IS 430 – Foundations of Information Processing
Instructor: Kevin Trainor
Assignment: Final Project
Course Component: Final Project
Grading Rubric

Submission

Timeliness (10 available points)

Requirements
Must be submitted by date and time indicated in the weekly schedule.

Percent Credit	Description
100	On Time
50	Up to 3 days late
0	More than 3 and up to 7 days late
0	Not submitted or submitted too late

File Submitted (10 available points)

Requirements
Submit only 1 file.
File type must be .ZIP.
File name must conform to all requirements stated in assignment instructions.
Contents of .ZIP file must be a properly named directory that represents a PyCharm project.
Directory contents must be properly named PyCharm project files.
The PyCharm project that you submit should contain all files necessary to test your code. This includes copies of any starter files.

Percent Credit	Description
100	Meets all expectations.
50	Meets nearly all expectations.
0	Does not meet expectations.
0	Not submitted or submitted too late.

Exercise 1 (Regular)

Completeness (2 available content points)

Requirements
Must produce the expected quantity of results.
Must produce the exact values expected.
Results must be formatted as expected.

Percent Credit	Description
100	Meets all expectations.
90	Meets nearly all expectations.
75	Meets most expectations.
50	Meets some expectations.
25	Meets few expectations.
10	Meets nearly no expectations.
0	Meets no expectations.
0	Not submitted or submitted too late.

Technique (3 available content points)

Requirements
The Python module must be properly named.
The Python module must begin with a proper Docstring.
The top-level code in the Python program must be located in a procedure named "main".
If present, lower-level code in the Python program must be located in procedures that have properly formed descriptive names.
If present, any variables used in the Python program must have properly formed descriptive names.
The only code in the Python module that is not contained inside of a procedure must be the code that calls the procedure "main".
The code that calls that procedure "main" should appear last in the Python module.
The Python module must not contain syntax errors.
The Python module must be run-able.
The Python module must make use of all techniques that have been demonstrated in the video tutorial for this assignment.
The Python module must conform to the PEP 8 Style Guide for Python Code.
The code should not contain any "magic numbers". Instead, it should define descriptive CONSTANTS to hold standard values.
The code should use f-strings for string interpolation and number formatting.
When processing items from Python lists and tuples, the code should unpack the values into variables with meaningful variable names to avoid using less meaningful indexed expressions.
When tested, the program should be able to handle missing input. This includes empty files, blank lines in files, or the user not entering any characters at the console. The program should behave reasonably and end gracefully.
All functions that are not main() should have descriptive, action-oriented names.
All functions should be of reasonable size.
All functions should have high cohesion, and low coupling.
Use of the break statement is allowed but not encouraged.
Use of the continue statement is forbidden.
Regular expression patterns should be expressed as Python raw strings.
Your finished code must be refactored to meet all good program design practices covered in this course.
Your refactored code must be retested to demonstrate that refactoring has not altered program functionality.
Python programs that will be called from a Jupyter notebook, should not run when imported. They should only run when they have been explicitly called.
Python programs that will be called from a Jupyter notebook should not prompt the user for input. Instead, parameters should be passed explicitly to the Python program to set all configuration values.
Python programs that will be called from a Jupyter notebook should be unit tested using PyCharm. Unit testing scenarios should not be repeated in the Jupyter notebook. The Jupyter notebook code should implement the production workflow.

Jupyter notebooks should not include extensive Python code. The Python code included in the notebook should be just enough to set configuration parameters, import Python functions that have been written and tested in PyCharm, and call those functions.

Percent Credit	Description
100	Meets all expectations.
90	Meets nearly all expectations.
75	Meets most expectations.
50	Meets some expectations.
25	Meets few expectations.
10	Meets nearly no expectations.
0	Meets no expectations.
0	Not submitted or submitted too late.

Exercise 2 (Regular)

Completeness (6 available content points)

Requirements
Must produce the expected quantity of results.
Must produce the exact values expected.
Results must be formatted as expected.

Percent Credit	Description
100	Meets all expectations.
90	Meets nearly all expectations.
75	Meets most expectations.
50	Meets some expectations.
25	Meets few expectations.
10	Meets nearly no expectations.
0	Meets no expectations.
0	Not submitted or submitted too late.

Technique (7 available content points)

Requirements
The Python module must be properly named.
The Python module must begin with a proper Docstring.
The top-level code in the Python program must be located in a procedure named "main".
If present, lower-level code in the Python program must be located in procedures that have properly formed descriptive names.
If present, any variables used in the Python program must have properly formed descriptive names.
The only code in the Python module that is not contained inside of a procedure must be the code that calls the procedure "main".
The code that calls that procedure "main" should appear last in the Python module.
The Python module must not contain syntax errors.
The Python module must be run-able.
The Python module must make use of all techniques that have been demonstrated in the video tutorial for this assignment.
The Python module must conform to the PEP 8 Style Guide for Python Code.
The code should not contain any "magic numbers". Instead, it should define descriptive CONSTANTS to hold standard values.
The code should use f-strings for string interpolation and number formatting.
When processing items from Python lists and tuples, the code should unpack the values into variables with meaningful variable names to avoid using less meaningful indexed expressions.
When tested, the program should be able to handle missing input. This includes empty files, blank lines in files, or the user not entering any characters at the console. The program should behave reasonably and end gracefully.
All functions that are not main() should have descriptive, action-oriented names.
All functions should be of reasonable size.
All functions should have high cohesion, and low coupling.
Use of the break statement is allowed but not encouraged.
Use of the continue statement is forbidden.
Regular expression patterns should be expressed as Python raw strings.
Your finished code must be refactored to meet all good program design practices covered in this course.
Your refactored code must be retested to demonstrate that refactoring has not altered program functionality.
Python programs that will be called from a Jupyter notebook, should not run when imported. They should only run when they have been explicitly called.
Python programs that will be called from a Jupyter notebook should not prompt the user for input. Instead, parameters should be passed explicitly to the Python program to set all configuration values.
Python programs that will be called from a Jupyter notebook should be unit tested using PyCharm. Unit testing scenarios should not be repeated in the Jupyter notebook. The Jupyter notebook code should implement the production workflow.

Jupyter notebooks should not include extensive Python code. The Python code included in the notebook should be just enough to set configuration parameters, import Python functions that have been written and tested in PyCharm, and call those functions.

Percent Credit	Description
100	Meets all expectations.
90	Meets nearly all expectations.
75	Meets most expectations.
50	Meets some expectations.
25	Meets few expectations.
10	Meets nearly no expectations.
0	Meets no expectations.
0	Not submitted or submitted too late.

Exercise 3 (Regular)

Completeness (6 available content points)

Requirements
Must produce the expected quantity of results.
Must produce the exact values expected.
Results must be formatted as expected.

Percent Credit	Description
100	Meets all expectations.
90	Meets nearly all expectations.
75	Meets most expectations.
50	Meets some expectations.
25	Meets few expectations.
10	Meets nearly no expectations.
0	Meets no expectations.
0	Not submitted or submitted too late.

Technique (7 available content points)

Requirements
The Python module must be properly named.
The Python module must begin with a proper Docstring.
The top-level code in the Python program must be located in a procedure named "main".
If present, lower-level code in the Python program must be located in procedures that have properly formed descriptive names.
If present, any variables used in the Python program must have properly formed descriptive names.
The only code in the Python module that is not contained inside of a procedure must be the code that calls the procedure "main".
The code that calls that procedure "main" should appear last in the Python module.
The Python module must not contain syntax errors.
The Python module must be run-able.
The Python module must make use of all techniques that have been demonstrated in the video tutorial for this assignment.
The Python module must conform to the PEP 8 Style Guide for Python Code.
The code should not contain any "magic numbers". Instead, it should define descriptive CONSTANTS to hold standard values.
The code should use f-strings for string interpolation and number formatting.
When processing items from Python lists and tuples, the code should unpack the values into variables with meaningful variable names to avoid using less meaningful indexed expressions.
When tested, the program should be able to handle missing input. This includes empty files, blank lines in files, or the user not entering any characters at the console. The program should behave reasonably and end gracefully.
All functions that are not main() should have descriptive, action-oriented names.
All functions should be of reasonable size.
All functions should have high cohesion, and low coupling.
Use of the break statement is allowed but not encouraged.
Use of the continue statement is forbidden.
Regular expression patterns should be expressed as Python raw strings.
Your finished code must be refactored to meet all good program design practices covered in this course.
Your refactored code must be retested to demonstrate that refactoring has not altered program functionality.
Python programs that will be called from a Jupyter notebook, should not run when imported. They should only run when they have been explicitly called.
Python programs that will be called from a Jupyter notebook should not prompt the user for input. Instead, parameters should be passed explicitly to the Python program to set all configuration values.
Python programs that will be called from a Jupyter notebook should be unit tested using PyCharm. Unit testing scenarios should not be repeated in the Jupyter notebook. The Jupyter notebook code should implement the production workflow.

Jupyter notebooks should not include extensive Python code. The Python code included in the notebook should be just enough to set configuration parameters, import Python functions that have been written and tested in PyCharm, and call those functions.

Percent Credit	Description
100	Meets all expectations.
90	Meets nearly all expectations.
75	Meets most expectations.
50	Meets some expectations.
25	Meets few expectations.
10	Meets nearly no expectations.
0	Meets no expectations.
0	Not submitted or submitted too late.

Exercise 4 (Regular)

Completeness (2 available content points)

Requirements
Must produce the expected quantity of results.
Must produce the exact values expected.
Results must be formatted as expected.

Percent Credit	Description
100	Meets all expectations.
90	Meets nearly all expectations.
75	Meets most expectations.
50	Meets some expectations.
25	Meets few expectations.
10	Meets nearly no expectations.
0	Meets no expectations.
0	Not submitted or submitted too late.

Technique (3 available content points)

Requirements
The Python module must be properly named.
The Python module must begin with a proper Docstring.
The top-level code in the Python program must be located in a procedure named "main".
If present, lower-level code in the Python program must be located in procedures that have properly formed descriptive names.
If present, any variables used in the Python program must have properly formed descriptive names.
The only code in the Python module that is not contained inside of a procedure must be the code that calls the procedure "main".
The code that calls that procedure "main" should appear last in the Python module.
The Python module must not contain syntax errors.
The Python module must be run-able.
The Python module must make use of all techniques that have been demonstrated in the video tutorial for this assignment.
The Python module must conform to the PEP 8 Style Guide for Python Code.
The code should not contain any "magic numbers". Instead, it should define descriptive CONSTANTS to hold standard values.
The code should use f-strings for string interpolation and number formatting.
When processing items from Python lists and tuples, the code should unpack the values into variables with meaningful variable names to avoid using less meaningful indexed expressions.
When tested, the program should be able to handle missing input. This includes empty files, blank lines in files, or the user not entering any characters at the console. The program should behave reasonably and end gracefully.
All functions that are not main() should have descriptive, action-oriented names.
All functions should be of reasonable size.
All functions should have high cohesion, and low coupling.
Use of the break statement is allowed but not encouraged.
Use of the continue statement is forbidden.
Regular expression patterns should be expressed as Python raw strings.
Your finished code must be refactored to meet all good program design practices covered in this course.
Your refactored code must be retested to demonstrate that refactoring has not altered program functionality.
Python programs that will be called from a Jupyter notebook, should not run when imported. They should only run when they have been explicitly called.
Python programs that will be called from a Jupyter notebook should not prompt the user for input. Instead, parameters should be passed explicitly to the Python program to set all configuration values.
Python programs that will be called from a Jupyter notebook should be unit tested using PyCharm. Unit testing scenarios should not be repeated in the Jupyter notebook. The Jupyter notebook code should implement the production workflow.

Jupyter notebooks should not include extensive Python code. The Python code included in the notebook should be just enough to set configuration parameters, import Python functions that have been written and tested in PyCharm, and call those functions.

Percent Credit	Description
100	Meets all expectations.
90	Meets nearly all expectations.
75	Meets most expectations.
50	Meets some expectations.
25	Meets few expectations.
10	Meets nearly no expectations.
0	Meets no expectations.
0	Not submitted or submitted too late.

Exercise 5 (Regular)

Completeness (6 available content points)

Requirements
Must produce the expected quantity of results.
Must produce the exact values expected.
Results must be formatted as expected.

Percent Credit	Description
100	Meets all expectations.
90	Meets nearly all expectations.
75	Meets most expectations.
50	Meets some expectations.
25	Meets few expectations.
10	Meets nearly no expectations.
0	Meets no expectations.
0	Not submitted or submitted too late.

Technique (7 available content points)

Requirements
The Python module must be properly named.
The Python module must begin with a proper Docstring.
The top-level code in the Python program must be located in a procedure named "main".
If present, lower-level code in the Python program must be located in procedures that have properly formed descriptive names.
If present, any variables used in the Python program must have properly formed descriptive names.
The only code in the Python module that is not contained inside of a procedure must be the code that calls the procedure "main".
The code that calls that procedure "main" should appear last in the Python module.
The Python module must not contain syntax errors.
The Python module must be run-able.
The Python module must make use of all techniques that have been demonstrated in the video tutorial for this assignment.
The Python module must conform to the PEP 8 Style Guide for Python Code.
The code should not contain any "magic numbers". Instead, it should define descriptive CONSTANTS to hold standard values.
The code should use f-strings for string interpolation and number formatting.
When processing items from Python lists and tuples, the code should unpack the values into variables with meaningful variable names to avoid using less meaningful indexed expressions.
When tested, the program should be able to handle missing input. This includes empty files, blank lines in files, or the user not entering any characters at the console. The program should behave reasonably and end gracefully.
All functions that are not main() should have descriptive, action-oriented names.
All functions should be of reasonable size.
All functions should have high cohesion, and low coupling.
Use of the break statement is allowed but not encouraged.
Use of the continue statement is forbidden.
Regular expression patterns should be expressed as Python raw strings.
Your finished code must be refactored to meet all good program design practices covered in this course.
Your refactored code must be retested to demonstrate that refactoring has not altered program functionality.
Python programs that will be called from a Jupyter notebook, should not run when imported. They should only run when they have been explicitly called.
Python programs that will be called from a Jupyter notebook should not prompt the user for input. Instead, parameters should be passed explicitly to the Python program to set all configuration values.
Python programs that will be called from a Jupyter notebook should be unit tested using PyCharm. Unit testing scenarios should not be repeated in the Jupyter notebook. The Jupyter notebook code should implement the production workflow.

Jupyter notebooks should not include extensive Python code. The Python code included in the notebook should be just enough to set configuration parameters, import Python functions that have been written and tested in PyCharm, and call those functions.

Percent Credit	Description
100	Meets all expectations.
90	Meets nearly all expectations.
75	Meets most expectations.
50	Meets some expectations.
25	Meets few expectations.
10	Meets nearly no expectations.
0	Meets no expectations.
0	Not submitted or submitted too late.

Exercise 6 (Regular)

Completeness (6 available content points)

Requirements
Must produce the expected quantity of results.
Must produce the exact values expected.
Results must be formatted as expected.

Percent Credit	Description
100	Meets all expectations.
90	Meets nearly all expectations.
75	Meets most expectations.
50	Meets some expectations.
25	Meets few expectations.
10	Meets nearly no expectations.
0	Meets no expectations.
0	Not submitted or submitted too late.

Technique (7 available content points)

Requirements
The Python module must be properly named.
The Python module must begin with a proper Docstring.
The top-level code in the Python program must be located in a procedure named "main".
If present, lower-level code in the Python program must be located in procedures that have properly formed descriptive names.
If present, any variables used in the Python program must have properly formed descriptive names.
The only code in the Python module that is not contained inside of a procedure must be the code that calls the procedure "main".
The code that calls that procedure "main" should appear last in the Python module.
The Python module must not contain syntax errors.
The Python module must be run-able.
The Python module must make use of all techniques that have been demonstrated in the video tutorial for this assignment.
The Python module must conform to the PEP 8 Style Guide for Python Code.
The code should not contain any "magic numbers". Instead, it should define descriptive CONSTANTS to hold standard values.
The code should use f-strings for string interpolation and number formatting.
When processing items from Python lists and tuples, the code should unpack the values into variables with meaningful variable names to avoid using less meaningful indexed expressions.
When tested, the program should be able to handle missing input. This includes empty files, blank lines in files, or the user not entering any characters at the console. The program should behave reasonably and end gracefully.
All functions that are not main() should have descriptive, action-oriented names.
All functions should be of reasonable size.
All functions should have high cohesion, and low coupling.
Use of the break statement is allowed but not encouraged.
Use of the continue statement is forbidden.
Regular expression patterns should be expressed as Python raw strings.
Your finished code must be refactored to meet all good program design practices covered in this course.
Your refactored code must be retested to demonstrate that refactoring has not altered program functionality.
Python programs that will be called from a Jupyter notebook, should not run when imported. They should only run when they have been explicitly called.
Python programs that will be called from a Jupyter notebook should not prompt the user for input. Instead, parameters should be passed explicitly to the Python program to set all configuration values.
Python programs that will be called from a Jupyter notebook should be unit tested using PyCharm. Unit testing scenarios should not be repeated in the Jupyter notebook. The Jupyter notebook code should implement the production workflow.

Jupyter notebooks should not include extensive Python code. The Python code included in the notebook should be just enough to set configuration parameters, import Python functions that have been written and tested in PyCharm, and call those functions.

Percent Credit	Description
100	Meets all expectations.
90	Meets nearly all expectations.
75	Meets most expectations.
50	Meets some expectations.
25	Meets few expectations.
10	Meets nearly no expectations.
0	Meets no expectations.
0	Not submitted or submitted too late.

Exercise 7 (Regular)

Completeness (6 available content points)

Requirements
Must produce the expected quantity of results.
Must produce the exact values expected.
Results must be formatted as expected.

Percent Credit	Description
100	Meets all expectations.
90	Meets nearly all expectations.
75	Meets most expectations.
50	Meets some expectations.
25	Meets few expectations.
10	Meets nearly no expectations.
0	Meets no expectations.
0	Not submitted or submitted too late.

Technique (7 available content points)

Requirements
The Python module must be properly named.
The Python module must begin with a proper Docstring.
The top-level code in the Python program must be located in a procedure named "main".
If present, lower-level code in the Python program must be located in procedures that have properly formed descriptive names.
If present, any variables used in the Python program must have properly formed descriptive names.
The only code in the Python module that is not contained inside of a procedure must be the code that calls the procedure "main".
The code that calls that procedure "main" should appear last in the Python module.
The Python module must not contain syntax errors.
The Python module must be run-able.
The Python module must make use of all techniques that have been demonstrated in the video tutorial for this assignment.
The Python module must conform to the PEP 8 Style Guide for Python Code.
The code should not contain any "magic numbers". Instead, it should define descriptive CONSTANTS to hold standard values.
The code should use f-strings for string interpolation and number formatting.
When processing items from Python lists and tuples, the code should unpack the values into variables with meaningful variable names to avoid using less meaningful indexed expressions.
When tested, the program should be able to handle missing input. This includes empty files, blank lines in files, or the user not entering any characters at the console. The program should behave reasonably and end gracefully.
All functions that are not main() should have descriptive, action-oriented names.
All functions should be of reasonable size.
All functions should have high cohesion, and low coupling.
Use of the break statement is allowed but not encouraged.
Use of the continue statement is forbidden.
Regular expression patterns should be expressed as Python raw strings.
Your finished code must be refactored to meet all good program design practices covered in this course.
Your refactored code must be retested to demonstrate that refactoring has not altered program functionality.
Python programs that will be called from a Jupyter notebook, should not run when imported. They should only run when they have been explicitly called.
Python programs that will be called from a Jupyter notebook should not prompt the user for input. Instead, parameters should be passed explicitly to the Python program to set all configuration values.
Python programs that will be called from a Jupyter notebook should be unit tested using PyCharm. Unit testing scenarios should not be repeated in the Jupyter notebook. The Jupyter notebook code should implement the production workflow.

Jupyter notebooks should not include extensive Python code. The Python code included in the notebook should be just enough to set configuration parameters, import Python functions that have been written and tested in PyCharm, and call those functions.

Percent Credit	Description
100	Meets all expectations.
90	Meets nearly all expectations.
75	Meets most expectations.
50	Meets some expectations.
25	Meets few expectations.
10	Meets nearly no expectations.
0	Meets no expectations.
0	Not submitted or submitted too late.

Exercise 8 (Challenge)

Completeness (2 available content points)

Requirements
Must produce the expected quantity of results.
Must produce the exact values expected.
Results must be formatted as expected.

Percent Credit	Description
100	Meets all expectations.
90	Meets nearly all expectations.
75	Meets most expectations.
50	Meets some expectations.
25	Meets few expectations.
10	Meets nearly no expectations.
0	Meets no expectations.
0	Not submitted or submitted too late.

Technique (3 available content points)

Requirements
The Python module must be properly named.
The Python module must begin with a proper Docstring.
The top-level code in the Python program must be located in a procedure named "main".
If present, lower-level code in the Python program must be located in procedures that have properly formed descriptive names.
If present, any variables used in the Python program must have properly formed descriptive names.
The only code in the Python module that is not contained inside of a procedure must be the code that calls the procedure "main".
The code that calls that procedure "main" should appear last in the Python module.
The Python module must not contain syntax errors.
The Python module must be run-able.
The Python module must make use of all techniques that have been demonstrated in the video tutorial for this assignment.
The Python module must conform to the PEP 8 Style Guide for Python Code.
The code should not contain any "magic numbers". Instead, it should define descriptive CONSTANTS to hold standard values.
The code should use f-strings for string interpolation and number formatting.
When processing items from Python lists and tuples, the code should unpack the values into variables with meaningful variable names to avoid using less meaningful indexed expressions.
When tested, the program should be able to handle missing input. This includes empty files, blank lines in files, or the user not entering any characters at the console. The program should behave reasonably and end gracefully.
All functions that are not main() should have descriptive, action-oriented names.
All functions should be of reasonable size.
All functions should have high cohesion, and low coupling.
Use of the break statement is allowed but not encouraged.
Use of the continue statement is forbidden.
Regular expression patterns should be expressed as Python raw strings.
Your finished code must be refactored to meet all good program design practices covered in this course.
Your refactored code must be retested to demonstrate that refactoring has not altered program functionality.
Python programs that will be called from a Jupyter notebook, should not run when imported. They should only run when they have been explicitly called.
Python programs that will be called from a Jupyter notebook should not prompt the user for input. Instead, parameters should be passed explicitly to the Python program to set all configuration values.
Python programs that will be called from a Jupyter notebook should be unit tested using PyCharm. Unit testing scenarios should not be repeated in the Jupyter notebook. The Jupyter notebook code should implement the production workflow.

Jupyter notebooks should not include extensive Python code. The Python code included in the notebook should be just enough to set configuration parameters, import Python functions that have been written and tested in PyCharm, and call those functions.

Percent Credit	Description
100	Meets all expectations.
90	Meets nearly all expectations.
75	Meets most expectations.
50	Meets some expectations.
25	Meets few expectations.
10	Meets nearly no expectations.
0	Meets no expectations.
0	Not submitted or submitted too late.

Total Available Points = 100