Python Functions Interview Questions & Answers

---

1. **What is a function in Python?**
   A function is a reusable block of code that performs a specific task.

2. **How do you define a function in Python?**
   Using the `def` keyword followed by the function name and parentheses:

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

3. **How do you call a function in Python?**
   You call a function by writing its name followed by parentheses:

```
greet()
```

4. **Significance of the `def` keyword**
   `def` is used to define a function in Python.

5. **Difference between a function definition and a function call**

- Definition: Creates the function (`def greet(): ...`)
- Call: Executes the function (`greet()`)

6. **Difference between `return` and `print`**

- `return` sends a value back to the caller.
- `print` displays output to the console.

7. **Purpose of `None` keyword**
   `None` represents the absence of a value or a null value.

8. **Parameters vs Arguments**

- Parameters: Variables in the function definition.
- Arguments: Values passed to the function when called.

9. **Can we call a function before defining it?**
   No, Python will raise a `NameError`.

10. **Calling a function inside another function**
    Yes.

```
def outer():
    inner()

def inner():
    print("Inner function")

outer()
```

11. **Using function result in an expression**
    Yes.

```
def square(x):
    return x * x
result = square(4) + 2
```

12. **Types of functions in Python**

- Built-in functions
- User-defined functions

13. **Pre-defined functions**
    Examples: `len()`, `max()`, `min()`, `print()`

14. **User-defined functions**
    Defined using `def`. Example:

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

15. **Types of function arguments**

- Positional arguments
- Keyword arguments
- Default arguments
- Variable-length arguments

16. **Non-default parameters**
    Must be listed before default parameters:

```
def func(a, b=2):
    pass
```

17. **Default values for parameters**
    Assign a default value in the function definition:

```
def greet(name="Guest"):
    print("Hello", name)
```

18. **Keyword arguments**
    Pass by parameter name:

```
def greet(name):
    print("Hello", name)

greet(name="John")
```

19. **Variable-length argument lists**
    Allows passing multiple values:

- `*args` for non-keyword
- `**kwargs` for keyword

20. **Purpose of `*args` and `**kwargs`**

- `*args` allows a variable number of non-keyword arguments.
- `**kwargs` allows a variable number of keyword arguments.

21. **What is a docstring?**
    A string literal that appears right after the function definition used for documentation.

```
def greet():
    """This function greets the user."""
    print("Hello")
```

---

1. **What is a function in Python?**
   A reusable block of code to perform a task.
2. **Scope in Python functions**
   Scope defines where a variable can be accessed. Types include local and global scopes.
3. **Variable scope**

- Local: Defined inside a function.
- Global: Defined outside all functions.

5. **Scope of global variables**
   Accessible inside a function using `global` keyword if modification is needed.
6. **Scope of local variables**
   Accessible only within the function they are defined in.
7. **Convert local to global**
   Use `global` keyword:

```
def func():
    global x
    x = 10
```

8. **Use of `global` keyword**
   Allows modifying a global variable within a function.

9. **Use of `nonlocal` keyword**
   Used in nested functions to modify a variable in the parent function's scope.

10. **Recursive function call**
    A function that calls itself:

```
def factorial(n):
    if n == 1:
        return 1
    else:
        return n * factorial(n-1)
```

11. **Lambda function vs regular function**
    Lambda is anonymous and defined with `lambda` keyword:

```
square = lambda x: x * x
```

12. **Purpose of `lambda` keyword**
    To define small, anonymous functions.

13. **Purpose of `filter()`**
    Filters elements based on a function:

```
filter(lambda x: x > 10, [5, 12, 17])
```

14. **Purpose of `map()`**
    Applies a function to each element:

```
map(lambda x: x * 2, [1, 2, 3])
```

15. **Purpose of `reduce()`**
    Reduces sequence to a single value:

```
from functools import reduce
reduce(lambda x, y: x + y, [1, 2, 3])
```

16. **Find numbers > 10 using `filter()`**

```
list(filter(lambda x: x > 10, [5, 12, 17]))
```

17. **List of doubles using `map()`**

```
list(map(lambda x: x * 2, [1, 2, 3]))
```

18. **Find biggest using `reduce()`**

```
reduce(lambda x, y: x if x > y else y, [3, 5, 2])
```

19. **Find smallest using `reduce()`**

```
reduce(lambda x, y: x if x < y else y, [3, 5, 2])
```

20. **Find sum using `reduce()`**

```
reduce(lambda x, y: x + y, [1, 2, 3, 4])
```