🐍 PYTHON QUESTIONS – SET 1 (1–10)

---

1. What is the Zen of Python?

The **Zen of Python** is a collection of guiding principles that describe how Python code should be written.
It focuses on **simplicity, readability, and clarity**.

You can view it by running:

```
import this
```

Some important principles are:

- Simple is better than complex
- Readability counts
- Explicit is better than implicit

These principles help developers write clean and maintainable code.

---

2. What is a string data type and how do you define a string?

A **string** is a data type used to store a sequence of characters such as letters, numbers, and symbols.
Strings in Python are **immutable** (cannot be changed after creation).

Examples:

```
name = "Python"

city = 'Bangalore'

message = """Welcome to Python"""
```

Strings can be defined using **single quotes**, **double quotes**, or **triple quotes**.

---

3. Give 3–4 string methods that you have used

String methods are built-in functions used to manipulate strings.

Examples:

```
text = " python programming "


text.upper()    # Converts string to uppercase

text.lower()    # Converts string to lowercase

text.strip()    # Removes leading and trailing spaces
```

```
text.split()    # Splits string into a list
```

---

4. What are f-strings in Python?

f-strings (formatted string literals) allow variables and expressions to be embedded directly inside strings.

Example:

```
name = "Rahul"

age = 22

print(f"My name is {name} and I am {age} years old")
```

f-strings are faster and more readable than older formatting methods.

---

5. How do you reverse a string by using a function?

A string can be reversed using slicing inside a function.

Example:

```
def reverse_string(s):
   return s[::-1]


print(reverse_string("Python"))
```

Output:

nohtyP

---

6. What is list datatype and how do you define the list datatype?

A **list** is an ordered and mutable collection of elements.
It can store different data types and allows duplicate values.

Example:

```
numbers = [1, 2, 3, 4]

items = ["apple", 10, 2.5]
```

Lists are defined using **square brackets [ ]**.

---

7. What is slicing of list?

List slicing is used to extract a portion of a list.

Syntax:

list[start : end : step]

Example:

```
nums = [10, 20, 30, 40, 50]
print(nums[1:4])
```

Output:

```
[20, 30, 40]
```

---

8. How do you modify the list datatype?

Lists are mutable, so their elements can be changed.

Example:

```
nums = [1, 2, 3]
nums[1] = 100
nums.append(4)
nums.remove(1)
print(nums)
```

---

9. What is the difference between sort() and sorted() methods in a list?

sort()	**sorted()**
Modifies original list	Returns a new list
Works only on lists	Works on any iterable
No return value	Returns sorted list

Example:

```
lst = [3, 1, 2]
lst.sort()
new_lst = sorted(lst)
```

---

10. What is the difference between del and len() methods in a list?

- del is used to **delete elements**
- len() is used to **count elements**

Example:

```
lst = [1, 2, 3]
del lst[0]
print(len(lst))
```

Output:

2

🐍 PYTHON QUESTIONS – SET 2 (11–20)

---

11. What is negative indexing?

Negative indexing allows accessing elements from the **end of a sequence**.

- -1 → last element
- -2 → second last element

Example:

```
lst = [10, 20, 30, 40]
print(lst[-1])
print(lst[-2])
```

Output:

40

30

---

12. How do you append values to a list?

The append() method is used to add an element at the end of a list.

Example:

```
numbers = [1, 2, 3]
numbers.append(4)
print(numbers)
```

---

13. What is list comprehension?

List comprehension provides a **short and readable way** to create lists using loops and conditions.

Syntax:

[expression for item in iterable if condition]

Example:

```
squares = [x*x for x in range(1, 6)]
print(squares)
```

---

14. What is tuple datatype and how do you define it?

A **tuple** is an ordered and **immutable** collection of elements.

Example:

```
tup = (10, 20, 30)
single = (5,)   # single-element tuple
```

Tuples use **parentheses ( )**.

---

15. What is the difference between tuple and list?

List	Tuple
Mutable	Immutable
Uses [ ]	Uses ( )
Slower	Faster
More memory	Less memory

---

16. What is the syntax of for loop? Write 4 examples of for loop

Syntax:

```
for variable in sequence:
    statements
```

Examples:

```
for i in range(5):
    print(i)


for ch in "Python":
    print(ch)
```

```
for num in [1, 2, 3]:
    print(num)

for i in range(2, 11, 2):
    print(i)
```

---

17. What is the syntax of while loop? Write 4 examples

Syntax:

```
while condition:
    statements
```

Examples:

```
i = 1
while i <= 5:
    print(i)
    i += 1

count = 3
while count > 0:
    print(count)
    count -= 1

num = 1
while num <= 10:
    if num % 2 == 0:
        print(num)
    num += 1

x = 5
while x != 0:
    print(x)
    x -= 1
```

---

18. What is if-else condition? Write 2 examples

if-else is used for decision making.

Example 1:

```
num = 10
if num > 0:
   print("Positive")
else:
   print("Negative")
```

Example 2:

```
age = 18
if age >= 18:
   print("Eligible to vote")
else:
   print("Not eligible")
```

---

19. What is elif? Write 2 examples

elif is used to check **multiple conditions**.

Example 1:

```
marks = 75
if marks >= 90:
   print("A Grade")
elif marks >= 60:
   print("B Grade")
else:
   print("C Grade")
```

Example 2:

```
day = 3
if day == 1:
   print("Monday")
elif day == 2:
```

```
    print("Tuesday")
elif day == 3:
    print("Wednesday")
```

---

20. What is the difference between integer and float?

Integer	Float
Whole numbers	Decimal numbers
Example: 10	Example: 10.5
More precise	Less precise

🐍 PYTHON QUESTIONS – SET 3 (21–30)

---

21. What is type casting? Write 3 examples

Type casting is the process of converting one data type into another.

Examples:

```
a = int("10")      # string to integer
b = float(5)       # integer to float
c = str(100)       # integer to string
```

---

22. What is a dictionary datatype and how do you define it?

A **dictionary** stores data in **key–value pairs**.
It is unordered, mutable, and keys must be unique.

Example:

```
student = {
    "name": "Amit",
    "age": 21,
    "marks": 85
}
```

---

23. What is the syntax of for loop for a dictionary?

A dictionary can be looped using keys, values, or both.

Example:

```
for key in student:
    print(key, student[key])


for value in student.values():
    print(value)


for key, value in student.items():
    print(key, value)
```

---

24. How do you read a list in a dictionary?

A dictionary value can be a list.

Example:

```
data = {
    "colors": ["red", "green", "blue"]
}


print(data["colors"])
print(data["colors"][0])
```

---

25. Write 5 dictionary methods

Common dictionary methods:

```
student.keys()
student.values()
student.items()
student.get("name")
student.update({"age": 22})
```

---

26. How do you modify the values in a dictionary?

You can modify values by using the key.

Example:

```
student["age"] = 23
student.update({"marks": 90})
```

---

27. What is a function and how do you define the function?

A function is a reusable block of code that performs a specific task.

Syntax:

```
def function_name():
    statements
```

Example:

```
def greet():
    print("Hello Python")
```

---

28. What are parameters and arguments?

- **Parameters**: Variables defined in function definition
- **Arguments**: Values passed to the function

Example:

```
def add(a, b):   # parameters
    return a + b

add(2, 3)      # arguments
```

---

29. What are types of arguments? Write 3 examples of each

1. Positional arguments

```
def sub(a, b):
    print(a - b)

sub(10, 5)
```

2. Keyword arguments

```
sub(b=5, a=10)
```

3. Default arguments

```
def greet(name="User"):
```

```
    print(name)


greet()
greet("Rahul")
```

4. Variable-length arguments

```
def total(*nums):
    print(sum(nums))


total(1, 2, 3)
```

---

30. Write a function to read a CSV file

```
import csv


def read_csv_file(filename):
    with open(filename, 'r') as file:
        reader = csv.reader(file)
        for row in reader:
            print(row)
```

PYTHON QUESTIONS – SET 4 (31–40)

---

31. Write a function to read a JSON file

```
import json


def read_json_file(filename):
    with open(filename, 'r') as file:
        data = json.load(file)
        print(data)
```

---

32. Write a function to handle duplicate values in a text / CSV / JSON file

Example for a text file:

```
def remove_duplicates(filename):
  with open(filename, 'r') as file:
    lines = file.readlines()

  unique_lines = set(lines)

  with open("output.txt", 'w') as file:
    file.writelines(unique_lines)
```

---

33. What is a class and how do you define a class?

A **class** is a blueprint for creating objects.

Example:

```
class Student:
  pass
```

---

34. What is an object or instance?

An **object** is an instance of a class.

Example:

```
s1 = Student()
```

---

35. What is self in a class and how do you use it?

self refers to the **current object**.

Example:

```
class Student:
  def display(self):
    print("This is a student")
```

---

36. What is a method in a class? Write 5 methods with an example

Methods are functions inside a class.

Example:

```
class Calculator:
```

```
  def add(self, a, b):
    return a + b

  def sub(self, a, b):
    return a - b

  def mul(self, a, b):
    return a * b

  def div(self, a, b):
    return a / b

  def square(self, a):
    return a * a
```

---

37. What is an attribute?

Attributes are variables that belong to a class or object.

Example:

```
class Student:
  def __init__(self, name, age):
    self.name = name    # attribute
    self.age = age
```

---

38. What are Python OOPS concepts?

The main OOPS concepts are:

- Class
- Object
- Inheritance
- Polymorphism
- Encapsulation
- Abstraction

---

39. What is OOPS?

OOPS stands for **Object-Oriented Programming System**.
It organizes code using objects and classes to improve reusability and maintainability.

---

40. Explain different OOPS concepts with an example

Inheritance

```
class A:
  def show(self):
    print("Parent")


class B(A):
  pass
```

Polymorphism

```
print(len("Python"))
print(len([1,2,3]))
```

Encapsulation

```
class Test:
  def __init__(self):
    self.__data = 10
```

Abstraction

```
from abc import ABC, abstractmethod


class Shape(ABC):
  @abstractmethod
  def area(self):
    pass
```

PYTHON QUESTIONS – FINAL SET (41–50) – UPDATED

---

41. What is the difference between polymorphism and encapsulation?

Polymorphism	Encapsulation
Same method, different behavior	Data hiding
Achieved using overloading/overriding	Achieved using private variables
Example: len()	Example: __var

---

42. What are generators? Give 2 examples

Generators return values one by one using yield.

```
def gen():
  for i in range(3):
    yield i
squares = (x*x for x in range(5))
```

---

43. What are decorators? Give 2 examples

Decorators modify function behavior.

```
def decorator(func):
  def wrapper():
    print("Before")
    func()
  return wrapper
@decorator
def show():
  print("Hello")
```

---

44. What is import io module?

The io module handles streams like files and memory buffers.

```
import io
text = io.StringIO("Python")
print(text.read())
```

---

45. Create a sample module using 5 functions and call one function

mymodule.py

```
def add(a, b): return a+b
def sub(a, b): return a-b
def mul(a, b): return a*b
def div(a, b): return a/b
def square(a): return a*a
```

main.py

```
import mymodule
print(mymodule.add(10, 5))
```

---

46. What is exception handling?

Exception handling prevents program crashes using try-except.

```
try:
    x = 10 / 0
except ZeroDivisionError:
    print("Error handled")
```

---

47. Write 3 exception handling scenarios

```
try:
    int("abc")
except ValueError:
    print("Value Error")


try:
    open("file.txt")
except FileNotFoundError:
    print("File not found")


try:
    lst = [1,2]
    lst[5]
```

```
except IndexError:
    print("Index error")
```

---

48. What is raise error exception?

raise is used to manually throw an exception.

```
age = -5
if age < 0:
    raise ValueError("Invalid age")
```

---

49. Difference between class and function?

Class	Function
Blueprint	Task
Supports OOPS	No OOPS
Has attributes	No attributes

---

50. Practice Programs (FINAL)

✓ Prime Number

```
num = 7
for i in range(2, num):
    if num % i == 0:
        print("Not Prime")
        break
else:
    print("Prime")
```

✓ Factorial

```
fact = 1
for i in range(1, 6):
    fact *= i
print(fact)
```

✓ Palindrome

```
s = "madam"
print("Palindrome" if s == s[::-1] else "Not Palindrome")
```

✓ Fibonacci

```
a, b = 0, 1
for i in range(5):
    print(a)
    a, b = b, a+b
```

✓ Armstrong Number (EXPANDED)

```
num = int(input("Enter number: "))
arm = 0
length = len(str(num))

for i in str(num):
    i = int(i)
    arm += i ** length

if arm == num:
    print("Armstrong Number")
else:
    print("Not Armstrong")
```

✓ Sum of Digits

```
num = 123
total = 0
for i in str(num):
    total += int(i)
print(total)
```

✓ Reverse String

```
s = "Python"
print(s[::-1])
```

✓ Sort Colours

```
colors = ["red", "blue", "green"]
```

```
print(sorted(colors))

print(sorted(colors, reverse=True))
```
