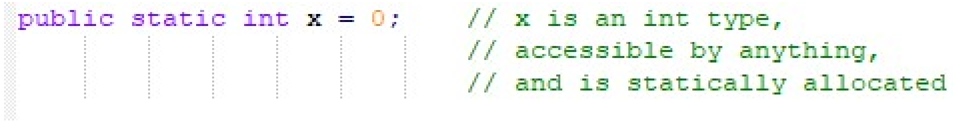
Drew Pulliam – DTP180003

Assignment 3

CS 4337.0U2

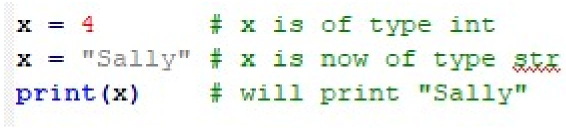
1 Variable characteristics

Variables in C++ have two main characteristics, type, and storage class. A variable's type specifies what object it contains (such as int, float, char, etc.). A variable's storage class specifies the variables lifetime and scope. These two characteristics combine to allow the programmer complete control of what the variable contains, as well as when and how it can be accessed by the program.



```
public static int x = 0;    // x is an int type,
                            // accessible by anything,
                            // and is statically allocated
```

Variables in Python are more flexible than C++. You do not have to declare a type when declaring a variable. Instead of saying "int x = 0" you can simply say "x = 0" and python will automatically make it an int type. Variables can even change types. "x = 0" and then "x = "Hello" "are valid statements. The type of x will simply change from int to string.

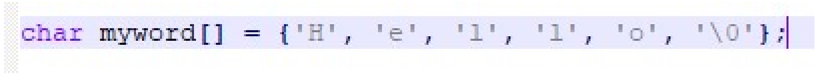


```
x = 4       # x is of type int
x = "Sally" # x is now of type str
print(x)    # will print "Sally"
```

---

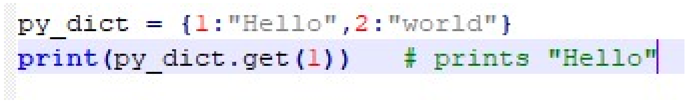
2 data types

There are 5 main data types in C++. They are bool, char, int, float, and double. Bool is the simplest and only stores true or false. Char is more complex and stores characters. Char arrays are used to hold strings, as C++ does not have a dedicated string object like other programming languages.



```
char myword[] = {'H', 'e', 'l', 'l', 'o', '\0'};
```

While python variables do not have to specify in their declaration what type they are, they still have types based on what data they contain. There are also more types in python than C++. The basic types are the same, integer, float, and Boolean. The biggest two different types are strings and dictionaries. Python Strings are generally easier to work with than C++ char arrays, and dictionaries do not exist in C++ at all. They are used to store key value pairs.

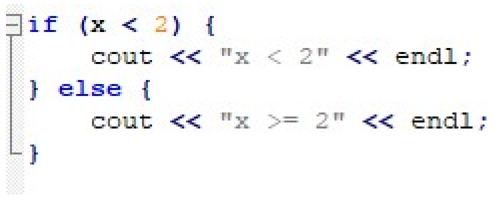


```
py_dict = {1:"Hello",2:"world"}
print(py_dict.get(1))   # prints "Hello"
```

---

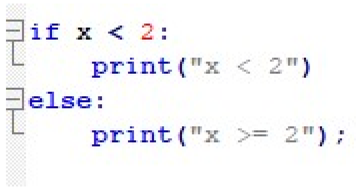
3 control statements

Control statements are the most important part of programs as they help programs do different things based on what input is given. C++ uses several different types of control statements. The most important ones are if/else blocks and switch blocks. Technically anything that can be accomplished with an if/else block can also be accomplished with a switch, and vice versa. However it is best practice to use if/else blocks for decisions that only have a small number of possible outcomes, and switch blocks for decisions with many possible outcomes.



```
if (x < 2) {
    cout << "x < 2" << endl;
} else {
    cout << "x >= 2" << endl;
}
```

Python uses the same concepts for control statements, just different syntax for everything. There are no important logical differences.

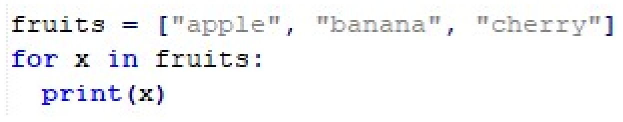


```
if x < 2:
    print("x < 2")
else:
    print("x >= 2");
```

---

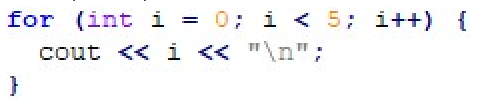
4 iterative statements

C++ and Python both have similar iterative statements, for loops, and do/while loops. Any iterative process can be accomplished by either method, but some problems are easier with one or the other. Python also allows “for each” loops which will automatically loop over each element in an array until the array is finished.



```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
  print(x)
```

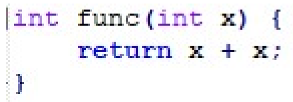
C++ require a bit more effort with for loops where you need to specify where the loop starts, ends, and how much to increment each time [2].



```
for (int i = 0; i < 5; i++) {
  cout << i << "\n";
}
```

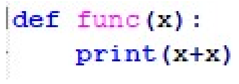
5 subprograms & their implementation

C++ subprograms are called functions. They all have some common attributes, they all have a return type, a function name, argument list, and a function body. The return type specifies what type of object the function will return upon completion, or none at all. The argument list is used to specify how many and of what type the function will receive arguments. The function body is where the function actually does something.



```
int func(int x) {
    return x + x;
}
```

Python functions are similar to variables in that they do not need to specify a return type. They are still similar to C++ in that they have a function name, argument list and body [4].



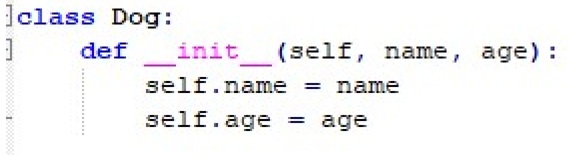
```
def func(x):
    print(x+x)
```

6 data abstraction

Data abstraction is essentially the process of hiding the complex part of the code. In C++ the code is fairly complex, although you can use create your own functions and do data abstraction yourself. Things like char arrays will always be a fairly complex data type to deal with in C++. This is almost the direct opposite of Python, Python is designed to be an incredibly simple language that makes a lot of tasks easy on the programmer, with extremely readable syntax. Python does not even use brackets like a lot of other languages, it simply uses white space.

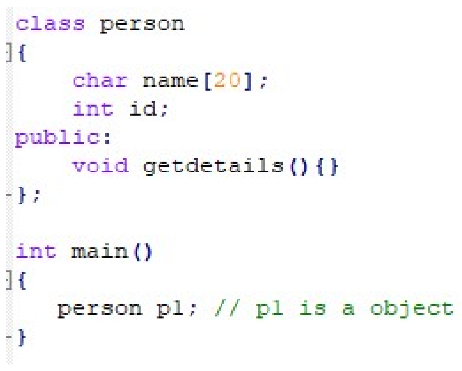
7 Support for OOP

Python allows extensive support for Object Oriented Programming (OOP). The most important part of OOP is the ability to create and use new classes and objects. Python can very easily create new classes. Python also has extensive libraries with built in objects, the most basic difference from C++ is the string data type, compared to C++ char arrays.



```
class Dog:
    def __init__(self, name, age):
        self.name = name
        self.age = age
```

C++ also has extensive support for OOP and user created classes. There are slightly less built in classes included with C++, but you can always code your own classes to do the same functions as Python classes [3].

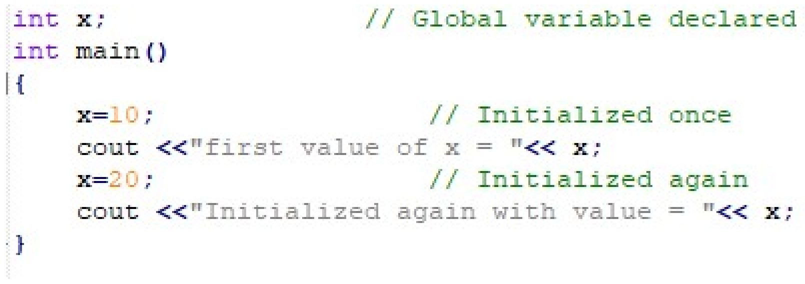


```cpp
class person
{
    char name[20];
    int id;
public:
    void getdetails(){}
};

int main()
{
   person p1; // p1 is a object
}
```

---

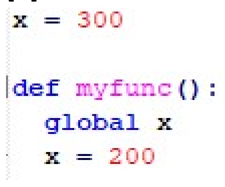
8 Scope

In C+++ variable scope is fairly simple, on a basic level, there are global variables and local variables. Global variables are declared outside of any functions, and they can be accessed by any function inside of the C++ code. Local variables are declared inside of functions and can only be accessed inside those particular functions. If the function needs to send any data to other functions it needs to be sent through the function's return statement. Alternatively, it is possible to modify global variables inside functions.



```cpp
int x;                 // Global variable declared
int main()
{
    x=10;                  // Initialized once
    cout <<"first value of x = "<< x;
    x=20;                  // Initialized again
    cout <<"Initialized again with value = "<< x;
}
```

Python scope is very similar to C++ except that functions must specify when they want to use a variable that is not local. Global variables are specified with the "global" keyword, and variables that are local to a different function are specified with "nonlocal". This allows Python to have greater control over what variables are being used inside which functions [5].



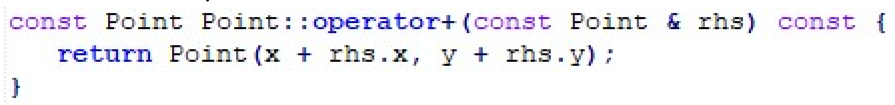
```python
x = 300

def myfunc():
  global x
  x = 200
```

9 Lifetime

Object lifetime is different between Python and C++. Python has a built-in garbage collection system. This means that any objects that are no longer referenced by the program are automatically deallocated and destroyed. C++ does not have garbage collection, and this means that the program must deallocate variables itself after it finishes using them [1]. Typically, this is a fairly simple process, but if the programmer forgets to do it can cause memory leaks.

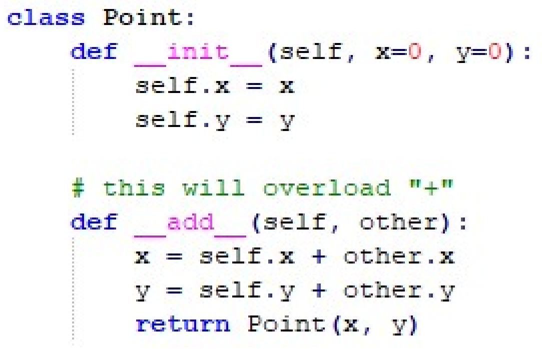
10 overloaded operators

C++ allows overloaded operators on most operators. The simplest example would be the "==" operator. The basic "==" operator checks if two objects are identical (5 == 5, etc.). This can sometimes be a problem with user created objects. If you create two objects and initialize them the same way, "==" will still return false, and this is because they point to two different memory locations. You can overload "==" so that it will return true in this case.



```cpp
const Point Point::operator+(const Point & rhs) const {
   return Point(x + rhs.x, y + rhs.y);
}
```

Python also allows for operator overloading, in the same way as C++, with just a different syntax.



```python
class Point:
    def __init__(self, x=0, y=0):
        self.x = x
        self.y = y

    # this will overload "+"
    def __add__(self, other):
        x = self.x + other.x
        y = self.y + other.y
        return Point(x, y)
```

References

1. A. Mertz, A. Bigagli, and D. Haim, “C++ Object Lifetimes,” Simplify C++!, 03-Jul-2018. [Online]. Available: https://arne-mertz.de/2016/01/c-object-lifetimes/. [Accessed: 27-Jul-2020].

2. “C++ For Loop.” [Online]. Available: https://www.w3schools.com/cpp/cpp_for_loop.asp. [Accessed: 27-Jul-2020].

3. “C++ Programming Language,” Object-oriented Programming (OOP) in C++, 2013. [Online]. Available: https://www3.ntu.edu.sg/home/ehchua/programming/cpp/cp7_OperatorOverloading.html. [Accessed: 27-Jul-2020].

4. K. Rungta, “What is a function in Python?,” JAXenter, 02-Sep-2015. [Online]. Available: https://jaxenter.com/what-is-a-function-in-python-120235.html. [Accessed: 27-Jul-2020].

5. Python Scope. [Online]. Available: https://www.w3schools.com/python/python_scope.asp. [Accessed: 27-Jul-2020].