PYTHON WORKBOOK – SECTION 5

Functions & Modular Programming

Programmer's Hub – by CodeWithVivek
https://www.youtube.com/@code-with-vivek

5.1 What Are Functions?

Quick Explanation

A **function** is a reusable block of code that performs a specific task.

Try This:
Why are functions important?

1. _____
2. _____
3. _____
4. _____

5.2 Defining & Calling Functions

Basic structure:

```
def function_name():
    # code
```

Example:

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

Call the function:

```
greet()
```

Try This:

Create a function named **welcome()** that prints:

"Welcome to Python Programming!"

```
# Write here:
```

Debug This:

What's wrong here?

```
def say_hello
    print("Hello!")
```

Your answer: _____

5.3 Parameters & Return Values

Explanation

Functions can take **inputs**, called parameters.

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

Function call:

```
greet("Vivek")
```

Your Turn:

Create a function that takes **two numbers** and prints their **sum**.

```
# Write code here:
```

Do You Understand?

✓ What is the difference between a **parameter** and an **argument**?

Return Values

Explanation

A function can **return** a result using return.

```
def add(a, b):
    return a + b
result = add(10, 20)
print(result)
```

Try This:

Create a function that returns the **square** of a number.

Write here:

Debug This:

```
def multiply(a, b):
   print(a * b)


x = multiply(4, 5)
print(x + 10)
```

Why does this give an error?

(Hint: What does the function return?)

Default Parameters

Example

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

Your Turn:

Write a function with a default parameter:

Write code here:

Keyword & Positional Arguments

Example

```
def intro(name, city):
    print(f"{name} is from {city}")
```

Try This:

Call the function using both positional and keyword arguments.

Write here:

Both Mixed:

```
def test(a, b, c):
    print(a, b, c)

test(10, c=30, b=20)
```

Variable-Length Arguments (*args, **kwargs)

***args**

Used for **multiple positional** arguments.

```
def add_all(*numbers):
    print(sum(numbers))
```

****kwargs**

Used for **multiple keyword** arguments.

```
def print_info(**data):
    for key, value in data.items():
        print(key, ":", value)
```

Try This:

Write a function that uses ***args** to print the **largest** number given.

```
# Write code:
```

Your Turn:

Write a function that uses ****kwargs** to print a formatted student record.

```
# Write here:
```

5.4 Lambda (Anonymous) Functions

Explanation

A **lambda function** is a small one-line function.

```
square = lambda x: x * x

print(square(5))
```

Try This:

Create a lambda that returns:

The last character of a string

```
# Write lambda here:
```

5.5 Map, Filter, Reduce

map() Function

Syntax:
map(function, iterable)

Example:
numbers = [1, 2, 3, 4, 5]

squared = list(map(lambda x: x * x, numbers))

print(squared)

Output:

[1, 4, 9, 16, 25]

Your Turn:
Create a list of numbers from 1 to 10. Use map() to multiply each number by 10. Store the result in a list. Print the result.

Write here:

filter() Function

Syntax:
filter(function, iterable)

Example:
numbers = [10, 15, 20, 25, 30]

even_numbers = list(filter(lambda x: x % 2 == 0, numbers))

print(even_numbers)

Output:

[10, 20, 30]

Your Turn:
Create a list of numbers. Use filter() to select only numbers greater than 50. Print the filtered list.

Write here

reduce() Function

Syntax:
from functools import reduce

reduce(function, iterable)

Example:
from functools import reduce

numbers = [1, 2, 3, 4, 5]

total = reduce(lambda a, b: a + b, numbers)

print(total)

Output:

15

Your Turn:
Create a list of numbers. Use reduce() to calculate the **product** of all elements. Print the result.

Write here

5. 6 Modules & Packages

Explanation

A **module** is a Python file containing functions and variables.

```
import math

print(math.sqrt(16))


from random import randint

print(randint(1, 10))
```

Your Turn:

Write code to:

1. Import the datetime module
2. Print the current date

```
# Write here:
```

Practice Problems

Problem 1

Write a function that checks if a number is **even or odd**.

Write code:

Problem 2

Write a function that returns the **factorial** of a number.

Write code:

Problem 3

Write a function that takes a list and returns a **new list** containing only **even numbers**.

Write code:

Problem 4

Write a function that counts vowels in a string.

Write code:

Section Summary

✓ Functions organize code
✓ Parameters allow input
✓ return makes a function reusable
✓ Default args simplify function calls
✓ *args / **kwargs accept flexible arguments
✓ Lambda functions are small, fast utilities
✓ Modules allow us to import ready-made functions

Mini Assignment

Create a file named **math_utils.py** and write 3 functions inside it:

1. A function to check **prime numbers**
2. A function to compute **GCD**
3. A function to compute **LCM**

Then create another file **main.py** that imports and uses these functions.