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

Python has several built-in data types categorized into:

* **Numeric**: int, float, complex
* **Sequence**: str, list, tuple, range
* **Set**: set, frozenset
* **Mapping**: dict
* **Boolean**: bool
* **Binary**: bytes, bytearray, memoryview
* **None Type**: NoneType

**2. Why is Python used extensively in Data Science?**

Python is popular in Data Science because:

* It has powerful **libraries** like Pandas, NumPy, Matplotlib, scikit-learn
* Simple and readable syntax
* Great for **data manipulation**, analysis, and machine learning
* Active community and large ecosystem

**3. Difference between lists and tuples in Python**

| **Feature** | **List** | **Tuple** |
| --- | --- | --- |
| Syntax | [1, 2, 3] | (1, 2, 3) |
| Mutability | **Mutable** (can be changed) | **Immutable** (cannot be changed) |
| Performance | Slower | Faster |
| Use Case | Dynamic data | Fixed data |

**4. What are Python’s predefined keywords and their uses?**

Python has **35+ keywords** like if, for, def, class, import, etc.  
Each has a special purpose in controlling program flow, defining structures, or exceptions.

Examples:

* if – Conditional logic
* def – Define functions

**5. How does Python handle mutability and immutability?**

* **Mutable** types (e.g., list, dict) allow in-place changes.
* **Immutable** types (e.g., str, tuple, int) do not allow changes; any modification creates a new object.

This behavior helps manage memory and program behavior.

**6. Significance of mutability in Python data structures**

Mutability affects:

* **Performance** (mutable types are slower but flexible)
* **Safety** (immutable types are safer in concurrent programming)
* **Memory usage**

For example, a list is ideal when data changes frequently, while a tuple is better for constant data.

**7. Types of operators in Python**

| **Type** | **Example** |
| --- | --- |
| Arithmetic | +, -, \*, /, %, \*\* |
| Comparison | ==, !=, >, <, >=, <= |
| Logical | and, or, not |
| Assignment | =, +=, -=, etc. |
| Bitwise | &, ` |
| Membership | in, not in |
| Identity | is, is not |

**8. How do you perform type casting in Python?**

By converting from one type to another using built-in functions:

int("5") # string to int

float(3) # int to float

str(100) # int to string

bool(0) # 0 to False

**9. Difference between implicit and explicit type casting**

| **Type** | **Description** | **Example** |
| --- | --- | --- |
| Implicit | Automatically done by Python | 3 + 2.0 → 5.0 |
| Explicit | Manually done using functions | int("42") → 42 |

**10. Significance of conditionals in Python**

They allow **decision-making** in code based on conditions:

if x > 0:

print("Positive")

elif x == 0:

print("Zero")

else:

print("Negative")

**11. How would you implement switch-case in Python?**

Python lacks a native switch, but you can use a **dictionary**:

def switch\_case(option):

switch = {

1: "One",

2: "Two",

3: "Three"

}

return switch.get(option, "Invalid")

print(switch\_case(2)) # Output: Two

**12. What are loops in Python? Difference between for and while loops**

* **for loop**: used when you know how many times to loop (e.g., over a list or range)

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

while i < 5: i += 1

**13. Use of break, continue, and pass**

* break: Exit the loop
* continue: Skip the rest of the loop body and move to the next iteration
* pass: Placeholder; does nothing

for i in range(5):

if i == 2:

continue

elif i == 4:

break

else:

pass

print(i)