Python Lab Manual 05: Defining and Calling Functions

Objective

To practice writing fundamental Python functions using the def keyword, covering the three main types: functions with no arguments or return, functions with arguments but no return, and functions with both arguments and a return value.

---

1. Simple No Argument Functions

Functions that take no input and do not return a value are often used for simple actions like printing a header or a menu.

Task 1.1: Welcome Message Printer (No Argument, No Return)

This task creates a simple function to print a welcome header to the console.

Guidance:	**Code Line:**	**Explanation:**
Define the function using def, choosing a descriptive name, and using empty parentheses.	def print_welcome():	The function definition starts here.
Use a print statement for the welcome message inside the function block.	print("**********")	
Add a second print statement for the main text.	print(" Welcome to Python! ")	
Add a final print statement to close the header.	print("**********")	
Outside the function, call it to execute the code inside.	print_welcome()	This line executes the function.

Complete Program (for Verification):

Python

```
def print_welcome():
    print("**********")
    print("  Welcome to Python!  ")
    print("**********")
```

```
print_welcome()
```

Expected Output:

```
**********
   Welcome to Python!
**********
```

---

2. Functions with Arguments (No Return Value)

These functions take data as input (arguments) but perform an action (like printing) without sending a value back to the main program.

Task 2.1: Simple Exponent Calculator (Argument, No Return)

This task defines a function that takes two numbers, calculates the first number raised to the power of the second, and prints the result.

Guidance:	**Code Line:**	**Explanation:**
Define the function, naming the two required parameters.	def display_power(base, exponent):	base and exponent are the parameters.
Calculate the power using the ** operator.	result = base ** exponent	Calculates base raised to the power of exponent.
Print the formatted result inside the function.	print(f"{base} raised to the power of {exponent} is: {result}")	
Outside the function, call it with two integer values as arguments.	display_power(5, 3)	Calls the function with base=5 and exponent=3.
Call the function again with different values.	display_power(2, 8)	Calls the function with base=2 and exponent=8.

Complete Program (for Verification):

Python

```
def display_power(base, exponent):
    result = base ** exponent
    print(f"{base} raised to the power of {exponent} is: {result}")

display_power(5, 3)
display_power(2, 8)
```

Expected Output:

```
5 raised to the power of 3 is: 125
2 raised to the power of 8 is: 256
```

Task 2.2: Name and Age Printer (Argument, No Return)

This task defines a function to take a string and an integer, and print a personalized summary.

Guidance:	**Code Line:**	**Explanation:**
Define the function, requiring two parameters: name and age.	def print_info(name, age):	Parameters define the required input data.
Print the formatted summary message.	print(f"User: {name}, Age: {age}. Registration Complete.")	
Outside the function, prompt the user for their name and age, storing them in variables.	user_name = input("Enter your name: ")	
	user_age = int(input("Enter your age: "))	Input must be converted to an integer.
Call the function, passing the collected variables.	print_info(user_name, user_age)	The variables are passed as arguments.

Complete Program (for Verification):

Python

```
def print_info(name, age):
    print(f"\nUser: {name}, Age: {age}. Registration Complete.")

user_name = input("Enter your name: ")
user_age = int(input("Enter your age: "))

print_info(user_name, user_age)
```

Expected Output (Example Interaction):

```
Enter your name: Zaki
Enter your age: 24

User: Zaki, Age: 24. Registration Complete.
```

---

3. Functions with Arguments and Return Value

These functions take data as input, perform a calculation or operation, and then use the **return** keyword to send the calculated value back to the caller.

Task 3.1: Area of a Circle Calculator (Argument, Return)

This task defines a function that accepts the circle's radius and returns its calculated area.

Guidance:	**Code Line:**	**Explanation:**
Define the function, requiring the radius parameter.	def calculate_area(radius):	
Define the value of Pi (or import math.pi).	PI = 3.14159	Using a constant for clarity.
Calculate the area using the formula: $Area = \pi \times radius^2$.	area = PI * (radius ** 2)	
Use the return keyword to send the result back.	return area	The function's main output.
Outside the function, call it with a radius (e.g., 5).	radius_1 = 5	
Store the returned value in a new variable.	area_1 = calculate_area(radius_1)	The function result is captured here.
Print the final result, formatted to two decimal places.	print(f"The area of a circle with radius {radius_1} is: {area_1:.2f}")	

Complete Program (for Verification):

Python

```
def calculate_area(radius):
    PI = 3.14159
    area = PI * (radius ** 2)
    return area

radius_1 = 5
area_1 = calculate_area(radius_1)
print(f"The area of a circle with radius {radius_1} is: {area_1:.2f}")

# Call it again directly in the print statement
radius_2 = 12
print(f"The area of a circle with radius {radius_2} is: {calculate_area(radius_2):.2f}")
```

Expected Output:

```
The area of a circle with radius 5 is: 78.54
The area of a circle with radius 12 is: 452.39
```

Task 3.2: Full Name Formatter (Argument, Return)

This task defines a function that takes a first name and a last name and returns the combined, capitalized full name.

Guidance:	**Code Line:**	**Explanation:**
Define the function, requiring first_name and last_name.	def get_full_name(first_name, last_name):	
Combine the two strings with a space in between, converting to title case using .title().	full_name = f"{first_name.title()} {last_name.title()}"	Ensures correct capitalization (e.g., 'ali' becomes 'Ali').
Return the combined string.	return full_name	Sends the formatted string back.
Outside the function, store the unformatted names.	f_name = "usman"	
	l_name = "khan"	
Call the function and assign the returned value to a final variable.	user_full_name = get_full_name(f_name, l_name)	
Print the final formatted name.	print(f"Formatted Name: {user_full_name}")	

Complete Program (for Verification):

Python

```
def get_full_name(first_name, last_name):
    # Combine the names and use .title() to ensure proper capitalization
    full_name = f"{first_name.title()} {last_name.title()}"
    return full_name

f_name = "usman"
l_name = "khan"
user_full_name = get_full_name(f_name, l_name)
print(f"Formatted Name: {user_full_name}")

print(f"Another User: {get_full_name('sara', 'ali')}")
```

Expected Output:

```
Formatted Name: Usman Khan
Another User: Sara Ali
```