Chapter 1: Review of Python basic (Part 1)

Introduction of Python

Python is an interpreter, interactive, **object-oriented**, and high-level programming language. It was created by Guido van Rossum in 1991 at National Research Institute for Mathematics and Computer Science Netherlands.

Python IDLE

The python IDLE tool offers an interactive and a more efficient platform to write your code in python.

Python Modes

- Interactive Mode
- Script Mode

Different types of files in python:

.py, .pyc, .pyd, .pyo, .pyw, .pyz

Structure of a python program:

- **Expressions:** Expression is a combination of variables, operations and values that give a result value. Like: a+b
- **Statements**: Statements represent an action or command, Like **print statement**
- **Comments**: Comments are additional information provided for a statement. (#) is used for single line comment and (''' ''') used for multiple line comments.
- **Function**: A **function** is a block of code which only runs when it is called.
- **Blocks**: A **block** is a piece of **Python** program text that is executed as a unit.

Variable:

A Variable is like a container that stores any value.

x=3, age=30

Rules for valid Variable name:

- A variable name must start with a letter or the underscore character
- A variable name cannot start with a number
- A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and _)
- Variable names are case-sensitive (age, Age and AGE are three different variables)
- A variable name cannot contain spaces.

Multiple Assignments:

1) Assigning multiple values to multiple variables: x,y,z=2,3,4
2) Assigning same value to multiple variable: a=b=c=10

Components of Variable/Object:

A). Identity of the Variable / Object: It refers to the Variable's memory location address which is unchanged once it has been created. We can check memory location using this method:

id()

B). Type of the Variable / Object (data type):

int	float	Complex	str	boolean	None
5	5.5	2+5j	"hello"	True/False	None

We can seen any data type by: **type()**

y="Hello"

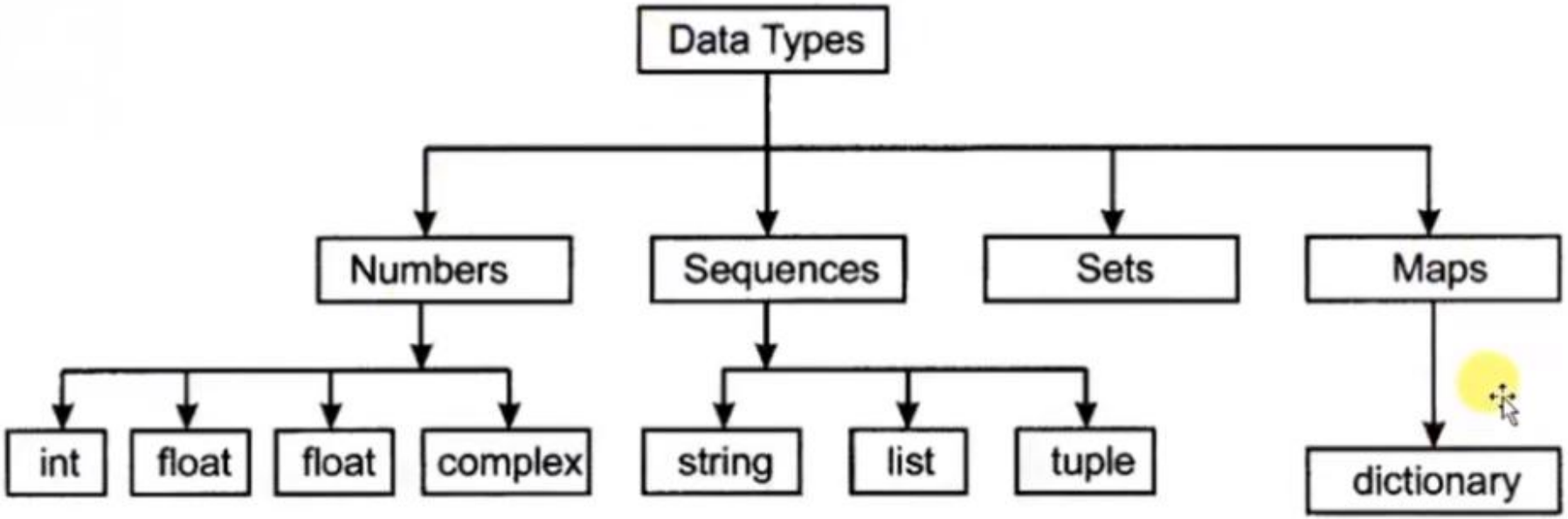
type(y)

C). Value of the Variable/Object:

The value stored in Variable like: age=20, so 20 is the value of variable.

L-value is age, and R-Value is 20

Data Types:



Python Data types

What is Dynamic Typing:

It refers to declaring a variable multiple times with values of different data types. And variable automatically changes its data types according to data.

```
a=10
a="Hello"
```

Chapter 1: Review of Python basic (Part 2)

- Keywords
- Mutable and Immutable Types
- Input and Output
- Types Casting
- Operators and Operands

Chapter 1: Review of Python basic (Part 2)

Keywords

Keywords are words that are already reserved for some particular purpose. The names of these keywords should not be used as identifiers/Variable. Keywords are also called reserved words.

```
import keyword

print(keyword.kwlist)
```

List of Keyword:

```
'False', 'None', 'True', 'and', 'as', 'assert',
'break', 'class', 'continue', 'def', 'del', 'elif', 'else',
'except', 'finally', 'for', 'from', 'global', 'if', 'import',
'in', 'is', 'lambda', 'nonlocal', 'not', 'or', 'pass', 'raise',
'return', 'try', 'while', 'with', 'yield'
```

Mutable and Immutable Types:

Mutable: A **mutable** object can be changed after it is created. Example: List, dictionary etc.

Immutable: A **immutable** object cannot be changed after it is created. Example: int, float, complex, bool, string, tuple etc.

Input and Output (Python built in functions)

1. input(): Its used to get user input in string format. ex: `input("Enter any number")`
2. eval(): Its used to evaluate string as a number if possible. `eval(input("Enter any number"))`
3. print(): Its used to display output on the screen. ex: `print(5)`

Types Casting

The process of converting the value of one data type (integer, string, float, etc.) to another data type is called type conversion. Python has two types of type conversion.

1. **Implicit Type Conversion:** Python automatically converts one data type to another data type. This process doesn't need any user involvement. `Ex: type(5/2)`
2. **Explicit (forced) Type Conversion:** In Explicit Type Conversion, users convert the data type of an object to required data type. `Ex: float(5), str(5)`

Operators and Operands

An **operator** is a symbol that performs an operation.

An operator acts on some variables called **operands**.

For example, if we write a + b, the operator + is acting on two operands a and b. Python provides some useful operators for performing various operations.

Binary Operators: **Operators** that operate on two operands are known as **binary operators**.

Ex: 3+3

Unary Operators: **Operators** that operate on one operand are known as **unary operators**.

Ex: -3

Types of Operators:

- Arithmetic operators
- Assignment operators
- Comparison operators
- Logical operators
- Identity operators
- Membership operators
- Bitwise operators

Flow of execution/Program Control and Flow

Control and flow of a program is classified into 4 categories:

1. Sequence Statement

2. Selection/Decision Statement

3. Iteration / Looping Statement

4. Jump Statement

1. Sequence: In this program executes in sequential order, one after another, without any jump in the program.

Ex: Program to calculate sum:

```
a=10
b=20
c=a+b
print(c)
```

What is String:

A string is a sequence of characters. We can enclose characters in quotes (single, double or triple.)

For example: `fruits='Mango Apple Grapes'`

Creating String /Types of string:

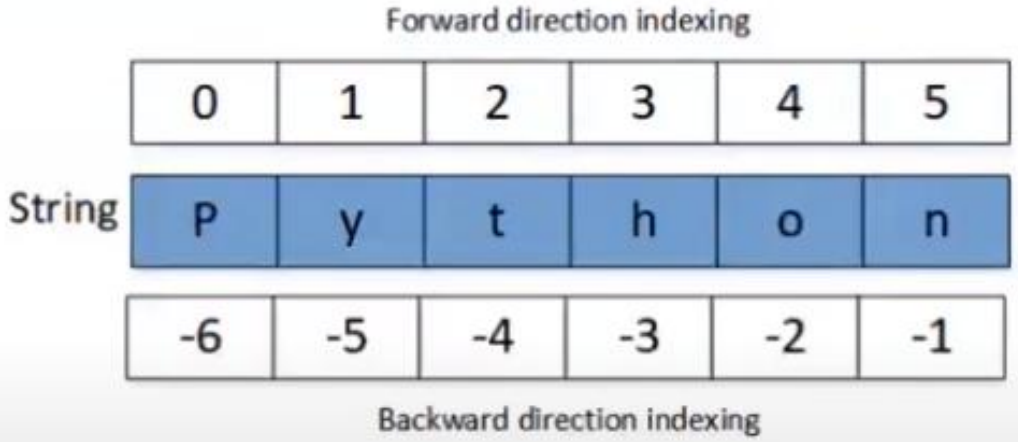
```
str1='Hello world'
str2="Python Programming"
str3='''Python
Programming'''
```

Empty String: An empty string is a string without any characters inside.

```
str=" "   ,   str=' '
```

Multiple line String: Multiline strings are represented using triple quotes (''' ''') or even single or double quotes.

Traversing a String: Traversing a string means accessing all the elements of the string one after the other by using the subscript / index value.



Special String Operators:

String Slicing: Slicing is used to retrieve a subset of values. Chunk of characters can be extracted from a string using slice operator. Example: var1 [start: end: step]

Updating: We can "update" an existing string by (re)assigning a variable to another string.

Example: var1 [range] +"x"

Concatenation: Concatenation refers to creating a new string by adding two strings. + is the operator to join two string.

Example: "Hello "+"world"

Repetition or Replicate: By repetition of string we can create multiple copies of any string.

* is the Operator for repetition.

Example: “Hello” *3.

Comparison Operators: Comparison operators are used to compare two strings. Python compares strings using ASCII or UNICODE.

Example:

"Tim"=="tim"

false

String Methods and Built-in-functions:

len()	index()	count()	lstrip()	join()
capitalize()	isalpha()	lower()	rstrip()	swapcase()
split()	isalnum()	islower()	strip()	partition()
replace()	isdigit()	upper()	isspace()	ord()
find()	title()	isupper()	istitle()	chr()

What is List:

A list is a collection of comma-separated values (items) within square brackets. Items in a list need not of the same type. It stores data in ordered sequence.

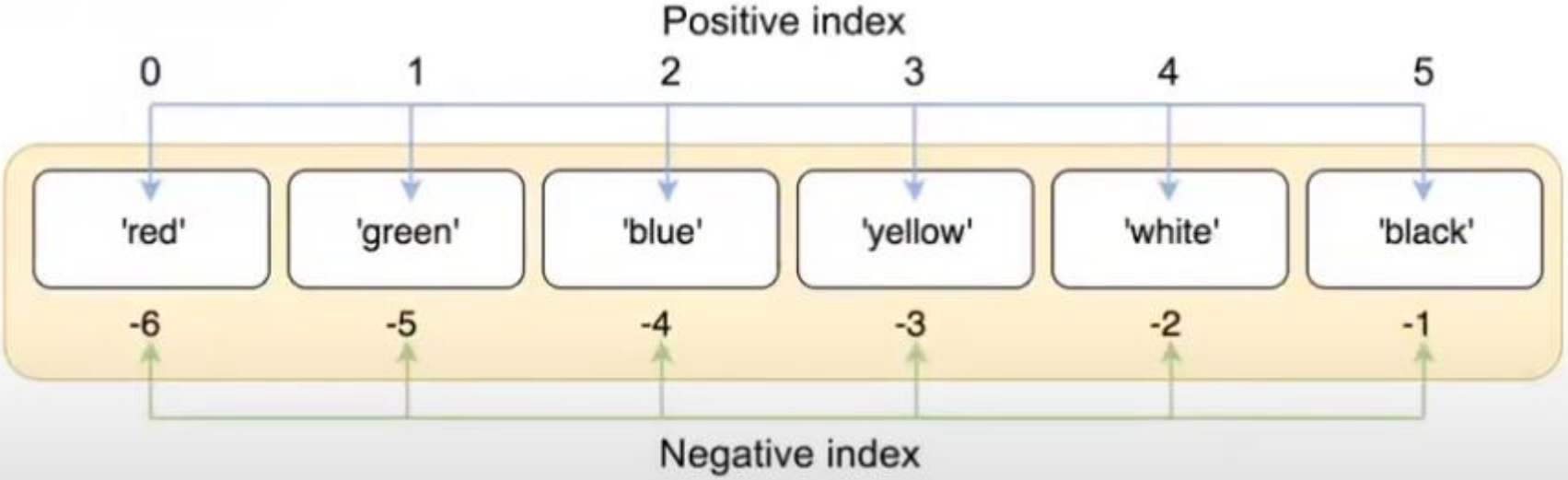
Declaring/Creating List:

Syntax to creating List:

List_name=["value1", "value 2", "value 3", "value 4"]

Nested_List=["value1", ["value 2", "value 3"], ["value 2", "value 3"]]

Accessing List Elements:



Traversing a list:

Traversing a list means accessing each element of a list. This can be done by using either **for** or **while** looping statement.

What is Tuple:

Tuple is a sequence of immutable Python object. Tuples are sequences, just like lists. Tuples are immutable it means we cannot perform insert, update and delete operation on them.

The only difference is that tuples are immutable , tuples use parentheses, and lists use square brackets.

Example: tuple= ('mango','apple','grapes')
Example: List= ['mango','apple','grapes']

Tuple creation:

Empty tuple: t=()

Tuple with one element: t=(10,)

Tuple with one element: t=10,

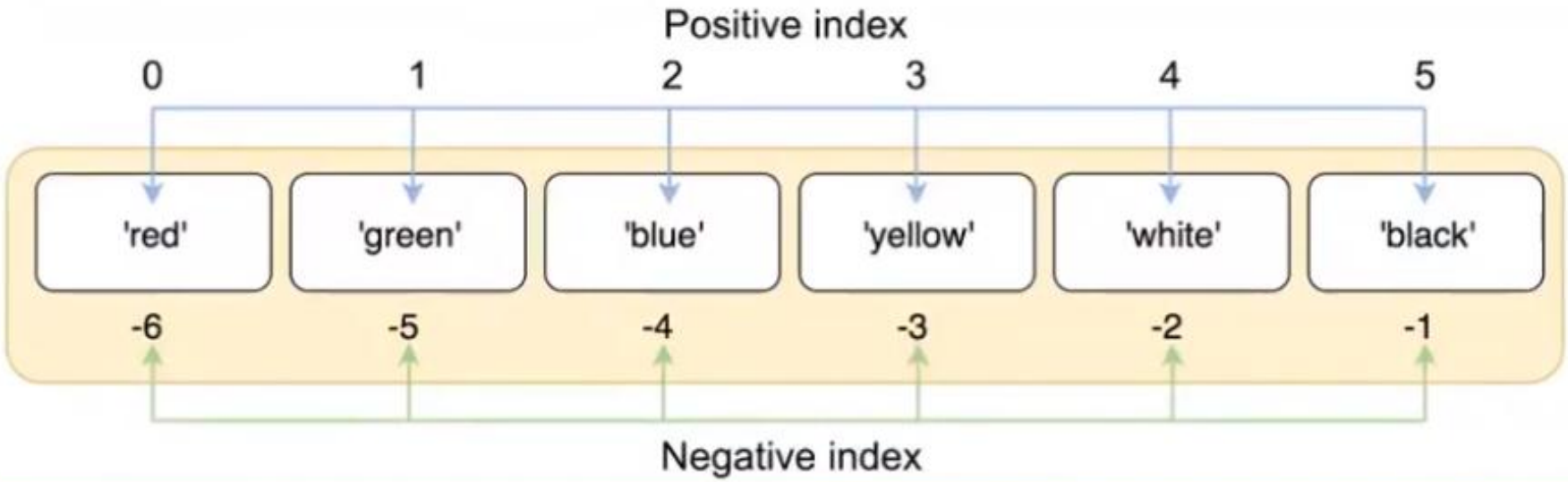
Tuple with multiple value: t(10, “Hello”,20,”Kanpur”)

Creating tuple using function tuple()

Nested Tuple:

Example: fruits= (10 , 20 , (‘Ram’ , ’Mohan’) , 40 , 50 , [‘Red’ , ’blue’] , 60)

Accessing a Tuple and Nested Tuple:



Traversing a Tuple:

Traversing a tuple means accessing each element of a tuple. This can be done by using either **for** or **while** looping statement.

What is Dictionary:

A python dictionary is a collection of key-value pairs. Python dictionary is an unordered collection of items.

d1 = { 'Input':'Keyboard','Output':'printer','Language':'python','OS':'Windows'}

Dictionaries are mutable which means they can be changed.

Creating a Dictionary:

Empty dictionary:

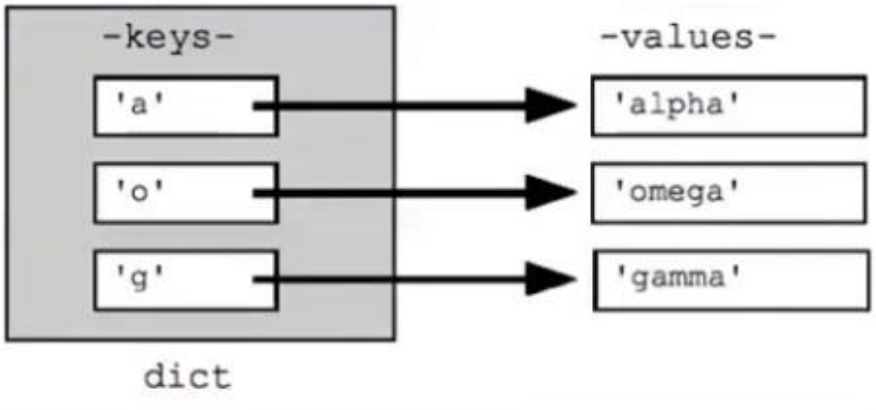
d={}

d=dict()

We can store int, float, string, list or tuple in dictionary as value of key.

Accessing a Dictionary:

To access dictionary elements, you can use the square brackets along with the key to obtain its value.



Traversing a Dictionary:

Traversing a dictionary means accessing each element of a Dictionary. This can be done by using **for** loop statement.

What is Functions:

A **function** is a block of code which only runs when it is called.

Advantages of Functions:

1. We can avoid rewriting the same logic/code again and again in a program.
2. We can call Python functions multiple times in a program.
3. Code becomes reusable.

Types of Function:

1. **Built-in-Functions:** It is the predefined functions that are already available in python.
   Ex: int(), str() , float(), input(), eval(), min(), max(), abs(), type(), len(), round(), range() print().

2. **User defined Functions:** The functions defined by user to perform specific task is called user defined functions.

3. **Modules:** A module is a file containing functions and variables defined in separate files.
   Ex: math module, random module, statistics module

User-defined functions:

The functions defined by user to perform specific task is called user defined functions.

How to define and call a function in python function:

```
def function_name():
    statement
    print(value)
```

Function Calling/invoking

```
function_name()
```

How function return a value

```
def function_name():
    statement
    return value
```

Function Calling/invoking

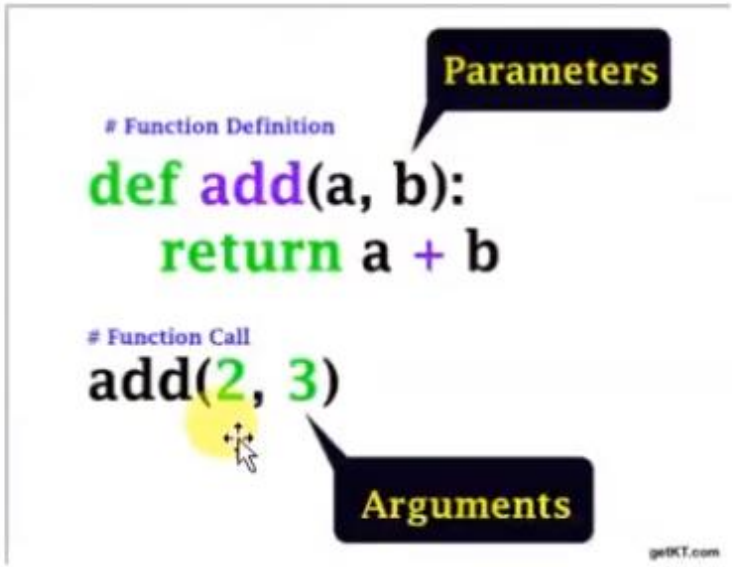
```
print(function_name())
```

(We can store return value in variable)

Parameters and arguments in functions:

Parameters: The values received in the function header are called parameter. This is also known as **formal parameter** and **formal argument**.

Arguments: The values being passed through a function-call statement are called arguments.
This is also known as **actual parameter** and **actual argument**.



Types of arguments:

1. **Positional argument:** Positional argument is arguments passed to a function in correct positional order.
2. **Default argument:** It is used to give default value if value is not provided in the function calling.
3. **Keyword argument:** In this we use the name or **keyword** instead of the position to specify the **arguments** to the function.
4. **Variable length argument:** **Variable length argument** is a feature that allows a function to receive any number of **arguments**. `def (*n)`

Scope of Variables:

Scope of variable refers to the part of the program where it is visible or area where you can use it.

Local Scope: A variable created inside a function belongs to the *local scope* of that function, and can only be used inside that function.

```
a=2
def f(x):
    y=x+a
    return y
p=f(5)
print(p)
```

Global variable

Local variable

Global Scope: A variable created inside a function belongs to the *local scope* of that function, and can only be used inside that function.

Global keyword: Name declared with global keyword in a function can be accessed inside or outside of the function