# IS 452 – Foundations of Information Processing
**Semester: Summer 2019**
**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 |
| 0 | 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 (lastName_firstName_assignmentName.zip). |
| Contents of .ZIP file must be a properly named directory that represents a PyCharm project. |
| Directory contents must be properly named PyCharm project 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

**Completeness (10 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 (10 available content points)**

| Requirements |
|---|
| The python module must be properly named. |
| Include a single-line comment with name of program file. |
| Include a single-line comment that describes the intent of the program. |
| Place your highest-level code in a function named main. |
| Your code should be factored such that there is a function in your program for each part of the problem. |
| Each function should contain code relating to the same thing – it should have high cohesion. |
| Functions should know as little as possible about the workings of other functions – they should have low coupling. |
| If the Python file that you are creating is a regular executable program, include a final line of code in the program that calls the main() function. |
| Follow all PEP-8 Python coding style guidelines enforced by the PyCharm Editor. For example, place two blank lines between the code making up a function (or class) and the code that surrounds that function (or class). |
| Output printed by the program (both prompts and results) should be polite and descriptive. |
| Choose names for your variables that are properly descriptive. |
| Choose names for your functions that are properly descriptive. |
| Choose names for your classes that are properly descriptive |
| Follow PEP-8 Python coding style guidelines for forming names of variables, functions, and classes. |
| Close all files before the conclusion of the program. |
| Model your solution after the code that I demonstrate in the tutorial videos. |
| Remember to test your program thoroughly before submitting your work. |
| Your code must pass all relevant test cases. Make sure that it passes tests at the boundaries created by if, else, and elif conditions in your program (boundary value tests). |

| 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

**Completeness (10 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 (10 available content points)**

| Requirements |
|---|
| The python module must be properly named. |
| Include a single-line comment with name of program file. |
| Include a single-line comment that describes the intent of the program. |
| Place your highest-level code in a function named main. |
| Your code should be factored such that there is a function in your program for each part of the problem. |
| Each function should contain code relating to the same thing – it should have high cohesion. |
| Functions should know as little as possible about the workings of other functions – they should have low coupling. |
| If the Python file that you are creating is a regular executable program, include a final line of code in the program that calls the main() function. |
| Follow all PEP-8 Python coding style guidelines enforced by the PyCharm Editor. For example, place two blank lines between the code making up a function (or class) and the code that surrounds that function (or class). |
| Output printed by the program (both prompts and results) should be polite and descriptive. |
| Choose names for your variables that are properly descriptive. |
| Choose names for your functions that are properly descriptive. |
| Choose names for your classes that are properly descriptive |
| Follow PEP-8 Python coding style guidelines for forming names of variables, functions, and classes. |
| Close all files before the conclusion of the program. |
| Model your solution after the code that I demonstrate in the tutorial videos. |
| Remember to test your program thoroughly before submitting your work. |
| Your code must pass all relevant test cases. Make sure that it passes tests at the boundaries created by if, else, and elif conditions in your program (boundary value tests). |

| 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

**Completeness (10 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 (10 available content points)**

| Requirements |
|---|
| The python module must be properly named. |
| Include a single-line comment with name of program file. |
| Include a single-line comment that describes the intent of the program. |
| Place your highest-level code in a function named main. |
| Your code should be factored such that there is a function in your program for each part of the problem. |
| Each function should contain code relating to the same thing – it should have high cohesion. |
| Functions should know as little as possible about the workings of other functions – they should have low coupling. |
| Follow all PEP-8 Python coding style guidelines enforced by the PyCharm Editor. For example, place two blank lines between the code making up a function (or class) and the code that surrounds that function (or class). |
| Output printed by the program (both prompts and results) should be polite and descriptive. |
| Choose names for your variables that are properly descriptive. |
| Choose names for your functions that are properly descriptive. |
| Choose names for your classes that are properly descriptive |
| Follow PEP-8 Python coding style guidelines for forming names of variables, functions, and classes. |
| Close all files before the conclusion of the program. |
| Model your solution after the code that I demonstrate in the tutorial videos. |
| Remember to test your program thoroughly before submitting your work. |
| Your code must pass all relevant test cases. Make sure that it passes tests at the boundaries created by if, else, and elif conditions in your program (boundary value tests). |
| Place your unit testing code for the class in the main() function. |
| Include statements at the end of your module file that cause the main() function to be called only when the module is run directly. |
| Make sure that the code in main() is not called when the module is imported into another program. |
| Make sure that all client code can access instance variables using Pythonic field access (instance.fieldname). |
| When typical getter/setter features are needed for an instance variable, implement Pythonic getter/setter features using the @property decorators. DO NOT create Non-Pythonic getter/setter methods with names like get_fieldname() and set_fieldname(). |
| Never store values as instance variables that may be derived from other instance variables. Instead, provide methods in the class with names like calculate_derived_value(). |
| Always provide an __init__() constructor. |

Always provide a __str__() method.

Always provide a __repr__() method.

| 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

**Completeness (10 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 (10 available content points)**

| Requirements |
| --- |
| The python module must be properly named. |
| Include a single-line comment with name of program file. |
| Include a single-line comment that describes the intent of the program. |
| Place your highest-level code in a function named main. |
| Your code should be factored such that there is a function in your program for each part of the problem. |
| Each function should contain code relating to the same thing – it should have high cohesion. |
| Functions should know as little as possible about the workings of other functions – they should have low coupling. |
| If the Python file that you are creating is a regular executable program, include a final line of code in the program that calls the main() function. |
| Follow all PEP-8 Python coding style guidelines enforced by the PyCharm Editor. For example, place two blank lines between the code making up a function (or class) and the code that surrounds that function (or class). |
| Output printed by the program (both prompts and results) should be polite and descriptive. |
| Choose names for your variables that are properly descriptive. |
| Choose names for your functions that are properly descriptive. |
| Choose names for your classes that are properly descriptive |
| Follow PEP-8 Python coding style guidelines for forming names of variables, functions, and classes. |
| Close all files before the conclusion of the program. |
| Model your solution after the code that I demonstrate in the tutorial videos. |
| Remember to test your program thoroughly before submitting your work. |
| Your code must pass all relevant test cases. Make sure that it passes tests at the boundaries created by if, else, and elif conditions in your program (boundary value tests). |

| 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**