**1.What Is Python? What are its key features?**

A. **Python** is a high-level, interpreted, general-purpose programming language. It was created by **Guido van Rossum** and first released in **1991**.

Python emphasizes **code readability** and a clean, easy-to-understand syntax

Key Features of Python:

|  |
| --- |
| Easy to Learn |
| Interpreted |
| Cross-platform |
| Large Library Support |
| Dynamic Typing |
| Object-Oriented |
| Community Support |

**2.What are the Difference between python 2 and python 3?**

| **Feature** | **Python 2** | **Python 3** |
| --- | --- | --- |
| **Release Year** | 2000 | 2008 |
| **Print Statement** | print "Hello" (like a statement) | print("Hello") (a function) |
| **Integer Division** | 5 / 2 gives 2 (truncates) | 5 / 2 gives 2.5 (true division) |
| **Unicode Support** | Strings are ASCII by default; Unicode must be declared | All strings are Unicode by default |
| **xrange vs range** | xrange() used for large ranges (memory efficient) | range() behaves like xrange() in Python 3 |
| **Input Method** | input() evaluates input; raw\_input() for string input | input() always returns a string |
| **Library Support** | Limited – most new libraries don’t support Python 2 | Fully supported by all modern libraries |
| **End of Life** | Reached in **January 1, 2020** | Actively maintained and updated |

**3.What is indentation in python and why is it important?**

**Indentation** in Python refers to the **spaces or tabs** used at the beginning of lines to define **code blocks**.

In many programming languages, code blocks are enclosed in {} braces — but in Python, **indentation is the only way** to group code.

**Why is Indentation Important in Python?**

|  |
| --- |
| It tells Python which statements belong to loops, functions, if-else blocks, etc. |
| Makes code visually clear and logically grouped. |
| Unlike other languages, Python won’t run without proper indentation. |
| Consistent indentation makes code easy to understand and maintain. |

**4.Explain the concept of python being dynamically typed how does it affect development ?**

In Python, **"dynamically typed"** means:

**You do not need to declare the data type of a variable when you create it.**  
Python automatically figures out the data type **at runtime** (while the program is running).

How Dynamic Typing Affects Development

|  |
| --- |
| No need to declare types — quick prototyping and fewer lines of code. |
| Same variable can hold any type of data, making functions and structures more general. |
| Type-related bugs may only appear when the code runs, not before. |
| If types are accidentally changed, it can cause confusing bugs. |

**5. what is interpreter ?and what is compiler?**

Both **interpreter** and **compiler** are programs that **convert source code** (what you write) into **machine code** (what the computer can understand), but they work **differently**.

### 1. **Interpreter**

* **Translates code line-by-line at runtime.**
* Executes the code **directly** without producing a separate file.
* **Slower execution**, but easier to debug.

.

### 2. **Compiler**

* **Translates the entire source code at once** into machine code (binary).
* Produces a **separate executable file** (.exe, .out).
* **Faster execution**, but debugging is harder

**Python Data Types-15 interview Questions**

**1. What are Python’s standard built-in data types?**

- Numeric: int, float, complex

- Sequence: list, tuple, range

- Text: str

- Set: set, frozenset

- Mapping: dict

- Boolean: bool

- Binary: bytes, bytearray, memoryview

**2. What is the difference between mutable and immutable data types? Give examples.**

Mutable: Can be changed (e.g., list, dict) Immutable: Cannot be changed (e.g., str, tuple, int)

**3. Is a string mutable in Python? Prove your answer with an example.**

No, String is an immutable data type

s = "hello" s[0] = "H" # Raises TypeError

**4. How are list and tuple different? When would you prefer one over the other?**

| **Feature** | **list** | **tuple** |
| --- | --- | --- |
| **Mutability** | Mutable (can be changed) | Immutable (cannot be changed) |
| **Syntax** | [1, 2, 3] | (1, 2, 3) |
| **Methods** | Has many (e.g. append(), remove()) | Very few methods |
| **Performance** | Slower (more overhead) | Faster (less overhead) |
| **Memory Use** | Uses more memory | More memory efficient |

**5. Can you modify a tuple after creating it? If not, how can you work around it?**

No. Tuples are immutable.

You can create a new tuple:

t = (1, 2, 3)

t = t + (4,)

Another Way To Modify list is we have to cobvert to list and re-convert it into the tuple

t = (1, 2, 3)

t = list(t)

t[0] = 5

t = tuple(t)

print(t) # (5, 2, 3)

**6. What will be the output of this code? Explain.**

x = (1)

print(type(x))#<class int>

y = (1,)

print(type(y)) #<class tuple>

Comma defines a tuple.

**7. How can you convert a list into a tuple and a tuple into a list? Give examples.**

Tuple:

my\_list = [1, 2, 3, 4]

my\_tuple = tuple(my\_list)

print(my\_tuple) # Output: (1, 2, 3, 4)

List:

my\_tuple = (10, 20, 30)

my\_list = list(my\_tuple)

print(my\_list) # Output: [10, 20, 30]

**8. What is the difference between is and == in Python? Show with strings and lists**

. ==: Value equality

is: Identity

Example:

a = [1, 2]

b = [1, 2]

a == b # True

a is b # False

**9. What is the data type of the result from the input() function? Howdo youconvertitto int or float?**

input() returns str.

Use int(input()), float(input()) for conversion.

**10. What is the difference between None, 0, and False? Are they equal? Why or why not?**

None

* A special constant that means "no value" or "nothing".
* Its type is NoneType.
* Often used to show that a variable is empty or a function has no return value.

0

* A numeric value representing the number zero.
* Its type is int (integer).
* Used in mathematics, counting, or as a default numeric value.

False

* A Boolean value used in logic and conditions.
* Its type is bool.
* Represents something that is logically false.

**11. What will be the output of the following code?**

a = [1, 2, 3]

b = a

b.append(4)

print(a) # [1, 2, 3, 4]

Lists are mutable and both refer to same list

**12. Can you use a list as a dictionary key? Why or why not?**

No. Lists are mutable and unhashable, so they cannot be used as dictionary keys

**13. What are the rules for dictionary keys and values in Python?**

Keys must be immutable (e.g., int, str, tuple)

Values can be any type

**14. Write Python code to:**

**- Add item to list**

**- Change dictionary value**

**- Slice a string**

my\_list = [1, 2]

my\_list.append(3)

my\_dict = {'a': 1}

my\_dict['a'] = 100

s = "Interview"

print(s[0:5]) # Inter

**15. Given the list below, how would you remove duplicates and sort it?**

my\_list = [4, 2, 7, 2, 4, 3]

my\_list = [4, 2, 7, 2, 4, 3]

unique = []

for item in my\_list:

if item not in unique:

unique.append(item)

n = len(unique)

for i in range(n):

for j in range(0, n - i - 1):

if unique[j] > unique[j + 1]:

unique[j], unique[j + 1] = unique[j + 1], unique[j] #swap

print(unique)

**15 Tricky Python Output-Based Questions**

1.

a = "python"

print(a[0] + a[-1])

# Output: 'pn'

2.

a = [1, 2, 3]

b = a

b.append(4)

print(a)

# Output: [1, 2, 3, 4]

3.

x = (1)

print(type(x))

# Output: <class ‘int’>

4.

x = (1,)

print(type(x))

# Output:<class ‘tuple’>

5.

a = [10, 20, 30, 40, 50]

print(a[1:4])

# Output: [20, 30, 40]

a[1:4] = [99]

print(a)

# Output: [10, 99, 50]

6.

text = "Vamsi"

text[0] = 'v'

print(text)

# Error: 'str' object does not support item assignment

7.

data = [1, 2, 3]

print(data + data \* 2)

# Output: [1, 2, 3, 1, 2, 3, 1, 2, 3]

8.

x = "Python"

print(x[::3])

# Output: 'Ph' (index 0 and 3)

9.

a = [1, 2, 3]

b = a[:]

b.append(4)

print(a)

# Output: [1, 2, 3] (b is a copy)

10.

info = {'name': 'Vamsi', 'age': 22}

info['name'] = ['Vamsi', 'Kiran']

info['name'][0] = 'Ram'

print(info)

# Output: {'name': ['Ram', 'Kiran'], 'age': 22}

11.

nums = [1, 2, 3]

nums[1:2] = [10, 20, 30]

print(nums)

# Output: [1, 10, 20, 30, 3]

12.

a = ['apple', 'banana', 'mango']

b = a

a = a + ['grape']

print(b)

# Output: ['apple', 'banana', 'mango'] (a now points to a new list)

13.

val = [100]

for i in val:

val.append(i + 1)

if i > 102:

break

print(val)

# Output: [100, 101, 102, 103,104] (val grows while looping)

14.

t = (1, 2, [3, 4])

t[2][0] = 99

print(t)

# Output: (1, 2, [99, 4]) (list inside tuple is mutable)

15.

a = "abcde"

print(a[10:]) # Output: ''

print(a[-10:]) # Output: 'abcde' (Python doesn't throw error for large negative slice)