PT01-Python Fundamentals

AI Bootcamp
Instinct Institute

MORK Mongkul

Exercise 1: Student Profile Generator

In this exercise, you will write Python code to introduce yourself, using variables to store your information and the `print()` function to display it.

Task:

1. Define the variables: `full_name`, `background` (*your field of study*), `interest` (*your interest in AI*), and `goal` (*why joining this AI Bootcamp?*).
2. Use the `print()` function to display your information.

Expected Output:

```
###### Student Profile ######
-----------------------------
Name: Heng Varathyuttey
Background: Computer Science student from RUPP.
Interst: Natural Language Processing (NLP).
Goal: To build chatbot for any purpose.
```

Exercise 2: Invoice Automation System

You are a **System Software Developer**. A retail client has hired you to build a backend feature that automatically calculates and issues digital invoices for their customers.

The customer has purchased 3 items with specific prices. Your system needs to calculate the subtotal, the tax (10%), and the final amount due.

1. Create variables for the following three items:
   - Item 1: `"Notebook"` at `2.50`
   - Item 2: `"Laptop Bag"` at `45.00`

- Item 3: `"Mouse Pad"` at `3.99`

2. Calculate the `subtotal`, a `tax` of 10%, and the `total`.
3. Display the output in a structured "Receipt" format as shown below.

Expected Output:

```
========================================
           XYZ.SHOP.
========================================
ITEM                PRICE
----------------------------------------
Notebook            $2.5
Laptop Bag          $45.0
Mouse Pad           $3.99
----------------------------------------
Subtotal:           $51.49
Tax (10%):          $5.149
----------------------------------------
TOTAL DUE:          $56.639
========================================
```

Bonus Transform your static script into a dynamic **Point of Sale (POS)** system.

Task: Use the `input()` function to allow the user to type the item names and prices manually.
Requirements:

1. Capture **Name** (string) and **Price** (float) for each item.
2. Use *Type Casting* to convert input strings into numbers for calculation.
3. Print a finalized receipt based on the user's specific entries.

Expected Output:

```
Enter Item 1 Name: Espresso
Enter Item 1 Price: 3.50
...
TOTAL DUE: $3.85
```

Exercise 3: The Quadratic Solver

In Machine Learning, we often look for the "minimum" of a curve. Many of these curves are defined by **Quadratic Equations**: $ax^2 + bx + c = 0$. Your task is to build a solver that finds the roots (x) for any given coefficients a, b, and c.

Task: Write a program that:

1. Takes three inputs: `a`, `b`, and `c` (convert them to `float`).

2. Calculates the discriminant `D`.
3. Uses `if/elif/else` to print the number of roots and their values.
4. *Formula for roots:* $x = \frac{-b \pm \sqrt{D}}{2a}$

Expected Output:

```
Enter a: 1
Enter b: -5
Enter c: 6

Discriminant: 1.0
Two real roots found: x1 = 3.0, x2 = 2.0
```

Exercise 4:Even Number Hunter

In Data Science, we often need to filter ”noisy” data. Imagine you are building a data pre-processor that only allows **Even Numbers** to pass through to your AI model, while discarding all odd values.

Task: Given the list: `numbers = [12, 4, 3, 17, 20, 9]`

1. Use a `for` loop to iterate through each element in the list.
2. Use the **Modulo Operator** (`%`) to check if a number is even.
3. If the number is even, print it to the console.
4. If the number is odd, do nothing (skip it).

Expected Output:

```
Found even number: 12
Found even number: 4
Found even number: 20
```

Exercise 5: The Password Gatekeeper

Every secure AI system requires an authentication layer. You are tasked with creating a **Security Gate** for your local machine that prevents unauthorized access until the correct key is provided.

Task:

1. Set a variable `secret_password = "python123"`.
2. Use a `while` loop to repeatedly ask the user to `"Enter password: "`.

3. The loop must continue **as long as** the input does not match the secret password.

4. Once the user enters the correct password, the loop should terminate and display: `"Access Granted!"`.

Expected Output:

```
Enter password: hello
Enter password: ai_is_cool
Enter password: python123
Access Granted!
```

Exercise 6: The Coke Machine (Harvard CS50P)

Imagine you are programming the firmware for a vintage **Vending Machine**. The machine sells a single product: a cold bottle of Coke for exactly **50 cents**.

Logic: The machine is old and only accepts three types of coins:

- `25` cents (Quarters)
- `10` cents (Dimes)
- `5` cents (Nickels)

Task: Write a program that prompts the user to insert a coin, one at a time, until they have paid at least 50 cents.

1. Use a `while` loop to keep track of the `amount_due`.
2. Inside the loop, check if the user's input is a valid coin (`25, 10, or 5`). If it is not, ignore the input and prompt again.
3. Subtract the valid coin value from the `amount_due`.
4. Once the total reaches or exceeds 50, stop the loop and output the `change_owed` to the user.

Expected Output:

```
Amount Due: 50
Insert Coin: 25
Amount Due: 25
Insert Coin: 10
Amount Due: 15
Insert Coin: 25
Change Owed: 10
```

Exercise 7: The Runner-Up Score (HackerRank)

You are building a leaderboard for an **E-sports Tournament**. You have a list of scores from the qualifying round, and you need to find the score of the **Runner-Up** (the 2nd place finisher). Note that there might be multiple players with the same "Top Score." The Runner-Up must have a score strictly less than the maximum score.

Task: Given the list: `scores = [2, 3, 6, 6, 5]`

1. Convert the list into a **Set** to automatically remove duplicate numbers.
2. Convert it back into a **List** so you can sort it.
3. Find and print the second largest value in the unique list.

Expected Output:

```
        Original Scores: [2, 3, 6, 6, 5]
        Unique Scores: [2, 3, 5, 6]
        The Runner-Up score is: 5
```

Exercise 8: The Inventory Management System

You are the **Operations Manager** for a tech retail store. Your mission is to build a system to track stock levels, pricing, and total asset value using Python's core data structures.

The Inventory Data: Copy these data structures into your script to begin:

- `items = ["Laptop", "Printer", "Monitor", "Mouse", "Keyboard"]`
- `quantities = (10, 5, 8, 15, 12)`
- `prices = {"Laptop": 1200, "Printer": 250, "Monitor": 350, "Mouse": 30, "Keyboard": 5`

Task:

1. **Display Items:** Use a `for` loop to iterate through the `items` list and print each one.
2. **Tuple Access:** Use indexing to access and print each value from the `quantities` tuple.
3. **Dictionary Lookup:** Access the `prices` dictionary to print the price for each item.
4. **Membership Test:** Use an `if` statement and the `in` keyword to check if `"Tablet"` exists in the list.
5. **List Update:** Add `"Tablet"` to the `items` list using a list method.
6. **The Mutability Challenge:** Attempt to update the `"Laptop"` quantity to `12`. *(Note: You may encounter an error here—how do you solve it?)*

7. **Financial Logic:** Use a loop to calculate the **Total Inventory Value** ($\sum$ Quantity $\times$ Price) and print the sum.

8. **Unique Casting:** Convert the `items` list into a **Set** and print it to see the unique values.

Expected Output:

```
Checking for Tablet... Not found.
Adding Tablet... Done.
Updating Laptop Quantity... Done.

========================================
       TOTAL INVENTORY REPORT
========================================
Total Asset Value: $21,750.0
Unique items count: 6
========================================
```

Exercise 9: The Weather Station

You are developing software for an IoT Weather Station. The hardware records temperatures in **Fahrenheit**, but your AI model requires the data in **Celsius** for global standardization. You need to build a modular system to convert and analyze this data.

Task 1: The Converter Function Define a function called `to_celsius(f)` that:

1. Takes a Fahrenheit value as an input parameter.
2. Performs the calculation: $C = (F - 32) \times \frac{5}{9}$
3. **Returns** the calculated Celsius value.

Task 2: The Analysis Function Define a function called `analyze_weather(temp_list)` that:

1. Iterates through a list of Fahrenheit temperatures.
2. Calls your `to_celsius()` function inside the loop for each value.
3. Prints a message for each temperature: `"Safe"` if below 35°C, or `"Heat Warning"` if 35°C or higher.

Task 3: Execution Create a list of test data: `temps = [72, 95, 105, 60]` and pass this list as an argument to your `analyze_weather()` function.

Expected Output:

```
Temp: 22.22C - Safe
Temp: 35.00C - Heat Warning
Temp: 40.56C - Heat Warning
Temp: 15.56C - Safe
```