Week 3

Functions and Scope

This week introduces **Python Functions**, **Variable Scope**, and **Error Handling**. The final task involved building a **To-Do List Manager**, consolidating skills developed from Weeks 1–3.

Exercise 1: Creating and Using Functions

Task: Creating a Function

Create a function called greet_friends. The function should take a list of names as a parameter and print a greeting for each name in the list. The greeting should be "Hello " followed by the name. For example, if the list contains the names "John", "Jane" and "Jack", the output should be as follows:

Hello John! Hello Jane! Hello Jack!

Code

```
def greet_friends(names):
    for name in names:
        print(f"Hello {name}!")

friend_list = ["John", "Jane", "Jack"]
greet_friends(friend_list)
```

Output when run:

```
Hello John!
Hello Jane!
Hello Jack!
```

Task: Tax Calculation

Task Tax Calculation:

1. Define a function called calculate_tax that takes two arguments: income and tax_rate.
2. Inside the function, calculate the tax amount by multiplying income by tax_rate.
3. Return the tax amount as the result.
4. Call the calculate_tax function with an income of 50,000£ and a tax rate of 0.2 and print the calculated tax.
5. Try using different incomes and tax rates in this function.

Code

```
def calculate_tax(income, tax_rate):
    # Calculate and return the tax amount
    tax = income * tax_rate
    return tax

# call the function with an income of £50,000 and tax rate of 0.2
tax1 = calculate_tax(50000, 0.2)
print("Tax on £50,000 with 20% rate:", tax1)

# different incomes and tax rates
tax2 = calculate_tax(35000, 0.15)
print("Tax on £35,000 with 15% rate:", tax2)

tax3 = calculate_tax(80000, 0.25)
print("Tax on £80,000 with 25% rate:", tax3)

tax4 = calculate_tax(100000, 0.3)
print("Tax on £100,000 with 30% rate:", tax4)
```

output when run:

```
Tax on £50,000 with 20% rate: 10000.0
Tax on £35,000 with 15% rate: 5250.0
Tax on £80,000 with 25% rate: 20000.0
Tax on £100,000 with 30% rate: 30000.0
```

Task Compound Interest Calculator Function:

Your goal for this task is to write a function called compound_interest() that calculates the total amount of money earned by the investment every year. Here are the specifications:

- The function should have three parameters: principal, duration and interest_rate.
- The function should print the total amount of money earned by the investment every year.
- If the interest rate is smaller than 0 or larger than 1 the function should print out a message that says "Please enter a decimal number between 0 and 1" and return None.
- If the duration is less than 0 the function should print out a message that says "Please enter a positive number of years" and return None.
- The function should use a for loop to calculate the amount of money earned by the investment every year (Hint: use the range function and keep in mind that this starts at 0). The format of the output should be: "The total amount of money earned by the investment in year Y is X £" where X is the total amount of money earned by the investment and Y the year.
- The function should return the final investment value as an integer using the int() function The formula to calculate the total amount of money earned by the investment for each year is given by: total_for_the_year = principal * (1 + interest_rate) ** year

You can test whether this function works by calling it like this: compound_interest(1000, 5, 0.03) and it should return £1159 for the result in year 5.

Code

```python
# define the function with the parameters: principal, duration, and interest_rate
def compound_interest(principal, duration, interest_rate):
    # check if interest_rate is valid (between 0 and 1)
    if interest_rate < 0 or interest_rate > 1:
        print("Please enter a decimal number between 0 and 1")
        return None  # exit the function early if rate is invalid

    # check if the duration is a valid positive number
    if duration < 0:
        print("Please enter a positive number of years")
        return None  # exit the function early if duration is invalid

    # calculating compound interest each year using for loop
    for year in range(1, duration + 1):
        # formula to calculate compound interest for the year
        total_for_the_year = principal * (1 + interest_rate) ** year
        print(f"The total amount of money earned by the investment in year {year}
is {int(total_for_the_year)} £")

    # return the final amount as an integer
    return int(total_for_the_year)

# test the function with the given example
compound_interest(1000, 5, 0.03)
```

output when run:

```
The total amount of money earned by the investment in year 1 is 1030 £
The total amount of money earned by the investment in year 2 is 1060 £
The total amount of money earned by the investment in year 3 is 1092 £
The total amount of money earned by the investment in year 4 is 1125 £
The total amount of money earned by the investment in year 5 is 1159 £
```

Task:

Error Fixing

```python
pritn("Hello, World!")

# correct code
print("Hello, World!")
```

output when run:

```
Hello, World!
```

Name Error

Correct the name error by defining the missing variable to print "My favorite color is Blue."

```
print("My favorite color is", favorite_color)

# correct code
favorite_color = "Blue"

print("My favorite color is", favorite_color)
```

output when run:

```
My favorite color is Blue
```

Value Error

Fix the value error by changing the string to an integer so the sum is correctly calculated and printed.

code

```
number1 = "5"
number2 = 3
result = number1 + number2

# correct code
# Convert number1 from string to integer
number1 = "5"
number2 = 3

# Use int() to convert string to integer before addition
result = int(number1) + number2

# Print the result
print("The sum is:", result)
```

output when run:

```
The sum is: 8
```

index error

Correct the index error by accessing the second element (index 1) of the list and printing it.

```
fruits = ["apple", "banana", "cherry"]
print(fruits[3])

# correct code
fruits = ["apple", "banana", "cherry"]

# access and print the second element (index 1)
print(fruits[1])
```

output when run:

```
banana
```

indentation error

Fix the indentation error so the code correctly prints "Good morning!" when the time is before 12:00.

```
time = 11
if time < 12:
print("Good morning!")

# correct code

time = 11

if time < 12:
    print("Good morning!")
```

output when run:

```
Good morning!
```

To_DO list Manager

code

```
# initialize an empty list to store tasks
tasks = []
```

```python
# create function to add a new task to the list
def add_task():
    task = input("Enter a new task: ")
    tasks.append(task)
    print(f"'{task}' has been added to your to-do list.")

# create function to view current tasks in the list
def view_tasks():
    if not tasks:
        print("Your to-do list is empty.")
    else:
        print("\nYour current tasks:")
        for index, task in enumerate(tasks, start=1):
            print(f"{index}. {task}")

# create function to remove a task from the list
def remove_task():
    task_to_remove = input("Enter the exact task to remove: ")
    if task_to_remove in tasks:
        tasks.remove(task_to_remove)
        print(f"'{task_to_remove}' has been removed.")
    else:
        print(f"'{task_to_remove}' was not found in your to-do list.")

# main program loop to display the menu
while True:
    print("\n--- To-Do List Manager ---")
    print("1. Add a task")
    print("2. View tasks")
    print("3. Remove a task")
    print("4. Quit")

    choice = input("Enter your choice (1-4): ")

    # use conditionals to execute the appropriate function
    if choice == "1":
        add_task()
    elif choice == "2":
        view_tasks()
    elif choice == "3":
        remove_task()
    elif choice == "4":
        print("Exiting program. Goodbye!")
        break
    else:
        print("Invalid choice. Please enter 1, 2, 3, or 4.")
```

Explanation

1. The user can type in a task to add it, view all current tasks, or remove a task if it's done.

2. If you try to remove a task that doesn’t exist or type an invalid menu number, the program politely tells you what went wrong.
3. All the tasks are saved in a list called tasks, which keeps track of everything until the program is closed.
4. The menu keeps showing up in a loop so you can keep managing your tasks—until you choose option 4 to exit the program.