1. What is Python and why is it popular?

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable.

* **Python is Interpreted** − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it
* **Python is Interactive** − You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
* **Python is Object-Oriented** − Python supports Object-Oriented style or technique of programming that encapsulates code within objects.

**Popular**

* The easy to Learn and Use
* Mature and Supportive Python Community
* Versatility, Efficiency, Reliability, and Speed
* Big data, Machine Learning and Cloud Computing
* First-choice Language
* The Flexibility of Python Language

2.What are the differences between Python 2 and Python 3?

| **Comparison Parameter** | **Python 2** | **Python 3** |
| --- | --- | --- |
| “Print” Keyword | Print is considered to be a statement and not a function. | Print is considered to be a function and not a statement. |
| Storage of Strings | Strings are stored as ASCII by default. | Strings are stored as UNICODE by default. |
| Division of Integers | On the division of two integers, we get an integral value in Python 2. For instance, 7/2 yields 3 in Python 2. | On the division of two integers, we get a floating-point value in Python 3. For instance, 7/2 yields 3.5 in Python 3. |
| Exceptions | Exceptions are enclosed in notations. | Exceptions are enclosed in parentheses. |
| Variable leakage | The values of global variables do change in Python 2 if they are used inside a for-loop. | The value of variables never changes in Python 3. |
| Ease of Syntax | Python 2 has more complicated syntax than Python 3. | Python 3 has an easier syntax compared to Python 2. |
| Libraries | A lot of libraries of Python 2 are not forward compatible. | A lot of libraries are created in Python 3 to be strictly used with Python 3. |

3.What is the difference between a tuple and a list in Python?

| **LIST** | **TUPLE** |
| --- | --- |
| The list is mutable | The tuple is immutable |
| The implication of iterations is Time-consuming | The implication of iterations is comparatively Faster |
| The list is better for performing operations, such as insertion and deletion. | Tuple data type is appropriate for accessing the elements |
| Lists consume more memory | Tuple consumes less memory as compared to the list |
| Lists have several built-in methods | Tuple does not have many built-in methods. |
| The unexpected changes and errors are more likely to occur | In tuple, it is hard to take place. |

Example:

# Create a list of fruits

fruits\_list = ['apple', 'banana', 'orange']

fruits\_list.append('pear')

fruits\_list[0] = 'kiwi'

Output: ['kiwi', 'banana', 'orange', 'pear']

# Create a tuple of fruits

fruits\_tuple = ('apple', 'banana', 'orange')

fruits\_tuple.append('pear')

fruits\_tuple[0] = 'kiwi'

Output: ('apple', 'banana', 'orange')

4.How do you create a dictionary in Python?

Python dictionary is an ordered collection of items. It stores elements in **key/value** pairs. Here, **keys** are unique identifiers that are associated with each **value**.

Here, **keys** and **values** both are of string type. We can also have **keys** and **values** of different data types.

Example:

capital\_city = {"Nepal": "Kathmandu", "Italy": "Rome", "England": "London"}

O/P:

{'Nepal': 'Kathmandu', 'Italy': 'Rome', 'England': 'London'}

Keys: "Nepal", "Italy", "England"

Values: "Kathmandu", "Rome", "London"

5.What is a function in Python and how do you define one?

A function is a block of organized, reusable code that is used to perform a single, related action. Functions provide better modularity for your application and a high degree of code reusing.

Example:

def my\_function():

print ("Hello from a function")

my\_function()

6.What is object-oriented programming (OOP) and how does it relate to Python?

* Python is a multi-paradigm programming language, meaning it supports OOP as well as other paradigms.
* Python provides all the standard features of object-oriented programming.
* Developers often choose to use OOP in their Python programs because it makes code more reusable and makes it easier to work with larger programs.
* OOP programs prevent you from repeating code because a class can be defined once and reused many times.

1. Data Encapsulation
2. Polymorphism
3. Aggregation
4. Operator Overloading
5. Information Hiding

7.How do you handle exceptions in Python?

Exception handling in Python is achieved using the try and except blocks.

The try block is used to enclose the code that may raise an exception, while the except block is used to handle the exception that may occur. If an exception is raised in the try block,

the program jumps to the except block to handle it, instead of halting the program altogether.

Syntax:

try:

# code that may raise an exception

except:

# code to handle the exception

8.How do you read and write files in Python?

Python provides inbuilt functions for creating, writing, and reading files. There are two types of files that can be handled in python, normal text files and binary files

* Text files: In this type of file, Each line of text is terminated with a special character called EOL (End of Line), which is the new line character (‘\n’) in python by default.
* Binary files: In this type of file, there is no terminator for a line, and the data is stored after converting it into machine-understandable binary language.

Read to a file in Python

with open ("filename.txt", "r") as f:

content = f.read()

print(content)

Write to a file in Python

with open ("output.txt", "w") as f:

f.write("Hello, world!")

9.How do you install and use external packages in Python?

In Python, you can install and use external packages using a package manager called **pip**. pip comes bundled with most Python installation.

Example

pip install package\_name

**package\_name** is the name of the package that you want to install

Once you've installed a package using **pip**, we can use it in your Python code like this

**import package\_name.**

10.How do you use the "if" statement in Python to perform conditional execution?

The if statement to perform conditional execution based on a certain condition

x = 5

if x > 0:

print("x is positive")

We can also use the else statement to perform alternative execution if the condition is not true

x = -5

if x > 0:

print("x is positive")

else:

print("x is not positive")

We can also use the **elif** statement to check multiple conditions in a single if statement

x = 0

if x > 0:

print("x is positive")

elif x < 0:

print("x is negative")

else:

print("x is zero")