Python Programming Assignment

Control Flow and Functions

Swapnil Singh

Instructions

Ensure your code is well-commented and follows proper style conventions. Test your functions with multiple examples to verify correctness.

1 Control Flow Problems

Problem 1: Tax Bracket Calculator (Easy)

Write a program that calculates income tax based on the following simplified tax brackets:

- Income up to \$10,000: 0% tax
- Income from \$10,001 to \$40,000: 10% tax
- Income from \$40,001 to \$85,000: 20% tax
- Income above \$85,000: 30% tax

Use conditional statements to determine which bracket applies and calculate the total tax owed. The tax should be applied progressively (e.g., someone earning \$50,000 pays 0% on the first \$10,000, 10% on the next \$30,000, and 20% on the remaining \$10,000).

Problem 2: GDP Growth Classification (Easy)

Write a program that takes a country's annual GDP growth rate (as a percentage) and classifies it as:

- "Recession" if growth < 0
- "Slow Growth" if $0 \leq$ growth < 2
- "Moderate Growth" if $2 \leq$ growth < 4
- "Strong Growth" if growth ≥ 4

Use if-elif-else statements to print the appropriate classification.

Problem 3: Price Index Calculator (Medium)

Write a program that calculates the Consumer Price Index (CPI) for a basket of goods over 5 years. Given:

- A list of prices for Year 1 (base year): `[2.5, 1.8, 3.2, 0.9, 4.1]`
- A list of prices for Years 2-5 (create your own reasonable values)

Use a for loop to calculate the CPI for each year using the formula:

$$\text{CPI} = \frac{\sum \text{Current Year Prices}}{\sum \text{Base Year Prices}} \times 100$$

Print the CPI for each year with appropriate labels.

Problem 4: Compound Interest with Conditions (Medium)

Write a program that calculates the future value of an investment with the following conditions:

- Initial investment: \$10,000
- Base annual interest rate: 5%
- If the investment balance exceeds \$15,000, the interest rate increases to 6%
- If the balance exceeds \$20,000, the interest rate increases to 7%

Use a while loop to calculate the balance year by year until it reaches at least \$25,000. Print the year and balance for each iteration, and show how many years it takes to reach the target.

2 Function Problems

Problem 5: Elasticity Calculator (Easy)

Write a function `price_elasticity(p1, q1, p2, q2)` that calculates the price elasticity of demand using the midpoint method:

$$E_d = \frac{(Q_2 - Q_1)/[(Q_2 + Q_1)/2]}{(P_2 - P_1)/[(P_2 + P_1)/2]}$$

The function should:

- Take four parameters: initial price, initial quantity, new price, new quantity
- Return the elasticity value
- Include a docstring explaining what the function does

Test your function with: $P_1 = 10$, $Q_1 = 100$, $P_2 = 12$, $Q_2 = 80$.

Problem 6: Descriptive Statistics Function (Medium)

Write a function `describe_data(data)` that takes a list of numerical values and returns a dictionary containing:

- `‘‘mean’’`: the arithmetic mean
- `‘‘median’’`: the median value
- `‘‘min’’`: the minimum value
- `‘‘max’’`: the maximum value
- `‘‘range’’`: the range (max - min)

Do not use external libraries like NumPy. Implement the calculations using basic Python operations and list methods.

Test your function with the dataset: `[23, 45, 12, 67, 34, 89, 23, 56, 78, 45]`.

Problem 7: Present Value Calculator (Medium)

Write a function `present_value(future_cash_flows, discount_rate)` that:

- Takes a list of future cash flows and an annual discount rate
- Calculates the present value of each cash flow using: $PV = \frac{CF_t}{(1+r)^t}$
- Returns both a list of individual present values and the total NPV

Use a default parameter value of 0.05 (5%) for the discount rate.

Test with cash flows: `[1000, 1500, 2000, 2500, 3000]` over 5 years.

Problem 8: Monte Carlo Simulation Function (Hard)

Write a function `simulate_returns(initial_investment, years, simulations=1000)` that:

- Simulates investment returns over a specified number of years
- Assumes annual returns are randomly drawn from a normal distribution with mean 7% and standard deviation 15%
- Runs the specified number of simulations
- Returns a dictionary with:
  - `‘‘mean_final_value’’`: average final portfolio value across all simulations
  - `‘‘median_final_value’’`: median final portfolio value
  - `‘‘percentile_5’’`: 5th percentile (worst case in 95% of scenarios)
  - `‘‘percentile_95’’`: 95th percentile (best case in 95% of scenarios)

Hint: Use `import random` and `random.gauss(mean, std_dev)` to generate random returns.

Test with: $10,000 initial investment over 10 years with 1000 simulations.

Bonus Challenge (Optional)

Combine multiple concepts: Write a function that simulates a simple market equilibrium using an iterative process. Start with an initial price guess, calculate supply and demand at that price using linear functions, adjust the price based on excess demand/supply, and repeat until equilibrium is reached (supply $\approx$ demand within a tolerance). Your function should return the equilibrium price and quantity.