# INTRODUCTION

Python is easy to learn yet powerful and versatile scripting language, which makes it attractive for Application Development.

Python's syntax and dynamic typing with its interpreted nature make it an ideal language for scripting and rapid application development.

Python supports multiple programming patterns including object-oriented, imperative and functional or procedural programming styles.

We don't need to use data types to declare variable because it is dynamically typed so we can write a=10 to assign an integer value in an integer variable. The interpreter implicitly binds the value with its type.

Python makes the development and debugging fast because there is no compilation step included in Python development, and edit-test-debug cycle is very fast. Python is a high-level programming language and provides lots of high-level data structures.

High level languages are easier to understand and are more like human language and less like machine language.

A high-level language does not require addressing hardware constraints when developing a program. **However, every single program written in a high-level language must be interpreted into machine language before being executed by the computer.** Python, C/C++ and Java are popular examples of high-level languages.

## PYTHON FEATURES

Python provides lots of features that are listed below.

1. **Easy to Learn and Use:** Python is easy to learn and use. It is developer-friendly and high-level programming language.
2. **Expressive Language:** Python language is more expressive means that it is more understandable and readable.
3. **Interpreted Language:** Python is an interpreted language i.e. interpreter executes the code line by line at a time. This makes debugging easy and thus suitable for beginners.
4. **Cross-platform Language:** Python can run equally on different platforms such as Windows, Linux, Unix and Macintosh etc. So, we can say that Python is a portable language.
5. **Free and Open Source:** Python language is freely available at official web address. The source-code is also available. Therefore, it is open source.
6. **Object-Oriented Language:** Python supports object-oriented language and concepts of classes and objects come into existence.
7. **Extensible:** It implies that other languages such as C/C++ can be used to compile the code and thus it can be used further in our python code.
8. **Large Standard Library:** Python has a large and broad library and provides rich set of module and functions for rapid application development.
9. **GUI Programming Support:** Graphical user interfaces can be developed using Python.
10. **Integrated:** It can be easily integrated with languages like C, C++, JAVA etc.

## PYTHON 2 VS. PYTHON 3

In most of the programming languages, whenever a new version releases, it supports the features and syntax of the existing version of the language, therefore, it is easier for the projects to switch in the newer version. However, in the case of Python, the two versions Python 2 and Python 3 are very much different from each other.

A list of differences between Python 2 and Python 3 are given below:

1. Python 2 uses **print** as a statement and used as print "something" to print some string on the console. On the other hand, Python 3 uses **print** as a function and used as print("something") to print something on the console.
2. Python 2 uses the function raw\_input () to accept the user's input. It returns the string representing the value, which is typed by the user. To convert it into the integer, we need to use the int () function in Python. On the other hand, Python 3 uses input () function which automatically interpreted the type of input entered by the user. However, we can cast this value to any type by using primitive functions (int (), str (), etc.).
3. In Python 2, the implicit string type is ASCII, whereas, in Python 3, the implicit string type is Unicode.
4. Python 3 doesn't contain the xrange () function of Python 2. The xrange () is the variant of range () function which returns a xrange object that works similar to Java iterator. The range () returns a list for example the function range (0,3) contains 0, 1, 2.
5. There is also a small change made in Exception handling in Python 3. It defines a keyword **as** which is necessary to be used. We will discuss it in Exception handling section of Python programming tutorial.

# TOKENS

Tokens can be defined as a punctuator mark, reserved words and each individual word in a statement. Token is the smallest unit inside the given program.

There are following tokens in Python:

* Keywords
* Literals
* Operators
* Identifiers

## KEYWORDS

Python Keywords are special reserved words which convey a special meaning to the compiler/interpreter. Each keyword has a special meaning and a specific operation. These keywords can't be used as variable. Following is the List of Python Keywords.

In Python there are a total of 33 keywords (we can access the list using keywords.kwlist)



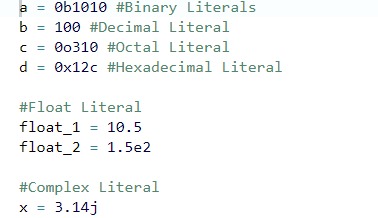
## IDENTIFIERS

A Python identifier is a name used to identify a variable, function, class, module or other object. An identifier starts with a letter A to Z or a to z or an underscore (\_) followed by zero or more letters, underscores and digits (0 to 9).

Python does not allow punctuation characters such as @, $, and % within identifiers. Python is a case sensitive programming language. Thus, Manpower and manpower are two different identifiers in Python.

## LITERALS

Literals can be defined as the data that is given in a variable or constant.



We assigned integer literals into different variables. Here, a is binary literal, b is a decimal literal, c is an octal literal and d is a hexadecimal literal. Similarly, we have string literals, Boolean literals and Collections literals.



## OPERATORS

The operator can be defined as a symbol which is responsible for a particular operation between two operands. Operators are the pillars of a program on which the logic is built in a particular programming language. Python provides a variety of operators described as follows.

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

# PYTHON VARIABLES

Variable is a name which is used to refer memory location. Variable also known as identifier and used to hold value.

In Python, we don't need to specify the type of variable because Python is a type infer language and smart enough to get variable type.

Variable names can be a group of both letters and digits, but they have to begin with a letter or an underscore. It is recommended to use lowercase letters for variable name. Rahul and rahul both are two different variables.

## IDENTIFIER NAMING

Variables are the example of identifiers. An Identifier is used to identify the literals used in the program. The rules to name an identifier are given below.

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

All the characters except the first character may be an alphabet of lower-case(a-z), upper-case (A-Z), underscore or digit (0-9).

* 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 for example my name, and MyName is not the same.
* Examples of valid identifiers: a123, \_n, n\_9, etc.
* Examples of invalid identifiers: 1a, n%4, n 9, etc.

## VARIABLE ASSIGNMENT

We don't need to declare explicitly variable in Python. When we assign any value to the variable that variable is declared automatically. **The equal (=) operator is used to assign value to a variable.**

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

We can apply multiple assignments in two ways either by assigning a single value to multiple variables or assigning multiple values to multiple variables. The values will be assigned in the order in which variables appears.



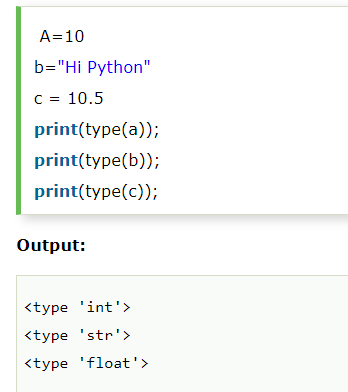


# DATA TYPES

In Python everything is an object hence we can say that all data types are classes and corresponding variables are the objects of those classes inheriting its features / properties.

Variables can hold values of different data types. The interpreter implicitly binds the value with its type.

Python enables us to check the type of the variable used in the program. Python provides us the type () function which returns the type of the variable passed.



Python provides various standard data types that define the storage method on each of them. The data types defined in Python are given below.

* Numbers
* String
* List
* Tuple
* Dictionary

## NUMBERS

Number stores numeric values. Python creates Number objects when a number is assigned to a variable.

Python supports 4 types of numeric data.

* int (signed integers like 10, 2, 29, etc.)
* long (long integers used for a higher range of values like 908090800L, -0x1929292L, etc.)
* float (float is used to store floating point numbers like 1.9, 9.902, 15.2, etc.)
* complex (complex numbers like 2.14j, 2.0 + 2.3j, etc.)

## STRING

The string can be defined as the sequence of characters represented in the quotation marks. In python, we can use single, double, or triple quotes to define a string.

String handling in python is a straightforward task since there are various inbuilt functions and operators provided.

In the case of string handling, the operator + is used to concatenate two strings as the operation "hello"+" python" returns "hello python".



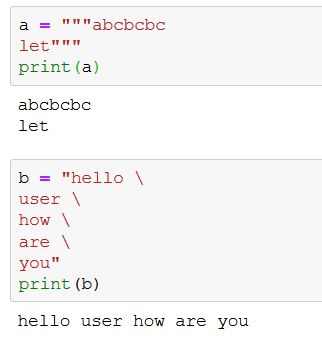
The operator \* is known as repetition operator as the operation.



**Multi line String-** A piece of text that is spread along multiple lines is known as Multiple line String.

There are two ways to create Multiline Strings:

* Adding black slash at the end of each line.
* Using triple quotation marks.

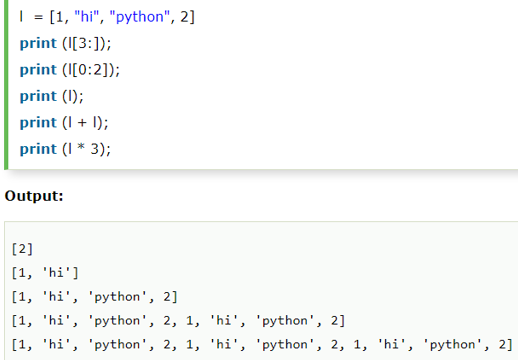


## LIST

Lists are similar to arrays in C. However; the list can contain data of different types. The items stored in the list are separated with a comma (,) and enclosed within square brackets [].

Lists are mutable data structures i.e. the items in the list can be changed after its created.

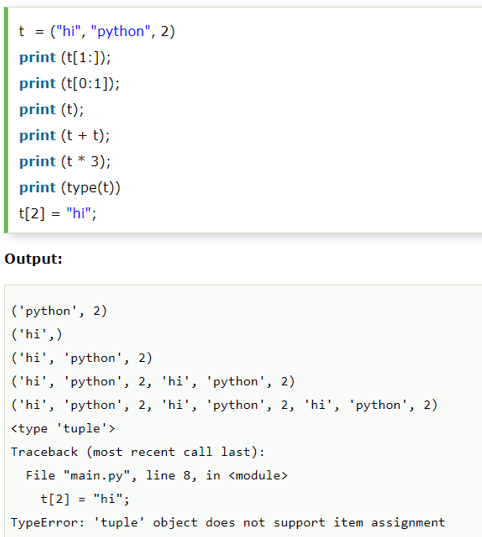
We can use slice [:] operators to access the data of the list. The concatenation operator (+) and repetition operator (\*) works with the list in the same way as they were working with the strings.



## TUPLE

Tuple is another form of collection where different type of data can be stored. A tuple is similar to the list in many ways. Like lists, tuples also contain the collection of the items of different data types. The items of the tuple are separated with a comma (,) and enclosed in parentheses ().

A tuple is a read-only (immutable) data structure as we can't modify the size and value of the items of a tuple once its created.

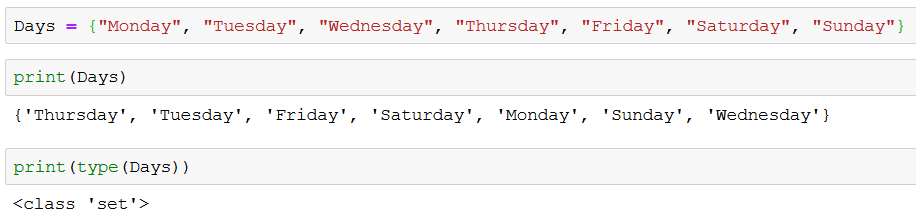


## SET

Set is an unordered collection of unique items. Set is defined by values separated by comma inside braces {}. Items in a set are not ordered.

We can perform set operations like union, intersection on two sets. Set have unique values.

Because sets cannot have multiple occurrences of the same element, it makes sets highly useful to efficiently remove duplicate values from a list or tuple and to perform common math operations like unions and intersections.



Unlike other collections in python, there is no index attached to the elements of the set, i.e., we cannot directly access any element of the set by the index. However, we can print them all together or we can get the list of elements by looping through the set.

## FROZENSETS

The frozen sets are the immutable form of the normal sets, i.e., the items of the frozen set cannot be changed and therefore it can be used as a key in dictionary.

The elements of the frozen set cannot be changed after the creation. We cannot append or delete the content of the frozen sets by using the methods like add () or remove ().

The frozenset () method is used to create the frozenset object. The iterable sequence is passed into this method which is converted into the frozen set as a return type of the method.

## DICTIONARY

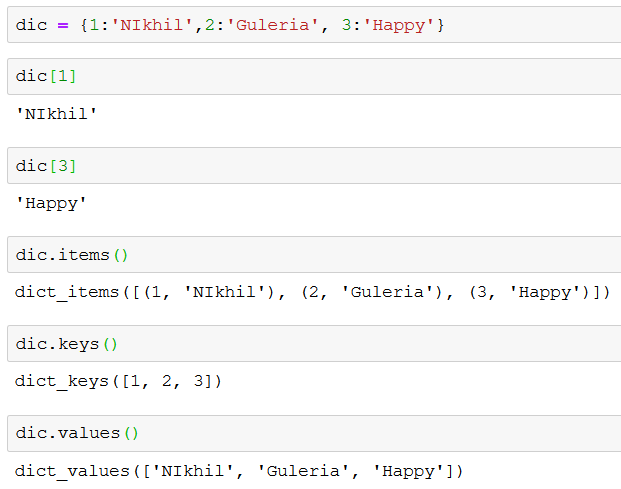
Dictionary is an ordered set of a key-value pair of items. It is like an associative array or a hash table where each key stores a specific value. Key can hold any primitive data type whereas value is an arbitrary Python object.

The items in the dictionary are separated with the comma and enclosed in the curly braces {}.

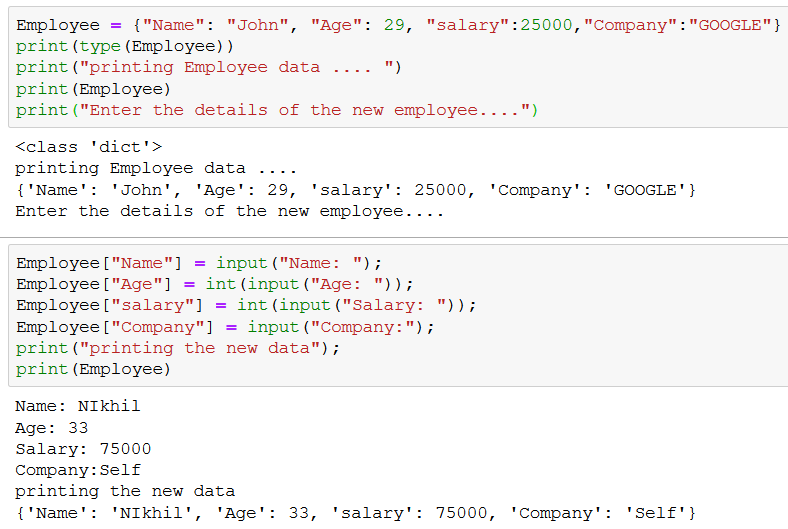
It is generally used when we have a huge amount of data. Dictionaries are optimized for retrieving data. We must know the key to retrieve the value.

Dictionary:

* Dictionary is a collection which works on a key-value pair.
* To access elements in the dictionary we make use of key instead of index.
* It works like an associated array where no two keys can be same.
* Dictionaries are enclosed by curly braces {} and values can be retrieved by square bracket [].

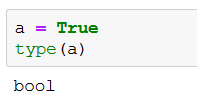


We can update items of the dictionary as shown below:



## BOOLEAN

Boolean values can be represented as one of the two constants “False” and “True”. Booleans behave like the integers 0 and 1 respectively.



# SEQUENCE VS COLLECTIONS

A sequence is a group of items with a deterministic ordering. The order in which we put them in is the order in which we get an item out from them.

Python collection, unlike a sequence, does not have a deterministic ordering. Examples include sets and dictionaries. In a collection, while ordering is arbitrary, physically, they do have an order.

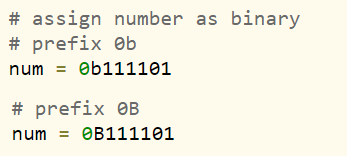
Every time we visit a set, we get it’s items in the same order. However, if we add or remove an item, it may affect the order.

# MISCELLANEOUS

Below are some important points to be known to work easily with python.

## ASSIGNING BINARY TO VARIABLES

To assign binary values to the variable, we use prefix 0b or 0B with the binary value.



## DEEP COPY VS SHALLOW COPY

The difference between shallow and deep copying is only relevant for compound objects (objects that contain other objects, like lists or class instances):

* A shallow copy constructs a new compound object and then (to the extent possible) inserts references into it to the objects found in the original.
* A deep copy constructs a new compound object and then, recursively, inserts copies into it of the objects found in the original.

