CST0006 – Computer Programming Foundations

INTRODUCTION to WEEK04

In this lecture and lab section we are going to learn about programming paradigms and syntax. Since this is a computer programming foundations course, we won't be focusing on any one language, but instead, generalizing programming concepts and breaking apart the similarities so that we understand what programming really is. However, because we still need to practise each new concept we still need to use a real programming languages, to run the examples and see results, so we will use C and Python to see the differences between two ways to present code.

Programming Paradigms

A programming paradigm is a style or approach to how you want to develop a program. Each programming paradigm has its own rules and functionality that allow the programmer to build software in different or better ways depending on the style or preferences of each programmer. You can think of a paradigm as the little world you wish to develop in.

There are many different types of programming paradigms, some are an evolution of other paradigms that add additional functionality and some paradigms offer a totally different approach. What's important to remember here is that each programmer decides which paradigm best suits themselves, and adopts that design for their project.

The biggest confusion with programming paradigms is that some paradigms overlap each other and that some programming languages can support more than one development paradigm. This blurs the lines that define each paradigm.

Imperative Programming	Programming with an explicit sequence of commands statements that changes state. Uses GOTO jumps.
Structured Programming	Imperative Programming with cleaner blocks and nested control structures (Loops). Removes the use of GOTO.
Procedural Programming	Structured programming with procedure calls. (functions)
Object-Oriented Programming	Procedural programming with user-defined data types called objects that send messages to each other.

Syntax

Syntax refers to the grammatical structure of the language.

It is important to know the difference between syntax rules and coding standards. Syntax rules are a part of the language and must be obeyed. Coding standards is something that can change and relies heavily on personal preferences. It's due to these personal preferences that coding convention is a topic of contention and I am positive at least one person has died over disagreements about this.

Code Blocks

Code blocks are one or more lines of code that are grouped together. You as the programmer will decide how to group these lines of code together, based on your algorithm to solve each problem.

Code blocks look different for each language. Some languages use curly braces as a block delimiter and others will use indentation.

Curly brace placement is an important consideration. This again is something that can divide an entire nation. So choose carefully or risk losing friends over it.

Curly brace placement example 1

```
#include <stdio.h>

int main() {

    int i = 0;

    for(i = 0; i < 10; i++) {
        printf("Number %d\n", i);
    }

    return 0;

}
```

Curly brace placement example 2

```
#include <stdio.h>

int main()
{
    int i = 0;

    for(i = 0; i < 10; i++)
    {
        printf("Number %d\n", i);
    }

    return 0;

}
```

Indentation example 1

```
for i in range(10):
    print("Number", i)
```

Indentation

There are a few reasons for code indentation. The first is that it makes the code easier to read and understand. Indentation allows you to separate thoughts and ideas into blocks of logic. However, some programming languages make indentation a part of the syntax of the language.

Python, for example, uses indentation to indicate control structures, so indentation is required or the program won't run or it will behave differently than expected. By using indentation, the need for curly braces is eliminated.

White Spaces

Indentation is in a category of characters called white spaces. They are the unseen characters that still take up space and are either ignored or have real meaning to the language. In the above code samples you will see a `"\n"` The backslash is used to escape characters and together are called an escape sequence. More specifically the `"\n"` is used to represent a new line character. In Python each print statement ends with a new line automatically, so there is no need to add the new line character to the print statement.

Tabs vs Spaces

This is another coding convention that can cause a lot of issues if you choose a style that others aren't used to. As we're now aware, indentation matters a lot. We can use tabs or spaces to indent. It's usually a good idea to indent by four spaces because four spaces is the default in most IDEs. Also, using tabs may be presented differently depending on which IDE/font/editor you're using. Luckily most IDEs offer an option to replace tabs with spaces, so pressing a tab, will automatically give you four spaces instead of one tab character.

Semantics

Semantics refers to the exact meaning of the symbols you've arranged into statements. One of the biggest challenges you'll face as a programmer is the art of dumbing down your mind so that you're able to think like a computer. This is an extremely frustrating part of programming. You have to constantly remind yourself that computers don't have feelings. They don't have the ability to get what you wanted to say. They just do EXACTLY what you tell them to.