20 interview questions and answers for Functions in Python:

1. What is a function in Python?

Answer: A function in Python is a reusable block of code that performs a specific task. It can take in inputs, perform operations on them, and return an output.

2. How do you define a function in Python?

Answer: To define a function in Python, use the `def` keyword followed by the function name, parentheses containing any parameters, and a colon. For example:

```

def my\_function(param1, param2):

# Code to perform task here

```

3. How do you call a function in Python?

Answer: To call a function in Python, simply write the function name followed by parentheses and any arguments. For example:

```

my\_function(1, 2)

```

4. Can a function return multiple values in Python?

Answer: Yes, a function in Python can return multiple values by separating them with commas. For example:

```

def my\_function():

return 1, 2, 3

a, b, c = my\_function()

```

5. What is the `\*args` parameter in a function definition?

Answer: The `\*args` parameter in a function definition allows the function to accept any number of arguments. These arguments are passed as a tuple. For example:

```

def my\_function(\*args):

for arg in args:

print(arg)

my\_function(1, 2, 3) # Output: 1 2 3

```

6. What is the `\*\*kwargs` parameter in a function definition?

Answer: The `\*\*kwargs` parameter in a function definition allows the function to accept any number of keyword arguments. These arguments are passed as a dictionary. For example:

```

def my\_function(\*\*kwargs):

for key, value in kwargs.items():

print(key, value)

my\_function(a=1, b=2, c=3) # Output: a 1 b 2 c 3

```

7. What is the difference between local and global variables in Python?

Answer: Local variables are defined inside a function and can only be accessed within that function, whereas global variables are defined outside a function and can be accessed anywhere in the code.

8. How do you declare a global variable inside a function?

Answer: To declare a global variable inside a function, use the `global` keyword followed by the variable name. For example:

```

def my\_function():

global my\_variable

my\_variable = 1

```

9. What is recursion in Python?

Answer: Recursion is a technique where a function calls itself, either directly or indirectly, to solve a problem.

10. What is a lambda function in Python?

Answer: A lambda function in Python is a small anonymous function that can take any number of arguments but can only have one expression. For example:

```

x = lambda a : a + 10

print(x(5)) # Output: 15

```

11. What is a decorator in Python?

Answer: A decorator in Python is a function that takes another function as an argument and extends the behavior of the latter function without modifying its code.

12. How do you define a decorator in Python?

Answer: To define a decorator in Python, you define a function that takes another function as an argument, wraps it with additional functionality, and returns the wrapped function. For example:

```

def my\_decorator(func):

def wrapper():

print("Before function call")

func()

print("After function call")

return wrapper

@my\_decorator

def my\_function():

print("Inside function")

```

13. What is a lambda function in Python?

Answer: A lambda function in Python is a small, anonymous function that can be defined on a single line. It is also known as an inline function or a function without a name. Lambda functions are often used as a shortcut for defining simple functions that are only used once. Here's an example:

```

add = lambda x, y: x + y

print(add(2, 3)) # Output: 5

```

In the example above, `add` is a lambda function that takes two arguments `x` and `y`, and returns their sum.

Sure, here are the answers to the remaining questions:

14. How do you define a generator in Python?

Answer: To define a generator in Python, you use the `yield` keyword instead of `return` to return a value from the function. This allows the function to return a sequence of values, which can be iterated over. For example:

```

def my\_generator():

yield 1

yield 2

yield 3

for value in my\_generator():

print(value) # Output: 1 2 3

```

15. What is the difference between `return` and `yield` in Python?

Answer: `return` in Python terminates a function and returns a value, whereas `yield` returns a value and temporarily suspends the function, allowing it to be resumed later from where it left off.

16. What is a closure in Python?

Answer: A closure in Python is a function object that has access to variables in its enclosing lexical scope, even when the function is called outside that scope.

17. How do you define a closure in Python?

Answer: To define a closure in Python, you define a nested function that references variables from its enclosing lexical scope. For example:

```

def outer\_function(x):

def inner\_function(y):

return x + y

return inner\_function

closure = outer\_function(10)

print(closure(5)) # Output: 15

```

18. What is a module in Python?

Answer: A module in Python is a file containing Python definitions and statements that can be imported and used in other Python code.

19. How do you import a module in Python?

Answer: To import a module in Python, use the `import` keyword followed by the module name. For example:

```

import my\_module

```

20. How do you alias a module in Python?

Answer: To alias a module in Python, use the `as` keyword followed by the alias name. For example:

```

import my\_module as mm

```