Conditionals and Loops

Learning Objectives

- Practice and reinforce knowledge of:
    - Type usage
    - Good programming practice
    - Conditionals (if/else)
    - Loops (for/while)
    - Booleans and logical operators (<, >, ==, !=)
    - Loop iteration and control (break/continue/else)

Part 0: Set-Up & Introduction

For this assignment, you will implement ideas discussed in class using conditionals and loops. You are a loan manager automating the loan approval process. You will design a point system based on applicant information such as employment, income, debts, and assets. Some starter code is provided, including both ethical and unethical examples.

Starter code includes functions for input validation, loops, and conditionals. You will:

- Implement and refine input functions for applicant data.
- Use loops and conditionals to validate and process input.
- Design a point system that fairly evaluates applicants.
- Add comments explaining your logic and ethical considerations.

Notes and Advice

- Use appropriate conditionals and logical operators for thresholds and binary choices.
- Use loops for repeated input and validation.
- Use loop control statements (break, continue, else) to manage flow.
- Print variables for debugging as needed.
- Avoid infinite loops; use clear end conditions.
- If stuck in a loop, use Ctrl+C (or Command+C on Mac) to stop execution.

Part 1- File Description

You will work with the loanApproval files:

- loanApproval_bad.py: An unethical, oversimplified system. It asks for basic info (name, age, gender, employment, address) and makes decisions based on discriminatory or irrelevant criteria. It demonstrates what not to do.
- loanApproval_mediocre.py: A binary (yes/no) system. It asks about employment, job stability, income, loans, and assets, but only as yes/no questions. It lacks nuance and does not use ranges or point systems.
- loanApproval_better.py: Uses ranges and collects more detailed info. It asks for specific amounts for income, debts, and assets, storing them in arrays. It uses loops and input validation, but still has room for improvement in precision and fairness.
- loanApproval_good.py: Implements a points-based system. Points are added or subtracted based on salary, additional income, debts, and property value, using thresholds and ranges. The approval decision is based on the total score.

You will review and modify these files, improving the criteria and logic for loan approval. The code should use variables for job stability, monthly income, previous loans, outstanding debts, and property value, with each factor affecting the approval score. The program should print the approval status and points, and include comments explaining your choices.

Part 2: Reflection

For the last part of this assignment you will write a short reflection (~ 1 page), going over your thought process working through the assignment, areas of strength and weaknesses, and 1 paragraph reflecting on the ethical considerations of your program and how the choices you made with your algorithm reflect those choices. As a programmer, try to be conscious of all the assumptions you are making about the user for your program. You are free to include potential areas of improvement for making your algorithm more encompassing or fair as well in that paragraph. While this reflection will not be graded for writing quality, please try to use at least a semi-formal writing style and tone. This is an essential part of the assignment so please don't forget to include it in your submissions.

Grading Rubric

Criteria	Proficient	Exemplary
Logic	• Passes all of the pytests • Uses while loops and for loops in appropriate areas • Uses if and elif and else in appropriate areas	• Meets all the Proficient requirements • No unnecessary/extraneous code • Removed commented out code

Style	• Passes pylint
	• Has comments
	• Variable names are relevant to the code
	• The questions and error messages being asked of the users indicate what response is being desired
	  ○ EX of a Proficient question: `How much debt do you have? :`
		• Meets all the Proficient requirements
		• Comments are descriptive and help improve legibility of the code
		• The questions and error messages being asked of the users are clear and are explicit in what sort of response they are requiring.
		  ○ Currency of the response is clarified
		  ○ The sources of the income or debt are specified within the question
		  ○ The error messages specify what the question is looking for:
		  ■ ie. Saying that a positive numerical value is needed
		  ○ EX of an Exemplary question: `Enter your cost of debt for {source} (USD):`

Reflection	• The student responded to all of the required questions • Responses are written in complete sentences • The responses meet the length requirements	• Meets all the Proficient requirements • The responses are thoughtful and are clear in their connections to the reflections being asked • There is a clear structure to the answers and they are coherent when read.