)
```

## Floats in Python

Floats are numbers with decimal points.

- All the operations that work on integers can be applied to floats as well.
- Float type conversion can be done using the method float() with the data passed as an argument to it.
- Strings can be converted to floats as well.

## Strings in Python

A String is a sequence type in Python. It is simply a sequence of characters.

- In Python, Strings are immutable.
- The data in a String cannot be modified, but copies can be created.
- Strings in Python 3 support unicode operations.
- In Python, Strings can be written using both single and double quotes, which allows double quotes to be written within single quotes and vice versa.

```
>>>'Python'
Python
>>>"Python"
Python
```

## Triple Quotes and Docstrings

Triple Quotes is a unique flavor offered by Python.

- They are most commonly used to create multi-line Strings that are sometimes used as comments and docstrings.
- One of their most common uses is in creating docstrings.
- A docstring is a string literal specified in a function or any piece of code as a comment, to document that specific segment of code. For example,

```
def add(x,y):
        '''The function adds two integers
        and returns the sum.'''
```

## Continued..

There are two ways to get the docstring.

```
# Method #1 -Using help()

help(add)

'Help on function add in module __main__:

add(x, y)
The function adds two integers and returns
the sum.'

# Method #2 - Using __doc__
add.__doc__

'The function adds two integers and
returns the sum.'
```

## Common String Operations

The following are some of the common String operations.

- String Concatenation.
- Duplication.
- Replacing a substring.
- Indexing and Slicing.
- Finding the length.
- Splitting and Stripping.
- String formatting operations.

## More Examples

More code examples are available at –
`https://github.com/vivek14632/Python-Workshop/tree/master/Introducing%20Python/Chapter%202`

## Summary

- We understood Built-in data types in Python.
- We learned how Python works as a Dynamically typed language.
- We understood the use of integers, strings and floats and the operations that could be performed on them.
- We understood type conversion between different types.