Programming Languages, Software Development, Variables, Working with User Input, Math Operators & Data Types
Communicating With Machines

What we say to dogs
Okay, Ginger! I've had it!
You stay out of the garbage!
Understand, Ginger? Stay out of the garbage, or else!

What they hear
blah blah GINGER blah
blah blah GINGER blah
blah blah GINGER blah
blah blah GINGER blah...
Communicating With Machines

- Computers operate entirely on binary data
- Humans think in symbols
- How do we tell the machines what we want them to do?

What’s 8.875 percent of $99.99?

```
10011101 01101011
01010001 10101110
11110110 11000100
...```

Programming Languages

- Use symbols humans understand (words, numbers, operators)

- Every arrangement of symbols in the language has a consistent, formal meaning (unlike natural human languages, which are full of ambiguity)

- Have a rigid structure (syntax) that can be broken down into meaningful units (parsed) by a computer program

- Are then converted into machine code, either by being compiled or interpreted
High-Level Programming Language Structure
Programming Language Structure: Key Words

- Defined list of words that make up the language
- Sometimes called “Reserved Words”

```
and    del     from    not    while
as      elif     global  or     with
assert  else     if      pass    yield
break   except   import  print
class   exec     in      raise
continue finally is      return
def     for      lambda  try
```
Programming Language Structure: Operators

- Special symbols that perform certain actions on pieces of data

answer = 5 + 2

name = ‘Harry’ + ‘Potter’

average = 250 / 300
Programming Language Structure: Syntax

- Set of rules that must be followed when writing a program

```python
if name == 'craig':
    print('Hi there!')
else:
    print('Who are you?')
```
Programming Language Structure: Statement

- Instructions that you write, consisting of keywords, operators, punctuation, etc

average = average * 2
Programming Language Structure:

Statement

- Instructions that you write, consisting of keywords, operators, punctuation, etc

Expression

- Anything that evaluates to return a value. (E.g. a mathematical expression.)

```
average = average * 2
5
average * 2
average == 5
```
Programming Paradigms

- Some languages are designed around particular paradigms.
- Some examples:
  - Procedural: Algol, Fortran, COBOL, BASIC, C
  - Object Oriented: C++, Java
  - Functional: Lisp, Haskell, Erlang, Ocaml
  - Logic: Prolog
  - Aspect Oriented: extension of OOP. AspectJ, Aspect C++

- Most modern languages accommodate multiple paradigms, but are more friendly to some than others.
This semester we will be working with Python

High level interpreted language

Used extensively as both a production language as well as a teaching language

Two modes
  - Interactive
  - Script

IDLE
  - Integrated Development Environment
Software Development
Software Development Loop

- Design
- Write
- Correct Syntax
- Test
- Correct Logic

The loop moves in a clockwise direction, starting with Design and ending with Test.
Design

- Programmers need to establish a solid foundation before they begin coding a project.
- This involves understanding the task that the program must perform.
- Next, we need to determine the steps that need to be taken in order to perform the task.
Understand the Task

- Most programming projects begin with an interview with the end user
- Programmers must ask lots of questions and get as many details as possible about the task
- Follow ups are usually required
- After an interview a programmer generally constructs a “software requirement” document
- This amounts to an agreement between the end user and the programmer on what the program should actually do
Understand the Task

- How the customer explained it
- How the Project Leader understood it
- How the Analyst designed it
- How the Programmer wrote it
- How the Business Consultant described it
- How the project was documented
- What operations installed
- How the customer was billed
- How it was supported
- What the customer really needed
Determining the Steps Needed to Perform a Task

- Next we break down the task into a series of concrete steps that can be followed (like a recipe).

- Remember that computers need each step to be broken down into minute detail.

- They don’t have the ability to infer intermediate steps like we can!
“A series of well defined, logical steps that must be taken in order to perform a task”

Algorithms serve as a necessary intermediate step between understanding the task at hand and translating it into computer code
Pseudocode

(aka “fake” code)

- A useful technique for breaking down an algorithm into meaningful chunks and aligning them with the toolset of a language
- In pseudocode we don’t have to worry about syntax or spelling
- Allows us to think through the algorithm in natural language

For example, write a thermostat program in pseudocode:

- Measure the temperature of the room
- If it's over the temperature threshold
  - turn on the air conditioner
Flowcharts

- A graphical model of the program logic that helps visualize the task at hand

- Some common elements in a flow chart:
Input, Processing & Output

- Programs typically perform the following 3 steps
  - INPUT is received
  - Some kind of PROCESSING is performed
  - OUTPUT is produced
Input

- Can be from a variety of sources
  - User: keyboard, mouse, etc.
  - Hardware: scanner, camera, etc.
  - Data: file, the Internet, etc.
A series of mathematical or logical processes are applied to the input
- Compare values
- Add, multiply, divide or subtract numbers
- Perform calculations on an item over and over again (i.e. blurring an image)
Output

- Some kind of tangible / visible / readable product is constructed
  - Printout
  - Screen display
  - 3D fabrication
Any questions?

.... then on to Python!
Python: Getting Started

- **IDLE: Integrated Develpoment Environment**
- **Has two modes**
  - Interactive – commands are immediately processed as they are received
  - Script – allows to write a program (saved as a “text file” on your computer) and have your commands processed whenever you’d like
- **We will mainly be using “script” mode during this semester**
Python: Creating a new program

- Open IDLE

- Click on File -> New Window

- Click on File -> Save

- Save your file somewhere on your computer. You will need to add the ‘.py’ file extension to your file if IDLE does not place it there automatically.

- With your program open, click on Run -> Run Module

- If you need to open your program you can click on File -> Open and browse to find the desired Python source file (.py
Your First Program: "Hello, World!"
>>> print ("Hello, world!")
Hello, world!