Python

1. Introduction to python programming :- Python is a popular programming language created by Guido van Rossum, and released in 1991, that is used on a server to create web applications.

2. Python Indentation :- Indentation refers to the spaces at the beginning of a code line. It is very important in python which is used to indicate a block of code.

   Example:

```
if 5 > 2:
      print("Five is greater than two!")
```

3. Variables and Data types :- Variables are containers for storing data values. Variables are created when you assign a value to it.

   Variable names are case-sensitive i.e a and A is different

   Variables in Python:

   X = 5
   Y = “Hello, World!”

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

```
x, y, z = "Orange", "Banana", "Cherry"
print(x)
print(y)
print(z)
```

   Python has no command for declaring a variable.

   Python has the following data types:

Text Type:	`str`
Numeric Types:	`int`, `float`, `complex`
Sequence Types:	`list`, `tuple`, `range`

Mapping Type:	`dict`
Set Types:	`set`, `frozenset`
Boolean Type:	`bool`
Binary Types:	`bytes`, `bytearray`, `memoryview`
None Type:	`NoneType`

The type() function is used to check the datatype of variables.

4. Variable Names :- A variable name must start with a letter or the underscore character, cannot start with number, must contain alpha-numeric characters and cannot be any of the python keywords.

5. Python String :-

   a. Python strings
   String variables can be declared either by using single or double quotes.
   Three double quotes are used to assign a multiline string to a variable.
   Strings in Python are arrays of bytes representing Unicode characters.

```
a = "Hello, World!"
print(a[1])
```

   Since strings are arrays, we can loop through the characters in a string, with a for loop.

```
for x in "banana":
  print(x)
```

   The `len()` function returns the length of a string:

   b. Slicing Strings
   Slicing is the process of extracting a portion of a string. You can use slicing to obtain a substring from a given string. It is done using square brackets and colons.

```
my_string = "Hello, World"
sub_string = my_string[7:12]  # Retrieves "World"
```

c. Modify Strings

In Python, strings are immutable, meaning you cannot change the characters in a string directly. To modify a string, you need to create a new string with the desired changes.

```
my_string = "Hello"
modified_string = my_string + " World"  # Creates a new string with "Hello World"
```

d. Concatenate Strings

Concatenation is the process of combining two or more strings to create a single string. You can use the `+` operator for concatenation.

```
str1 = "Hello"
str2 = " World"
result = str1 + str2  # Combines str1 and str2 into "Hello World"
```

e. Format Strings:

String formatting allows you to insert variables or values into a string in a structured way. Python offers various methods for string formatting, including f-strings, `str.format()`, and `%` formatting.

```
name = "Alice"
age = 30
formatted_string = f"My name is {name} and I am {age} years old."
```

f. Escape Characters:

Escape characters are used to insert special characters in strings. They start with a backslash (`\`). Common escape characters include `\"` for double quotes, `\'` for single quotes, `\\` for a literal backslash, and `\n` for a newline character.

```
my_string = "This is a \"quoted\" string"
```

g. String Methods:

Python provides a wide range of built-in string methods to manipulate and process strings. Some common methods include:

- `upper()`: Converts a string to uppercase.
- `lower()`: Converts a string to lowercase.
- `strip()`: Removes leading and trailing whitespace.
- `split()`: Splits a string into a list based on a specified delimiter.
- `replace()`: Replaces one substring with another.
- `find()`: Searches for a substring and returns its position.

```
my_string = "Hello, World"
upper_string = my_string.upper()  # Converts to uppercase
words = my_string.split(", ")  # Splits the string into a
list of words
```

These string operations and concepts are fundamental when working with text data in Python. They allow you to manipulate and work with strings effectively in your programs.

6. Multi Words Variable Names :-

   a. Camel Case
   ```
   myVariableName = "Nisha"
   ```
   b. Pascal Case
   ```
   MyVariableName = "Nisha"
   ```
   c. Snake Case
   ```
   my_variable_name = "Nisha"
   ```

7. Comments :- # is used for single line comment and “”” ……””” is used for multiline comment.

8. Casting :- If you want to specify the data type of a variable, this can be done with casting.

```
x = str(3)    # x will be '3'
y = int(3)    # y will be 3
z = float(3)  # z will be 3.0
```

9. Global Variables :- Variables that are created outside of a function are known as global variables and variables that are created inside of a function are known as Local variable. It can be used by everyone, both inside of functions and outside.

```
x = "Global Variable"

def myfunc():
  x = "Local Varible"
  print("I am " + x)

myfunc()

print("I am " + x)
```

Output :-

I am Local Variable

I am Global Variable

10. Python Numbers :- There are three numeric types in Python: int , float, complex

```
x = 1    # int
y = 2.8  # float
z = 1j   # complex
```

11. Random Number :- 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))
```

12. List:
    a. A list is an ordered and mutable collection of elements in Python. It is defined by enclosing elements within square brackets ([]) and separating them with commas.
    b. Lists allow you to store different data types, and you can change their contents (mutable).
    c. Lists are indexed, meaning you can access elements by their position, starting from 0.
    d. Common list operations include appending, extending, inserting, removing, and slicing elements.

```
my_list = [1, 2, 3, "apple", "banana"]
my_list[3] = "cherry"  # Modifies the element at index 3
```

13. Tuple:
    a. A tuple is an ordered and immutable collection of elements in Python. It is defined by enclosing elements within parentheses (()) and separating them with commas.
    b. Tuples are similar to lists but cannot be changed once created (immutable).
    c. They are often used for data that should not change, such as coordinates or settings.

```
my_tuple = (1, 2, 3, "apple", "banana")
# my_tuple[3] = "cherry"  # This will raise an error because tuples are immutable
```

14. Set:
    a. A set is an unordered and mutable collection of unique elements in Python. It is defined by enclosing elements within curly braces ({}) or using the set() constructor.
    b. Sets are useful for tasks that require uniqueness and mathematical set operations like union, intersection, and difference.
    c. They do not allow duplicate values.

```
my_set = {1, 2, 3, 3, 4, 4, 5}  # Creates a set with unique values
my_set.add(6)  # Adds an element to the set
```

15. Dictionary:
    a. A dictionary is an unordered and mutable collection of key-value pairs. Each key is unique and maps to a specific value. Dictionaries are defined by enclosing key-value pairs within curly braces ({}) and separating them with colons (:).

b. Dictionaries are often used for storing and retrieving data using a specific key.

```
my_dict = {"name": "John", "age": 30, "city": "New York"}
my_dict["age"] = 31  # Modifies the value associated with the "age" key
```

16. Difference between List, Tuple, Set and dictionary:

Characteristic	List	Tuple	Set	Dictionary
Mutability	Mutable	Immutable	Mutable	Mutable
Ordering	Ordered	Ordered	Unordered	Unordered
Uniqueness	Duplicates allowed	Duplicates allowed	Unique elements	Unique keys
Syntax	Square brackets	Parentheses	Curly braces	Curly braces with key-value pairs
Typical Use Cases	General-purpose	Immutable data	Unique elements, set operations	Key-value storage