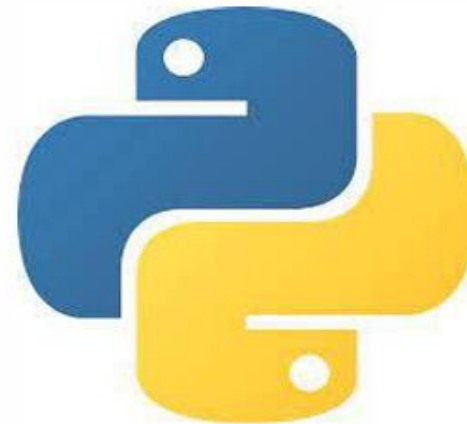






MONTY PYTHON'S  
FLYING  
CIRCUS



# Using C Program

```
// C program to add two numbers
#include<stdio.h>

int main()
{
    int A, B, sum = 0;

    // Ask user to enter the two numbers
    printf("Enter two numbers A and B : \n");

    // Read two numbers from the user || A = 5, B = 7
    scanf("%d%d", &A, &B);

    // Calculate the addition of A and B
    // using '+' operator
    sum = A + B;

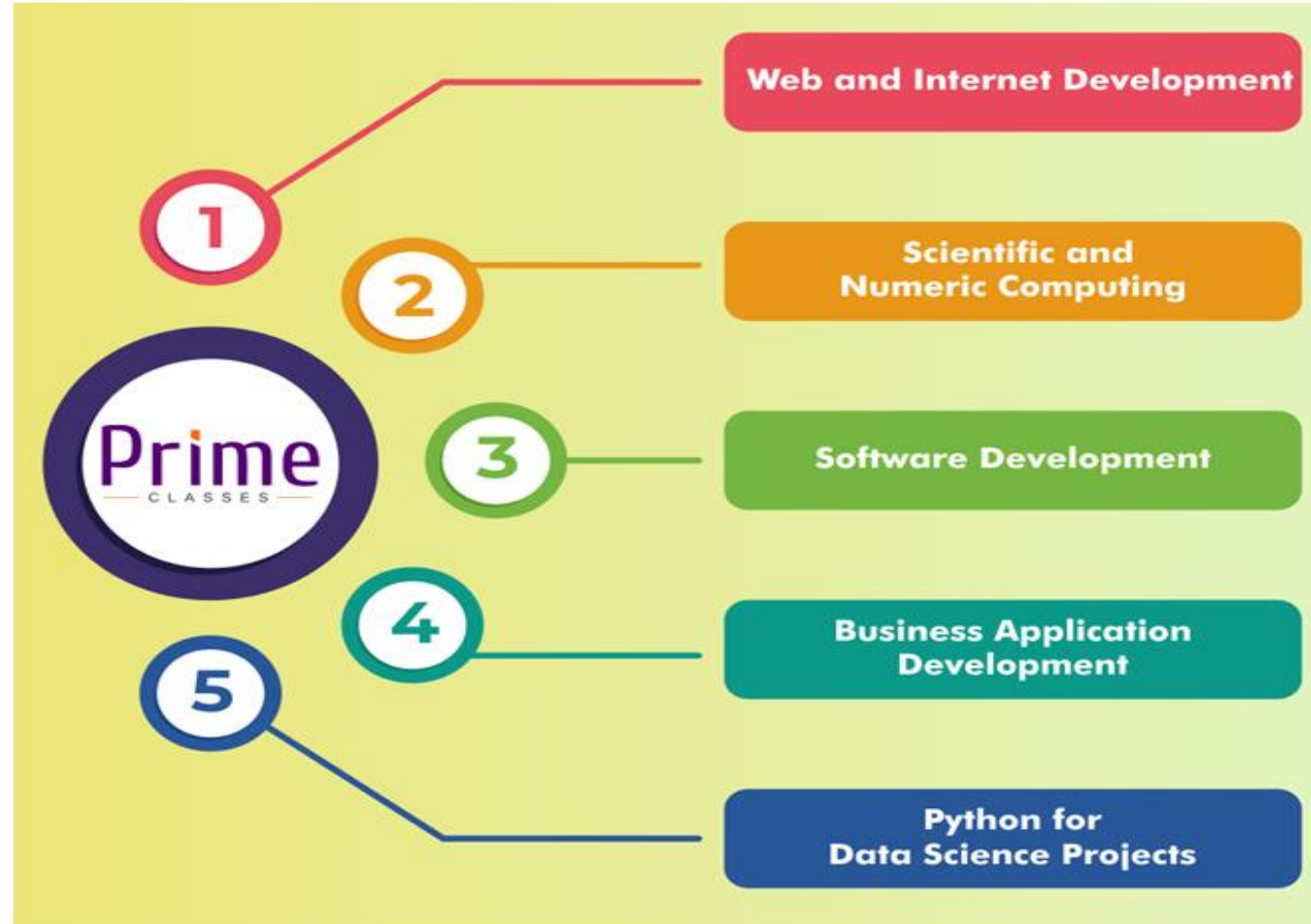
    // Print the sum
    printf("Sum of A and B is: %d", sum);

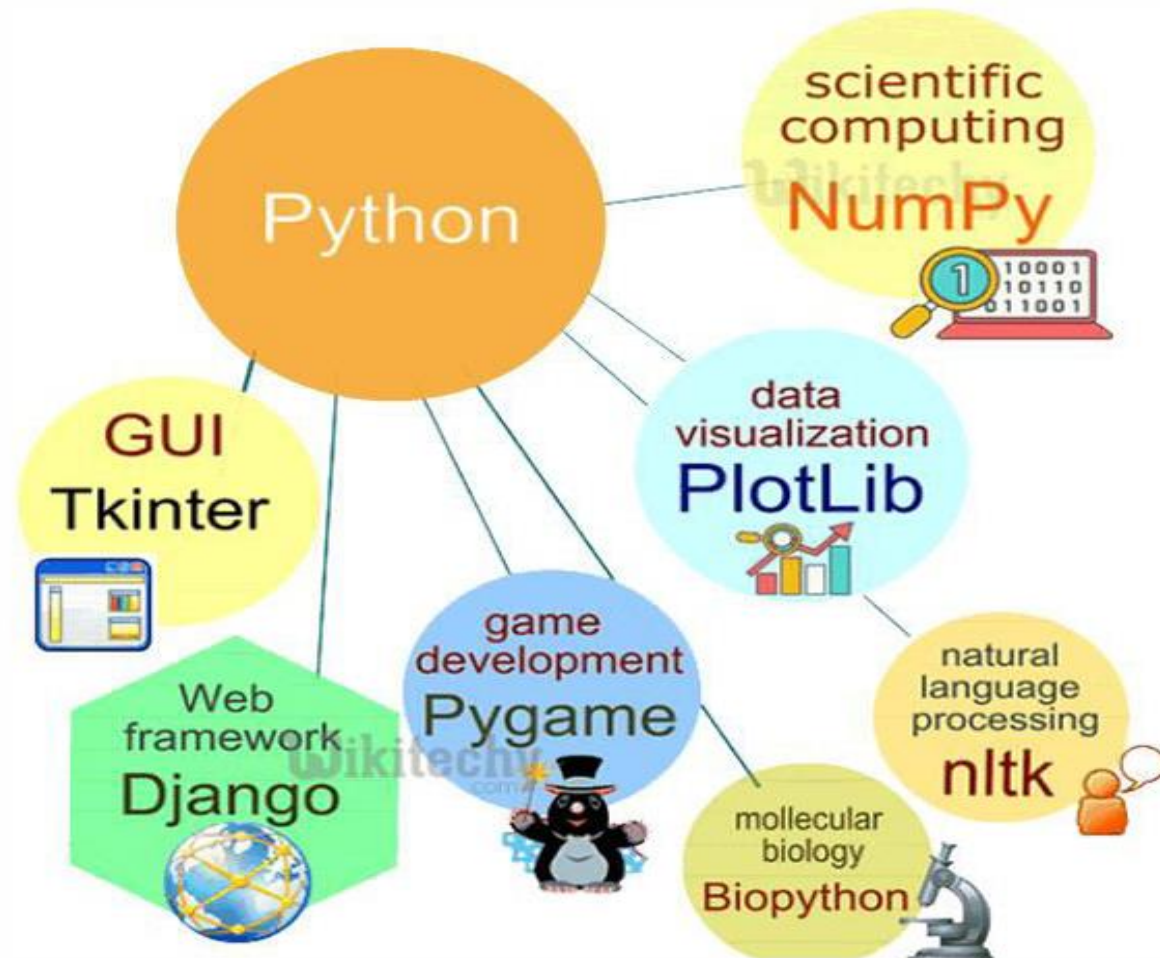
    return 0;
}
```

# Top Companies using Python



# Python Applications







# Tools for Implementing Python Code

---



---

# Python Introduction



# What is Python?

---

- Python is a popular programming language. It was created by Guido Van Rossum, and released in 1991.
- It is used for:
  - Web Development (server – side),
  - Software Development,
  - Mathematics,
  - System Scripting.

# What can python do?

---

- Python can be used on a server to create web applications.
- Python can be used alongside software to create workflows.
- Python can connect to database systems. It can also read and modify files.
- Python can be used to handle big data and perform complex mathematics.
- Python can be used for rapid prototyping, or for production-ready software development.

# Why Python?

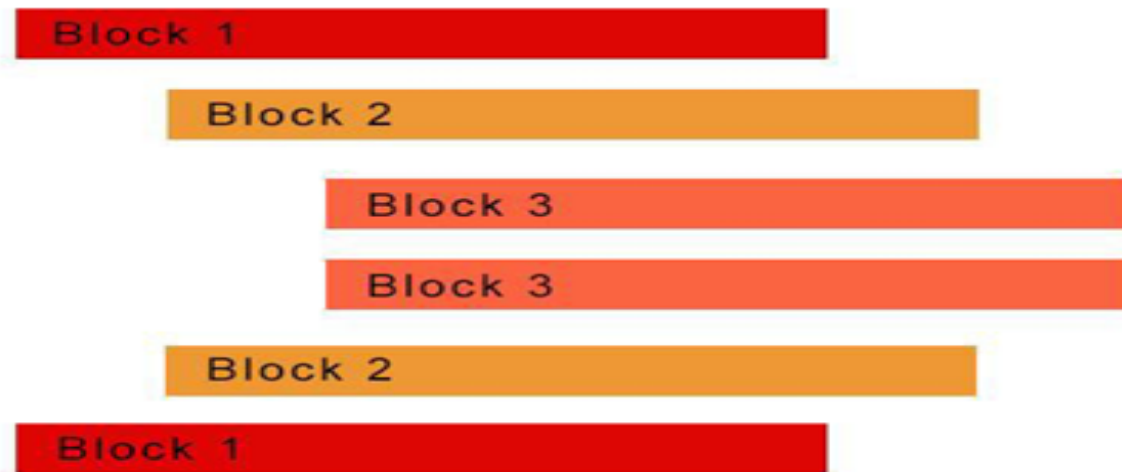
---

- Python works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc).
- Python has a simple syntax similar to the English language.
- Python has syntax that allows developers to write programs with fewer lines than some other programming languages.
- Python runs on an interpreter system, meaning that code can be executed as soon as it is written. This means that prototyping can be very quick.
- Python can be treated in a procedural way, an object-orientated way or a functional way.

# Python Indentation

---

- Indentation refers to the spaces at the **beginning of a code line**.
- Where in other programming languages the indentation in code is for **readability only**, the indentation in Python is very important.
- Python uses indentation to **indicate a block of code**.

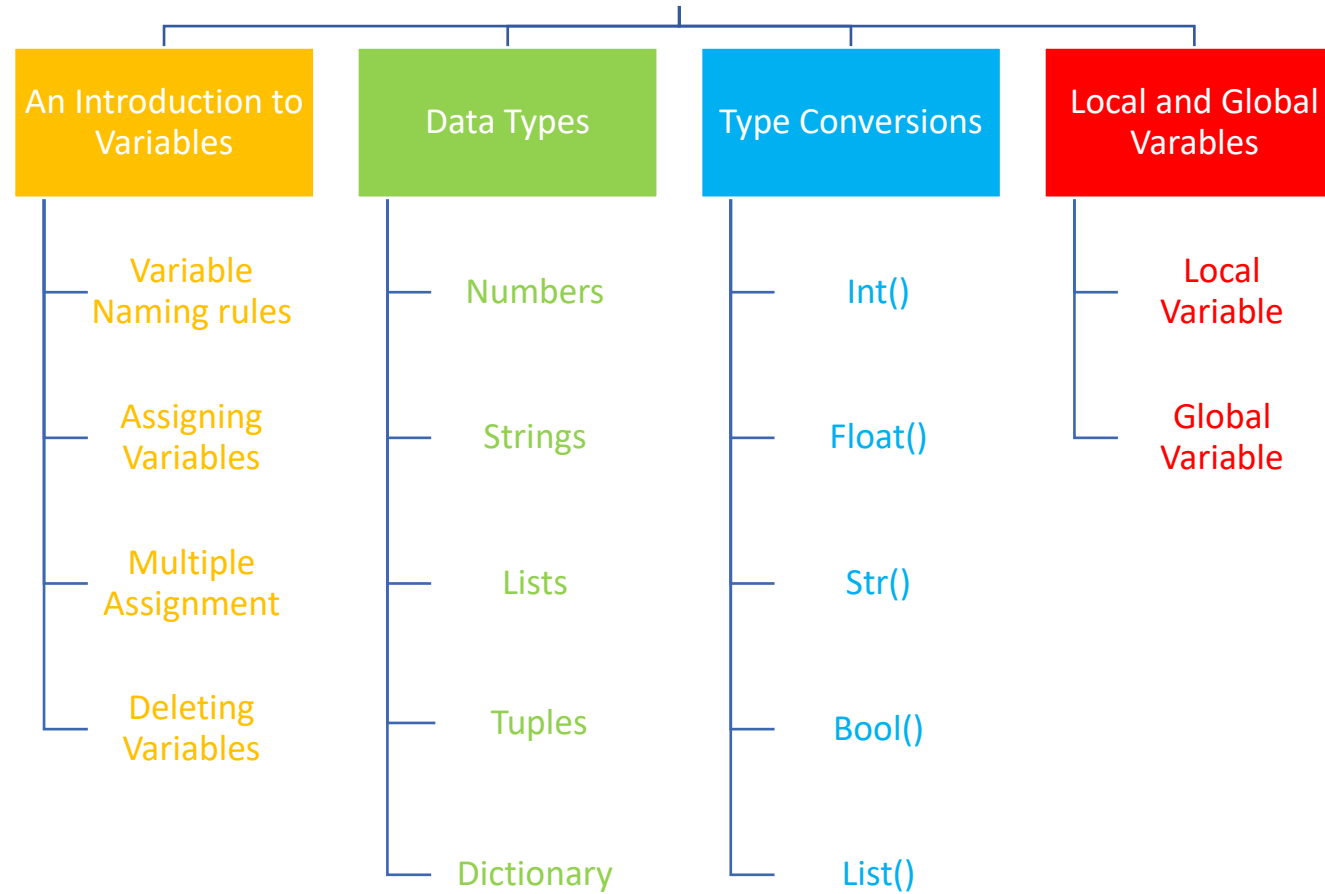


# Python Comments

---

- **Comments can be used to**
  - **explain Python code.**
  - **make the code more readable.**
  - **prevent execution when testing code.**

# Python Variables & Data Types



# Creating Variables

---

- Variables are **containers** for **storing data values**.
- Unlike other programming languages, Python has **no command** for declaring a variable.
- A variable is **created** the moment you first **assign a value to it**.
- Variables do not need to be declared with any particular type and can even change type after they have been set.

String variables can be declared either by using single or double quotes:



# Variable Names

---

A variable can have a **short name** (like x and y) or a more **descriptive name** (age, carname, total\_volume).

## Rules for Python variables:

- 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)

# Assign Values to Multiple Variables

---

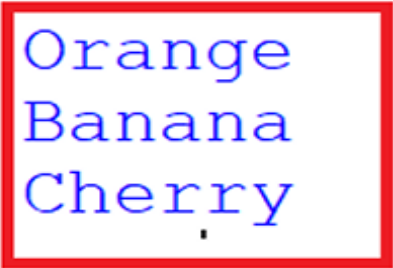
- Python allows to assign values to multiple variables in one line:

## Example 4:

---

```
x, y, z = "Orange", "Banana", "Cherry"
```

```
print(x)  
print(y)  
print(z)
```



```
Orange  
Banana  
Cherry
```

---

And you can assign the *same* value to multiple variables in one line:

### Example 5:


---

```
x = y = z = "Orange"
```

```
print(x)
```

```
print(y)
```

```
print(z)
```



```
Orange  
Orange  
Orange
```

# Output Variables

---

- The Python `print` statement is often used to output variables.
- To combine both text and a variable, Python uses the `+` character:

## Example 6:

```
x = "awesome"  
print("Python is " + x)
```

Python is awesome

- You can also use the + character to add a variable to another variable:

```
x = "Python is "  
y = "awesome"  
z = x + y  
print(z)
```

```
Python is awesome
```

- For numbers, the + character works as a mathematical operator:

```
x = 5  
y = 10  
print(x + y)
```

```
15
```

- 
- If you try to combine a string and a number, Python will give you an error:

```
x = 5  
y = "John"  
print(x + y)
```

```
TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

## Arithmetic Calculations

Operators used for calculations

-

1. Addition ----- +
2. Subtraction - -
3. Multiplication - \*
4. Float Division - /
5. Integer Division - //
6. Exponent - \*\*
7. Remainder - % (Modulus)
8. `round(2**0.5, 5)`

```
print(2+3+10)
print(2/4)
print(2//4)
print(2**4)
print(round(2**0.5, 5))
print(3%2)
print((2+3)*5/8%6)
print(2+3*5)
```



## Printing Emoji

---

```
print("\U0001F600")
```

- for printing emoji, we have to use Unicode of emoji from browser
- in place of +, we have to add three zeros and in front of U add back slash compulsory clear

## Escape Sequences

---

```
# output : "line a \n line b "  
print("Line A \n Line B")  
# treat as normal text  
print("this is double backslash \\")  
# output : \" \"  
print(" \" \" \" \" ")
```

```
# \' - '  
# \\ - \  
# '\\\' - \'
```

```
#raw strings
```

```
print(r" Indian \n air\t force")
```

```
# *****print following lines*****
```

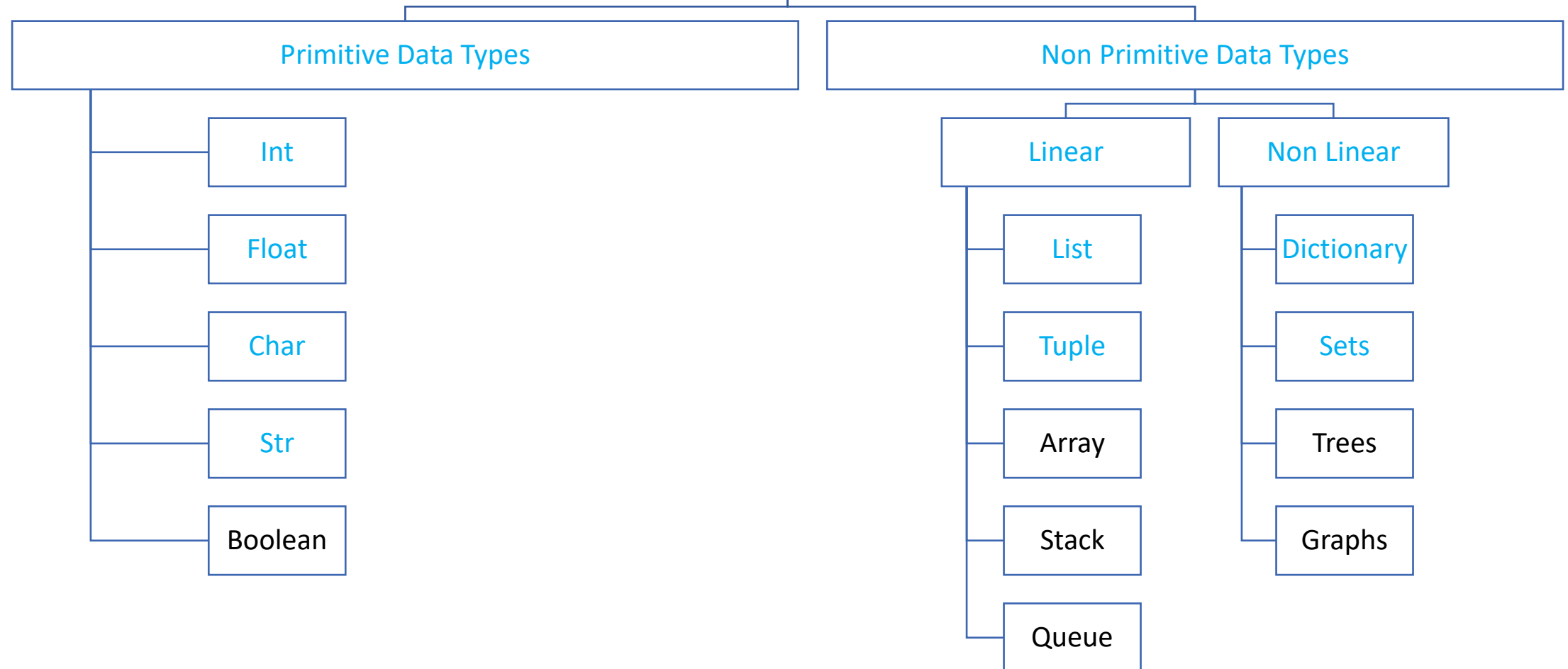
```
# this is \\ double backslash
```

```
#these are /\ /\ /\ mountains
```

```
#he is     awesome (use escape sequence  
instead of manual space)
```

```
# \" \n \t \' (print this as an output)
```

# Python Data Types



# Built-in Data Types

---

- In programming, data type is an important concept.
- Variables can store data of different types, and different types can do different things.

<b>Text Type:</b>	<b>str</b>
<b>Numeric Types:</b>	<b>int, float, complex</b>
<b>Sequence Types:</b>	<b>list, tuple, range</b>
<b>Mapping Type:</b>	<b>dict</b>
<b>Set Types:</b>	<b>set, frozenset</b>
<b>Boolean Type:</b>	<b>bool</b>
<b>Binary Types:</b>	<b>bytes, bytearray, memoryview</b>

Example	Data Type
<code>x = "Hello World"</code>	<code>str</code>
<code>x = 20</code>	<code>int</code>
<code>x = 20.5</code>	<code>float</code>
<code>x = 1j</code>	<code>complex</code>
<code>x = ["apple", "banana", "cherry"]</code>	<code>list</code>
<code>x = ("apple", "banana", "cherry")</code>	<code>tuple</code>
<code>x = range(6)</code>	<code>range</code>
<code>x = {"name" : "John", "age" : 36}</code>	<code>dict</code>
<code>x = {"apple", "banana", "cherry"}</code>	<code>set</code>
<code>x = frozenset({"apple", "banana", "cherry"})</code>	<code>frozenset</code>
<code>x = True</code>	<code>bool</code>
<code>x = b"Hello"</code>	<code>bytes</code>
<code>x = bytearray(5)</code>	<code>bytearray</code>
<code>x = memoryview(bytes(5))</code>	<code>memoryview</code>

K Vinod

## Getting the Data Type

---

You can get the data type of any object by using the **type()** function:

```
x = 5  
print(type(x))
```

```
<class 'int'>
```

# Python Numbers



There are three numeric types in Python:

- **Int** : Int, or integer, is a whole number, positive or negative, without decimals, of unlimited length. (x = 1 )
- **Float** : Float, or "floating point number" is a number, positive or negative, containing one or more decimals. (y = 2.8)
- **Complex** : Complex numbers are written with a "j" as the imaginary part:  
(z = 1j )

# Type Conversions

## 1.2 Strings

You can convert from one type to another with the `int()`, `float()`, and `complex()` methods:

```
#convert from int to float:
```

```
x = float(1)
```

```
#convert from float to int:
```

```
y = int(2.8)
```

```
#convert from int to complex:
```

```
z = complex(x)
```

```
print(x)
```

```
print(y)
```

```
print(z)
```

```
1.0
```

```
2
```

```
(1+0j)
```

```
<class 'float'> <class 'int'> <class 'complex'>
```

**Note:** You cannot convert complex numbers into another number type.

# Random Number

## 1.2 Strings

Python does not have a `random()` function to make a random number, but Python has a built-in module called `random` that can be used to make random numbers:

```
import random  
  
print(random.randrange(1, 10))
```



6

# Python Strings

- String literals in python are surrounded by either single quotation marks, or double quotation marks.

ex: 'hello' is the same as "hello".

You can display a string literal with the `print()` function:

```
print("Hello")  
print('Hello')
```

```
Hello  
Hello
```

- Assigning a string to a variable is done with the variable name followed by an equal sign and the string:

```
a = "Hello"  
print(a)
```

Hello

- You can assign a multiline string to a variable by using **three quotes** (single or double):

Three double quotes: " " "      " " "

Example : You can use three double quotes:

```
a = """start with a letter
cannot start with a number
contain alpha numeric characters
case-sensitive """
print(a)
```

```
start with a letter
cannot start with a number
contain alpha numeric characters
case-sensitive
```



### Three single quotes: ''' '''

Example : You can use three single quotes:

```
a = '''start with a letter  
cannot start with a number  
contain alpha numeric characters  
case-sensitive '''  
print(a)
```

```
start with a letter  
cannot start with a number  
contain alpha numeric characters  
case-sensitive
```

# Python User Input

- In Python, the `input()` function is used to get user input from the keyboard.
- When you call `input()`, the program pauses and waits for the user to type something and press Enter.
- The `input` is then returned as a `string`.
- Note: By default, `input()` always returns the user input as a string

If you need a different type (like an integer or a float), you'll need to convert the string using functions like `int()` or `float()`.

```
name = input("Type your name")
print("Hello" + name)
age = input("enter your age") # it takes as "24", not a number
print("your age is " + age)
```

Multivalues Assignment using input ----- split method

```
name , age , gender = input("enter your name , age , gender : ").split(",")
print(name)
print(age)
print(gender)
```

```
name = input("enter your name : ")
age = input("enter your age:")
print("hello" + name + "age is" + str(age))  #ugly syntax
# Python2
# python3
# python 3.6 (best)
print("hello {} your age is {}".format(age, name))  #python3
print(f"hello {name} your age is {age} ")  #never forgot to use (f)
# # you can do calculations also age + 5
```

```
language = "python"  
# p = 0 , -6  
# y = 1 , -5  
# t = 2 , -4  
# h = 3 , -3  
# o = 4 , -2  
# n = 5 , -1  
print(language[2])  
print(language[-1])  
print(language[-4])
```

# String Slicing

## 1.2 Strings

```
lang = "python"
```

```
# syntax = [start argument : stop argument - 1]
print(lang[-6:-4])
print(lang[:])
print(lang[:3])
print(lang[1:])
```

## Step Argument Slicing

```
# syntax = [start argument : stop argument -1 : step]
print("vinod"[0:5])
print("vinod"[0:5:1])
print("vinod"[0:5:2])
print("vinodhghdsgfhdshgfdh"[0::3])
# reverse
print("vinod"[4::-1])
```

# String Method 1

## 1.2 Strings

```
name = "INDIAN army"
# len () function : to count the length of the string
length = len(name)
print(length)

# lower() method {difference between function and method}
# method we have to use . dot
print(name.lower())

# 3. upper() method
print(name.upper())
# 4. title () method {most useful}
print(name.title())
# 5. count()
print(name.count("a"))
```



## Strip Method

```
name = "    india  ARMY  "  
dots = "....."  
print(name + dots)  
# to remove spaces  
# print(name.lstrip() + dots)  
# print(name.rstrip() + dots)  
# print(name.strip() + dots)
```

## Replace Method

```
middle = "ind  ia"  
print(middle.replace(" ", ""))  
  
# replace() syntax = replace("old" , "new" , count)  
# find() syntax = find("string" , start , end)  
  
string = "she is beautiful and she is a singer"  
print(string.replace(" ", "_" , 3))  
is_pos1 = string.find("is")  
print(string.find("is" , is_pos1 + 1))
```

### # string are immutable

```
string = "strings"  
print(string[1])
```

# in ruby ---> strings are mutable

```
name = "he"  
# name = name + "she" ///// name += "she"  
# print(name)  
name += "she"  
print(name)
```

```
age = 23  
age -= 2  
print(age)
```

# Thank You

