CST0006 – Computer Programming Foundations

INTRODUCTION to WEEK05

In this lecture and lab section we are going to finally get into the real meat and potatoes of programming. We are going to learn about the different data types we can use and how to create variables and how to do operations on them.

We are going to cover the general concepts of variables and data types. It is a huge topic, but I want to cover the basics and some gotchas that we need to be aware of.

Data Types

Not all programming languages provide the same data types. When a programming language has a supported data type, those data types are known as built-in or primitive data types.

boolean	1/0, True/False, On/Off needs 1 bit
int	integer: a whole number
float	floating point number: a number with a fractional part
double	double-precision floating point value
char	single character
void	valueless special purpose type
array	a data structure that is a contiguous collection of elements of a data type
string	an array of chars

Signed/Unsigned

Sometimes we don't need to show negative numbers, so some languages allow you to specify if it will be signed or unsigned data that we will be storing.

Type

In most languages understanding how memory is used for variables is extremely important if you want to write good code that manages memory effectively. Consider the memory being allocated as a container that holds the size of the type of data that you are storing. Each memory container is exactly the size of the type of data you are storing.

Statically Typed

When declaring a variable, you give it a name and a type. This means that the name of the variable is always assigned to the same type and you cannot change the type of data that that variable holds.

Dynamically Typed

Python is a dynamically typed language. This means that the data type belongs to the value, and not to the variable that points to the value. Consider variables in this situations like like a tag.

Variables

Variables are named locations in storage (memory) that hold a data type. Some languages demand you include the data type you want to use, others guess at it by what you put into that named storage location.

One of the reasons we need to declare a type when naming a variable is because the size of the data needed to store the information. Otherwise you run the risk of overflowing the space allotted.

Variable Rules

What's true for most programming languages is:

- No keywords
- Case sensitivity matters var and Var are different. Camel Case vs Underscores
- Don't start with a digit

Declaring vs Defining

When you declare a variable, you are telling the compiler or interpreter about the type, size and name of the storage location needed for the variable.

When you define a variable, you are giving the declared variable the actual information that you want stored in that memory location.

C programming example:

```
int i;
i = 0;
```

Python example:

```
i = 0
```

Declaring and Defining

C programming example of declaring and defining in one statement:

```
int i = 0;
```

Mutable vs Immutable

Mutable: allows the data to change

Immutable: doesn't allow the date to change

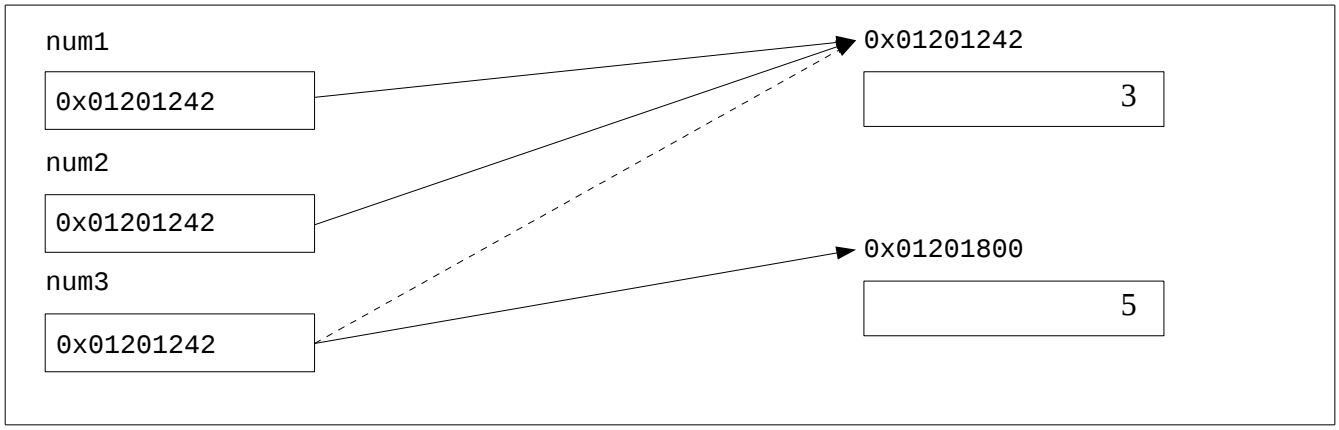
Variables

Reference to objects coding example in Python:

```
num1 = 3
num2 = num1
num3 = 3
num3 = 5

print(str(hex(id(num1))).upper())
print(str(hex(id(num2))).upper())
print(str(hex(id(num3))).upper())
```

Visual example



Operators

An operator is a special character that preforms mathematical, logical or relational operation on data. These operations come predefined within the language. WARNING: Not all operators do the same thing in every language. So you need to know ahead of time what you want to do and find the operator that does that task.

Operators follow the BEDMAS rules of calculation.

Bitwise Operators

Bitwise operators preform their operations on the binary representation of operands and do calculations based on those 1s and 0s.