Java

- **Keyword**
  - final
- **Variables**
  - private static final double PI = 3.14159;
  - These variable names, by naming conventions, should be in all caps. This variable cannot be changed anywhere in the code. If you try you will be met with a compile error.
  - You can have final objects. The contents of the object can change, but the reference to the object cannot.
- **Classes**
  - public final class MyClass{}
  - A class created this way cannot be subclassed.
- **Methods**
  - public final method(){}
  - Final methods cannot be overridden.

C++

- **Keyword**
  - const
- **Variables**
  - const int pi = 3.14159;
  - This variable cannot be changed anywhere in the code. If you try you will be met with a compile error.
- **Pointers**
  - const int* x = &p
    - This is simply a pointer that points to a const int.
  - int* const x = &p
    - Here, x is a constant pointer. We cannot change the pointer, but we can change the value that it points to.
- **Function Arguments**
  - void method(const int x){}
  - The variable x then cannot be changed in the method.
- **Functions**
  - void method(cosnt int x) const {}
  - Here, the const means that the method will not change the object, and that all member variables will be treated as if they were also const.
- **Objects**
  - const object obj;
  - The data members of the object cannot be changed

C#

- Keyword
  - const
- Variable
  - const int pi = 3.14159;
  - This variable cannot be changed anywhere in the code. If you try you will be met with a compile error. This variable must be declared when instantiated.

Python

- Python does not have any way to declare constants within the language that enforces them in any way.
- However, the python style guide located here: https://www.python.org/dev/peps/pep-0008/#constants, has guidelines for the naming conventions of constants. They are usually in all caps with an underscore separating the words. The variables should then not be changed by the user. Again though, there is nothing stopping them from doing so.
- One possible work around for this is instead to declare a property that has a getter function but no setter functions. In this way, the identifier can be accessed, but not changed.
- Another is to declare a function that returns a value, and use that in place of a constant.