Python Q&A with Detailed Explanations

---

1. What are the key features of Python as a programming language?

- **Easy to Learn and Readable** → Simple, English-like syntax.
- **Interpreted** → Runs line by line, no need to compile.
- **Dynamically Typed** → No need to declare variable types.
- **High-Level Language** → Abstracts memory management.
- **Extensive Libraries** → numpy, pandas, matplotlib, etc.
- **Portable & Cross-platform** → Works on Windows, macOS, Linux.
- **Object-Oriented + Functional** → Supports OOP and functional programming.
- **Free and Open Source** → Managed by Python Software Foundation.

---

2. How is Python interpreted and dynamically typed?

- **Interpreted**: Python code is executed **line by line** by the Python interpreter (CPython, PyPy, etc.), unlike C/C++ which must be compiled first.
- **Dynamically Typed**: Variable types are assigned at runtime:
- x = 5 # int
- x = "Hi" # str → allowed (type changes dynamically)

---

3. Explain the difference between Python 2 and Python 3.

- **Python 2**: Older, no longer supported (EOL in 2020).
- **Python 3**: Actively maintained and modern.
- Key differences:
  - print → print "hi" (Python 2) vs print("hi") (Python 3).
  - Division → 5/2 = 2 (Python 2) vs 5/2 = 2.5 (Python 3).
  - Unicode strings → Default in Python 3.
  - Libraries → New libraries support Python 3 only.

---

4. What is PEP 8 and why is it important?

- **PEP 8** = Python Enhancement Proposal 8 → Style guide for Python code.
- Importance:

- Improves **readability**.
- Maintains **consistency** across projects.
- Used in industry coding standards.
- Example: use snake_case for variables, 4 spaces for indentation.

---

5. How do you write comments in Python?

- **Single-line**:
- # This is a single-line comment
- **Multi-line** (using triple quotes, though not true comments):
- """
- This is a
- multi-line comment
- """

---

6. What are Python's built-in data types? Give examples.

- **Numeric**: int, float, complex
- **Sequence**: list, tuple, range
- **Text**: str
- **Set types**: set, frozenset
- **Mapping**: dict
- **Boolean**: bool
- **None**: NoneType
- x = 10 # int
- y = 3.14 # float
- z = [1, 2, 3] # list
- d = {"a": 1} # dict

---

7. What is the difference between mutable and immutable types? Provide examples.

- **Mutable**: Can be changed after creation.
  Examples → list, dict, set.
- **Immutable**: Cannot be changed once created.
  Examples → str, tuple, frozenset.

- l = [1, 2, 3]
- l.append(4) # ✅ mutable
-
- s = "hello"
- # s[0] = "H" ❌ error → strings are immutable

---

8. How is None different from 0 and False?

- None: Represents "nothing" or "no value".
- 0: Numeric zero (int).
- False: Boolean value.
- print(None == 0) # False
- print(None == False) # False

---

9. What is type casting? Give examples using int(), float(), and str().

- Converting from one type to another.
- x = int("10") # str → int (10)
- y = float("3.14") # str → float (3.14)
- z = str(100) # int → str ("100")

---

10. How do you check the type of a variable?

x = 5

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

---

11. What are the different types of operators in Python?

1. Arithmetic → + - * / % // **
2. Comparison → == != > < >= <=
3. Logical → and or not
4. Bitwise → & | ^ ~ << >>
5. Assignment → = += -= *=
6. Membership → in, not in
7. Identity → is, is not

---

12. Explain the difference between / and //.

- / → Floating-point division.
  5/2 = 2.5
- // → Floor division (integer part).
  5//2 = 2

---

13. How does the is operator differ from ==?

- == → Compares **values**.
- is → Compares **memory identity** (whether they are the same object).
- a = [1,2,3]
- b = [1,2,3]
- print(a == b) # True
- print(a is b) # False

---

14. What does the % operator do?

- Returns **remainder** of division.
- 10 % 3 = 1

---

15. Explain operator precedence in Python.

- Order in which operators are evaluated.
- Example order:
  () > ** > * / // % > + - > comparison > logical.
- print(2 + 3 * 4) # 14 (multiplication first)

---

16. How do you write an if-elif-else statement? Give an example.

```
x = 10

if x > 20:

   print("Greater than 20")

elif x > 5:

   print("Between 6 and 20")

else:
```

```
print("5 or less")
```

---

17. What is the difference between nested if and multiple elif conditions?

- **elif** → Used when multiple exclusive conditions exist.
- **Nested if** → An if inside another if.

---

18. Can Python have an else without if? Explain.

- No, else must follow an if or try.
- Example:
- try:
- x = 5 / 0
- except:
- print("Error")
- else:
- print("No error") # else works with try

---

19. What is the difference between for and while loops in Python?

- for → Used when **number of iterations is known**.
- while → Runs until **condition is False**.
- for i in range(5):
- print(i)
-
- while i < 5:
- print(i)
- i += 1

---

20. How does break differ from continue?

- break → Exits loop completely.
- continue → Skips current iteration, continues loop.

---

21. What is the use of the pass statement?

- Placeholder for code.
- def func():
- pass # do nothing yet

---

22. How do you use a for loop with the range() function?

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

---

23. How do you define and call a function in Python?

```
def greet(name):
  return "Hello " + name


print(greet("Aromal"))
```

---

24. What is the difference between a function with and without a return value?

- With return → gives output.
- Without return → performs action only.

---

25. Explain default arguments in Python functions.

- Function parameters with default values.
- def greet(name="Guest"):
- print("Hello", name)
-
- greet() # Hello Guest
- greet("Aromal") # Hello Aromal

****26. What is the difference between *args and kwargs?**

- *args → Variable number of positional arguments (tuple).
- **kwargs → Variable number of keyword arguments (dict).

---

27. Explain the difference between a list, tuple, and set.

- list → Mutable, ordered, allows duplicates.
- tuple → Immutable, ordered, allows duplicates.
- set → Mutable, unordered, no duplicates.

---

28. How do you add and remove elements from a list?

```
l = [1,2,3]
l.append(4)    # add
l.remove(2)    # remove value
l.pop()        # remove last
```

---

29. How do you access dictionary values?

```
d = {"a": 1, "b": 2}
print(d["a"])
print(d.get("b"))
```

---

30. How do you merge two dictionaries in Python 3.9+?

```
d1 = {"a": 1}
d2 = {"b": 2}
d3 = d1 | d2
```

---

31. How do you slice a string in Python?

```
s = "Python"
print(s[0:4])  # Pyth
print(s[::-1]) # reverse
```

---

32. What is the difference between .find() and .index()?

- .find() → Returns -1 if not found.
- .index() → Raises error if not found.

---

33. How do you remove whitespace from a string?

```
s = "  hello  "
```

```
print(s.strip())   # removes both sides
print(s.lstrip())  # left only
print(s.rstrip())  # right only
```

---

34. What is string interpolation in Python? Give examples using f-strings.

```
name = "Aromal"
age = 22
print(f"My name is {name}, I am {age} years old.")
```

---

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

```
# Write
with open("file.txt", "w") as f:
    f.write("Hello")

# Read
with open("file.txt", "r") as f:
    print(f.read())
```

---

36. What is the difference between read(), readline(), and readlines()?

- read() → Reads entire file.
- readline() → Reads one line.
- readlines() → Returns list of all lines.

---

37. Why is the with statement recommended for file handling?

- It automatically **closes the file** after use, even if errors occur.

---

38. How do you handle exceptions in Python?

```
try:
    x = 1/0
except ZeroDivisionError:
    print("Division by zero error")
```

---

39. What is the difference between try-except and try-finally?

- try-except → Handles errors.
- try-finally → Executes code no matter what (cleanup).

---

40. How do you raise a custom exception?

raise ValueError("Invalid value")

---

41. How do you import a module in Python?

import math

print(math.sqrt(16))

---

42. What is the difference between import module and from module import function?

- import math → Must call math.sqrt().
- from math import sqrt → Can call sqrt() directly.

---

43. How do you install third-party packages in Python?

pip install package_name

---

44. What is a lambda function?

- Anonymous (one-line) function.
- square = lambda x: x**2
- print(square(5)) # 25

---

45. Explain list comprehension with an example.

squares = [x**2 for x in range(5)]

---

46. What are Python's built-in functions? Give five examples.

- Examples: len(), max(), sum(), type(), range().

---

47. What is the purpose of the dir() function?

- Lists all attributes and methods of an object/module.
- print(dir(str))

---

48. How do you check Python's version from within a script?

```
import sys

print(sys.version)
```