List of Programming Standards for CIS115 and CIS130

Table of Contents

Section I. Program name

The name of the program should follow these standards:

- FIlenames must be valid programmer defined symbols
- Program name must start with 2 alphabets, consisting of the first character of your first name, followed by the first character of your last name.
  - Do **NOT** name a program **assignment1.py** (or **assignment1.cpp)**. It is not acceptable. Instead, name the program using 1 of these conventions:
    - **mt_assignment1.py** (or **mt_assignment1.cpp**, depending upon the language), where **mt** stands for the first characters of your first and last name. For example, my name is Maya Tolappa, and I will prefix my programs with **mt.**
    - **maya_tolappa_assignment1.py** (or **maya_tolappa_assignment1.cpp**, depending upon the language), since my name is Maya Tolappa
- Do not use embedded blanks in a program name.
  - For example, **mt assignment 1.py** (or **mt assignment 1.cpp**) is not an acceptable program name. Instead, name the program **mt_assignment_1.py** (or **mt_assignment_1.cpp**)
- The program name should be self-documenting.
  - For example, module1.py (or source.cpp) is not an acceptable program name. Instead, name the program **mt_assignment1.py** (or **mt_assignment1.cpp**)

Section II. Comments

Programs should contains 4-5 lines of comments at the start of the program. Comments should contains:

- Name of the program
- Name of the programmer
- 3-4 lines explaining the purpose of writing the program

Examples are shown below. Line numbers are for your benefit and are not part of the program.

Sample comment block in a Python Script	Sample comment block in a C++ program

`# Script Name : mt_lab1.py`
`# Author : Maya Tolappa`
`# Purpose: The purpose of this lab is`
`# to demonstrate the process`
`# of creating a Python`
`# script and executing it.`
	`/*`
	`Program Name : mt_lab1.cpp`
	`Author : Maya Tolappa`
	`Purpose: The purpose of this lab is to`
	`demonstrate the process of`
	`creating a c++ program and`
	`executing it.`
	`*/`

Section III. Variable Names

Do NOT use global variables! Variables contain values. Variable names should follow these standards:

- Variable names must start with a lower case alphabet.
- Variable names in the program should always be **nouns**, and not **verbs.**
  - For example, **numberOfClasses** is a valid variable name, **displayName** is not, since it sounds like an action, and not a container.
- Variable names must be at least **5** characters long.
  - For example, naming a variable **dist** does not meet my programming standard, since it is only 4 characters long. Instead name it **dist_from_home**.
- Variable names must be self-documenting. This means that the name of the variable should provide an indication of what it is being used for in a program.
  - For example, naming a variable **_name** does not meet my programming standard. Instead, use something like **student_name** or **product_name**
- 1-character variable names are only permitted when used as a subscript variable to access an element in a list (in Python) or in an array (in C++).

Section IV. Loop Structures

Do **not** write infinite loops with break statements. These types of loops are shown below.

Invalid loop structure in Python	Invalid loop structure in C++

`num = 1`
 	
`while True:`
`  print(num)`
`  num += 1`
`  if num == 10:`
`  break`
	`int lcv = 0;`
	
	`while (true)`
	`{`
	`  cout << lcv << endl;`
	`  lcv ++;`
	`  if (lcv == 4)`
	`  break;`
	`}`

Instead, write out while statements with proper conditions.

Section V. Function Names and format

Functions perform actions. Function names should follow these standards:

- The function definition for the **main** function should be before all other function definitions in the program.
- Function names must start with a lower case alphabet.
- Function names in the program should always be **verbs**, and not **nouns.**
  - For example, **displayName** is a valid function name, **numberOfClasses** is not, since it sounds like a container/variable.
- Function names must be at least **5** characters long.
  - For example, naming a function **disp** does not meet my programming standard, since it is only 4 characters long. Instead name it something like **disp_student_info**.
- Function names must be self-descriptive or self-documenting. This means that the name of the function should provide an indication of what it is being used for in a program.
  - For example, naming a variable **display** does not meet programming standard. Instead, use **display_student_name** or **display_instuct_name**
- **Functions should not contain multiple return statements.** Use decision structures to set up the value in a variable and return that variable from the function instead of using return statements in a decision structure. (*This standard does not apply to recursive functions)*

Section VI. Object Oriented Programming

Object names must be nouns, since they hold information about a business entity. Accessor and Mutator functions in a class must follow the following standards:

- Mutator functions must be written as follows:
  - Mutator functions do not return a value
  - Mutator functions must start with the word **set.**
  - Mutator functions are invoked with 1 argument
- Accessor functions must be written as follows:
  - Accessor functions always return a value
  - Accessor functions must start with the word **get.**
  - Accessor functions are invoked without any arguments

Additional Specifications for C++ programs

- Do not use **goto** statements.
- Statements inside a block should be indented.
- Statements that execute as part of an if-block, should be on separate lines.

Examples are shown below:

<table>
<tr><th>Improper Style</th><th>Proper Style</th></tr>
<tr><td>

```
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6  int num1 = 12;
7  if (num1 < 12)
8      cout << "Hello\n";
9  else
10    cout << "Bye!\n";
11
12 return 0;
13 }
```

Statements inside a block (inside the { and })) should be indented

</td><td>

```
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6      int num1 = 12;
7      if (num1 < 12)
8          cout << "Hello\n";
9      else
10         cout << "Bye!\n";
11     return 0;
12 }
```

</td></tr>
<tr><td>

```
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6      int num1 = 12;
7      if (num1 < 12) cout << "Hello\n";
8      else cout << "Bye!\n";
9
10     return 0;
12 }
```

Statements that execute as part of the `if` should not be on the same line

</td><td>

```
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6      int num1 = 12;
7      if (num1 < 12)
8          cout << "Hello\n";
9      else
10         cout << "Bye!\n";
11     return 0;
12 }
13
```

</td></tr>
</table>

Functions in C++

Programs that contain functions must follow these guidelines:

- The int main function must be the first function definition the the program
- All other functions must be defined after int main.
- Place function prototypes for all functions (except int main) before the int main function definition.

Objects in C++

Assignments that contain classes must be coded as follows:

- Classes must be implemented in 2 files.
  - The class .h file must contain the class definition and function prototypes.
  - The class .cpp file must contain the implementations of the functions defined in the class .h file.
- Code a separate cpp file to test the class.