**Level 5: The Vault**

**Functions**

**Defining and Calling Functions**

**MCQs:**

1. Which keyword is used to define a function in Python?  
   a) function  
   b) define  
   c) def ✅  
   d) lambda
2. What is the correct way to call a function named hello()?  
   a) call hello()  
   b) hello ✅  
   c) hello() ✅  
   d) function hello()

**Fill in the Blanks:**

1. A function is a reusable block of code that performs a \_\_\_\_\_\_\_\_\_\_ task. (specific)
2. The return statement is used to return a \_\_\_\_\_\_\_\_\_\_ from a function. (value)

**True/False:**

1. A function must always have parameters. (False ❌)

**Types of Functions**

**MCQs:**

1. What is a lambda function?  
   a) A function without a name ✅  
   b) A built-in function  
   c) A function without parameters  
   d) A function that returns None
2. Which of the following is a built-in function?  
   a) custom()  
   b) print() ✅  
   c) myFunc()  
   d) define()

**Fill in the Blanks:**

1. Built-in functions are provided by \_\_\_\_\_\_\_\_\_\_. (Python’s standard library)
2. User-defined functions are created using the \_\_\_\_\_\_\_\_\_\_ keyword. (def)

**True/False:**

1. Lambda functions can contain multiple expressions. (False ❌)

**Function Parameters and Return Values**

**MCQs:**

1. What is the default value of a function parameter if not provided?  
   a) None ✅  
   b) 0  
   c) False  
   d) []
2. Which of these functions does **not** return a value?  
   a) sum()  
   b) print() ✅  
   c) max()  
   d) len()

**Fill in the Blanks:**

1. Default parameters allow functions to have \_\_\_\_\_\_\_\_\_\_ values if no arguments are passed. (predefined)
2. The \*args syntax allows a function to accept multiple \_\_\_\_\_\_\_\_\_\_ arguments. (positional)

**True/False:**

1. Keyword arguments allow passing values without considering order. (True ✅)

**Function Parameters**

**MCQs:**

1. What does the \*args parameter do in a function?  
   a) Accepts multiple keyword arguments  
   b) Accepts multiple positional arguments ✅  
   c) Makes a function return multiple values  
   d) Defines a default argument
2. What will be the output of the following function call?

def add(a, b):

return a + b

print(add(2, 3))

a) 5 ✅  
b) 6  
c) None  
d) Error

**Fill in the Blanks:**

1. The \*\*kwargs parameter allows a function to accept multiple \_\_\_\_\_\_\_\_\_\_ arguments. (keyword)
2. Positional arguments are assigned to parameters based on their \_\_\_\_\_\_\_\_\_\_. (order)

**True/False:**

1. The number of arguments passed to a function must always match the number of parameters. (False ❌ – Functions can use \*args and default parameters)

**Return Values**

**MCQs:**

1. What happens if a function does not have a return statement?  
   a) It returns None ✅  
   b) It returns 0  
   c) It raises an error  
   d) It executes indefinitely
2. What is the output of the following function?

def square(x):

return x \* x

print(square(4))

a) 8  
b) 16 ✅  
c) None  
d) Error

**Fill in the Blanks:**

1. The return statement is used to return a \_\_\_\_\_\_\_\_\_\_ from a function. (value)
2. A function can return multiple values using a \_\_\_\_\_\_\_\_\_\_. (tuple)

**True/False:**

1. A function can return more than one value. (True ✅)

**Default Arguments**

**MCQs:**

1. What is the default value in the following function when called as greet("John")?

python

CopyEdit

def greet(name, msg="Hello"):

print(msg, name)

a) Error  
b) Hello John ✅  
c) John  
d) None

1. What will be the output of this function call?

def multiply(a, b=5):

return a \* b

print(multiply(3))

a) 15 ✅  
b) 5  
c) Error  
d) None

**Fill in the Blanks:**

1. Default arguments must be placed at the \_\_\_\_\_\_\_\_\_\_ of the parameter list. (end)
2. If an argument is not provided for a default parameter, the function uses the \_\_\_\_\_\_\_\_\_\_ value. (default)

**True/False:**

1. Default values can be changed during function calls. (True ✅)

**Recursion**

**MCQs:**

1. What is recursion in Python?  
   a) A function calling itself ✅  
   b) A function calling another function  
   c) A loop inside a function  
   d) A function without parameters
2. What is a **base case** in recursion?  
   a) The first call to the function  
   b) A condition that stops recursion ✅  
   c) The last recursive call  
   d) The argument passed to the function

**Fill in the Blanks:**

1. Recursion is useful when solving problems that can be broken into \_\_\_\_\_\_\_\_\_\_ subproblems. (smaller)
2. If a recursive function does not have a base case, it results in a \_\_\_\_\_\_\_\_\_\_. (infinite loop / stack overflow)

**True/False:**

1. Recursion is always more efficient than loops. (False ❌ – Recursion can be slower and consume more memory)