



CMSC 150

INTRODUCTION TO COMPUTING

ACKNOWLEDGEMENT: THESE SLIDES ARE ADAPTED FROM SLIDES PROVIDED WITH INTRODUCTION TO PROGRAMMING USING PYTHON, LIANG (PEARSON 2013)

LECTURE 1

- INTRODUCTION TO COURSE
- COMPUTER SCIENCE
- HELLO WORLD



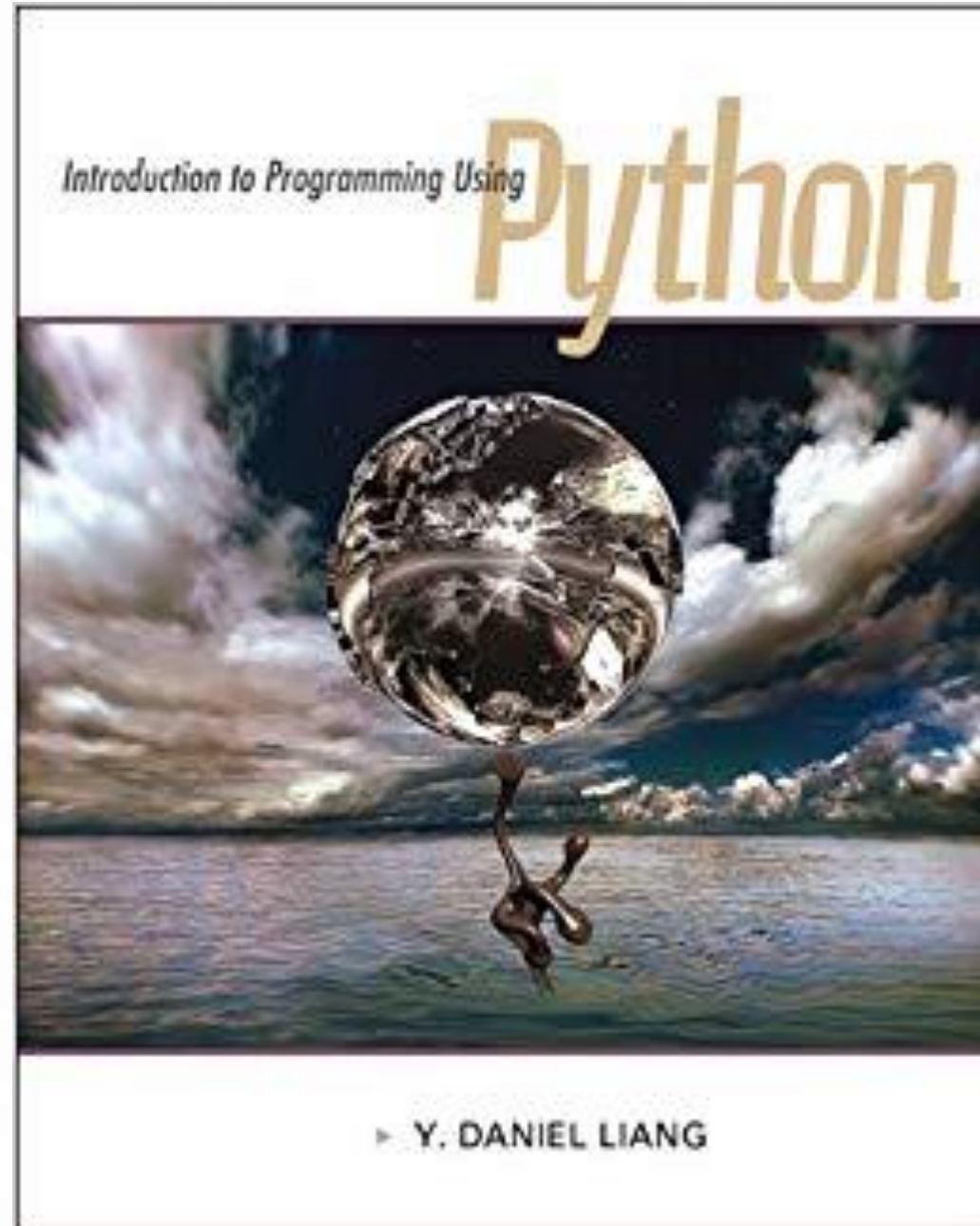
WELCOME

- Questions?
- 



SYLLABUS

- Questions?



The background features a light blue, concentric circular pattern. In the four corners, there are decorative circuit-like lines in a slightly darker blue, consisting of straight lines and small circles, resembling a stylized PCB or network diagram.

WHAT IS COMPUTER SCIENCE AND COMPUTING?

COMPUTER SCIENCE

- Your thoughts?
- Google: “The study of the principles and use of computers”
- Wikipedia: “The scientific and practical approach to computation and its applications”
- Dictionary.com: “The science that deals with the theory and methods of processing information in digital computers, the design of computer hardware and software, and the applications of computers”
- Edsgar Dijkstra: “Computer Science is no more about computers than astronomy is about telescopes”

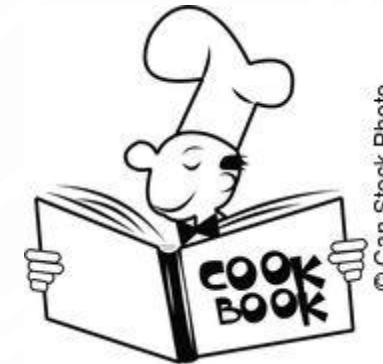
COMPUTER SCIENCE

- Study of algorithms
- Study of computing tools
- It is not just:
 - Programming
 - Microsoft office
 - Typing
 - Electronics
 - Etc.

Input



Algorithm



Output



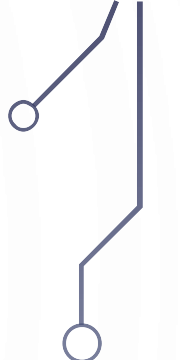
ROBOTICS

- Design, automation, and application of robots
- A **robot** is a mechanical machine capable of carrying out a complex series of actions automatically
- We will study computer science through this application
 - Allows a physical manifestation of our programs
 - Complex – allows a deep-dive into the subject
 - Fun!





PROBLEM

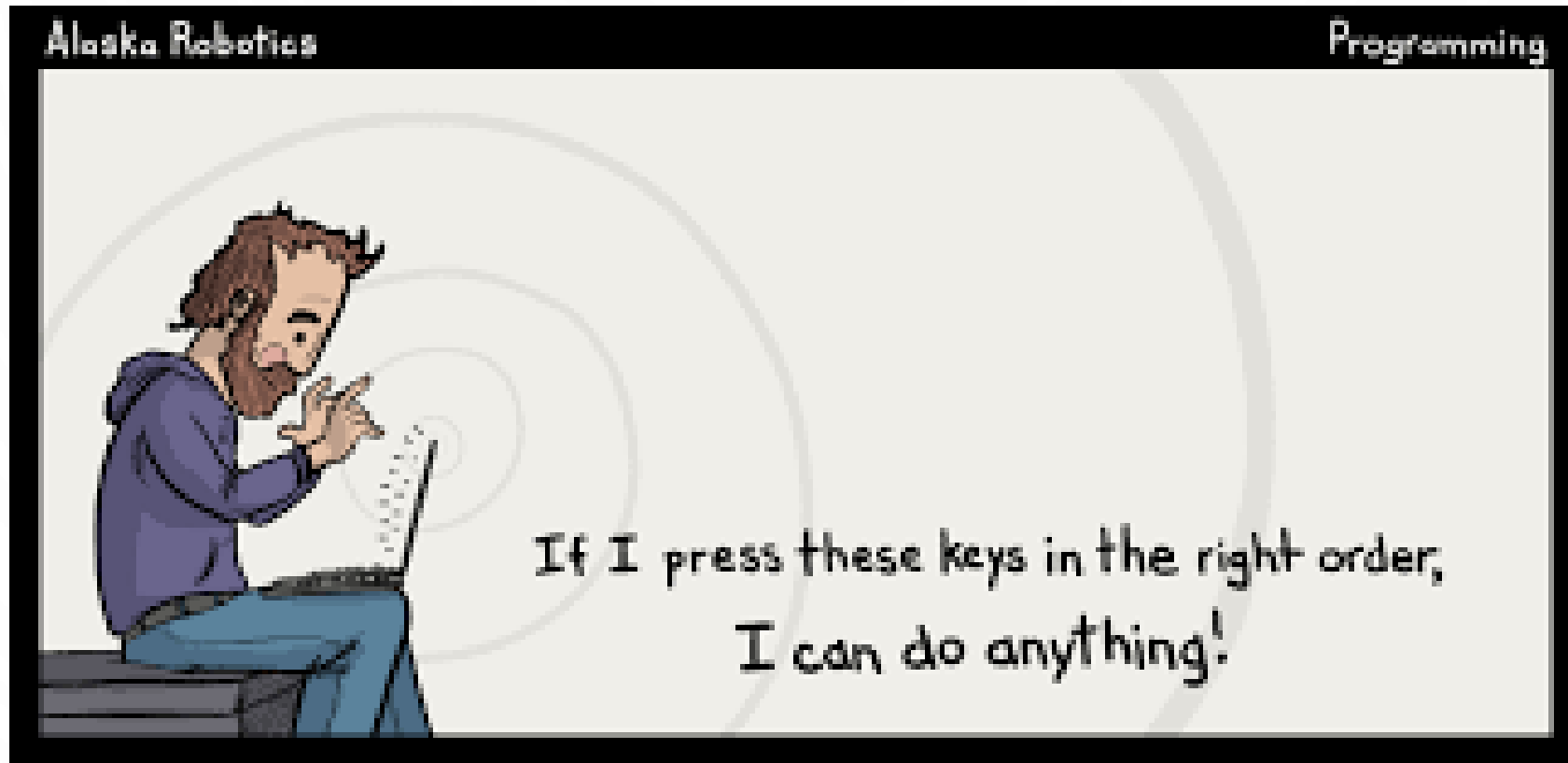
- Work in pairs/triplets
 - Create a methodology to move a robot from point A to point B
 - Determine the input
 - Determine the algorithm
 - Determine the output
 - Put another way...tell a computer how to do this task
- 

PROGRAMMING

- Even though computer science is not about the computer, we still need to tell the computer what to do!
- We do this through **programming**, or the act of writing a **computer program**, known as **software** – its just instructions to the computer
- Programming allows us to push the boundaries of science, view imaginary worlds, and improve our daily lives!

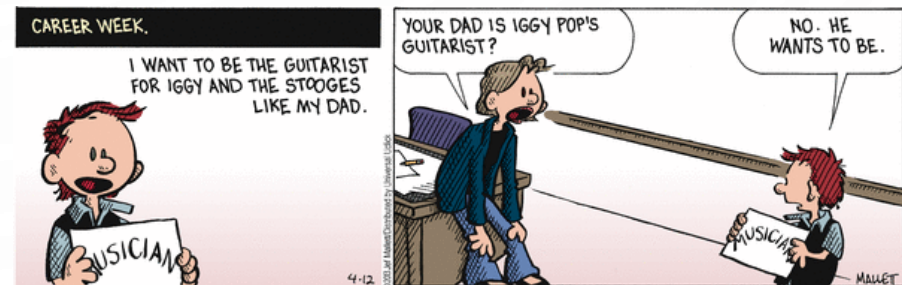


PROGRAMMING



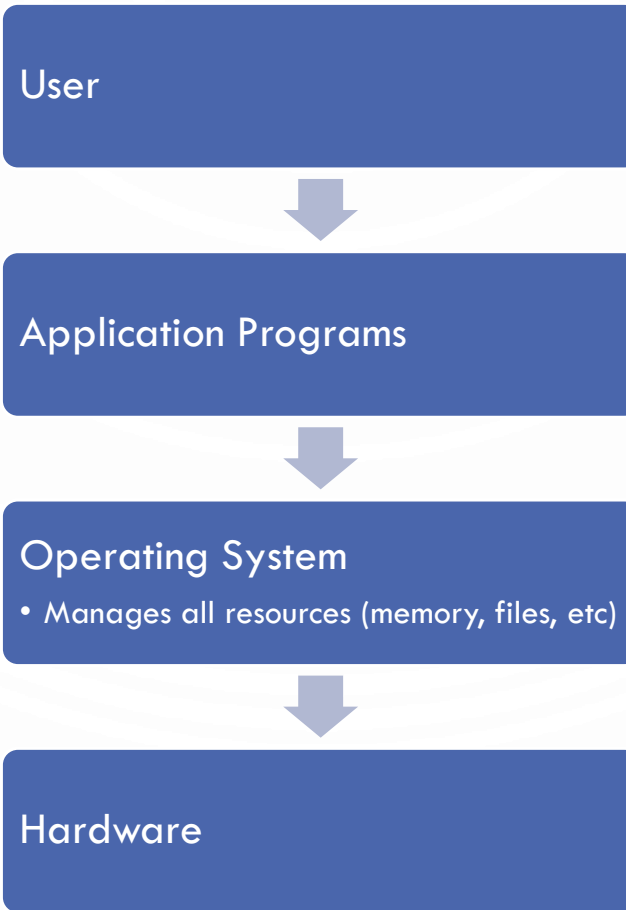
A BRIEF NOTE ON PROGRAMMING LANGUAGES

- Machine code – 0's and 1's...or simple commands. It is the set of primitive instructions built into the computer's architecture or circuits. Extremely tedious and error prone
- Assembly code – simple commands (`ADD ra rb rc`) to make programming easier to understand. An assembler translates the commands to machine code. Extremely tedious but less error prone.
- High level languages – English-like commands that allow programming to be less tedious, less error prone, and much more expressive! Examples: Java, C++, Matlab, etc
- Why we don't use Natural language (English) – Its ambiguous...which vs which or break vs break or run vs run...ah the madness!!!!



COMPUTER ORGANIZATION

A SOFTWARE PERSPECTIVE




COMPUTER ORGANIZATION

A HARDWARE PERSPECTIVE

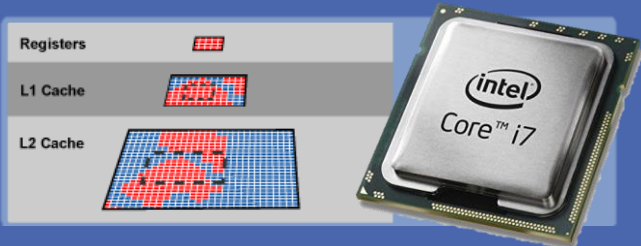
Input

- Files
- Keyboard
- Mouse
- Etc.




Central Processing Unit (CPU)

- Processes commands as 0's and 1's
- Performs arithmetic
- Requests (reads) and writes to/from memory




Output

- Monitor
- Force feedback
- Files
- Etc.

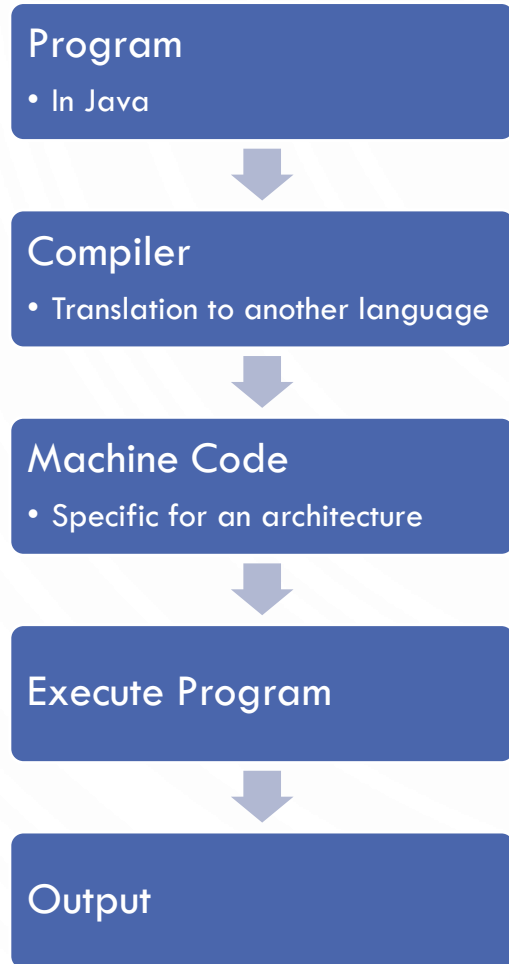


Memory

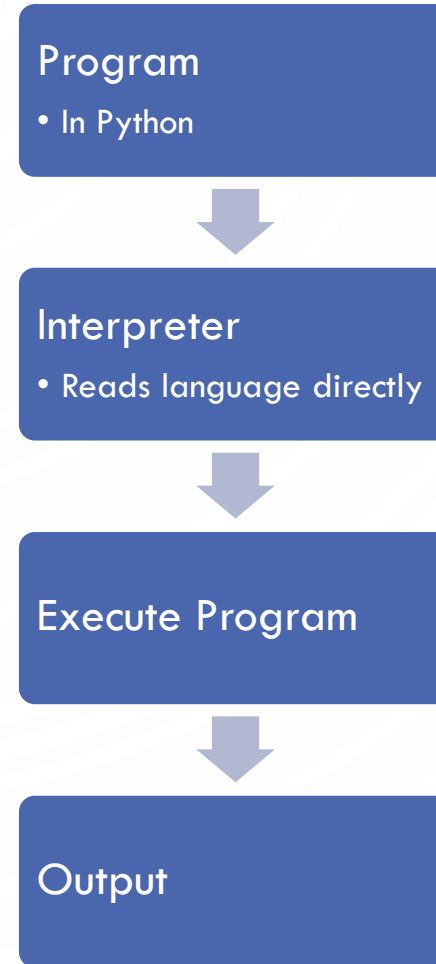
- Data encoded as 0s and 1s
- Cache
- Random Access Memory (RAM)
- Hard drive



COMPILING A HIGH LEVEL PROGRAM



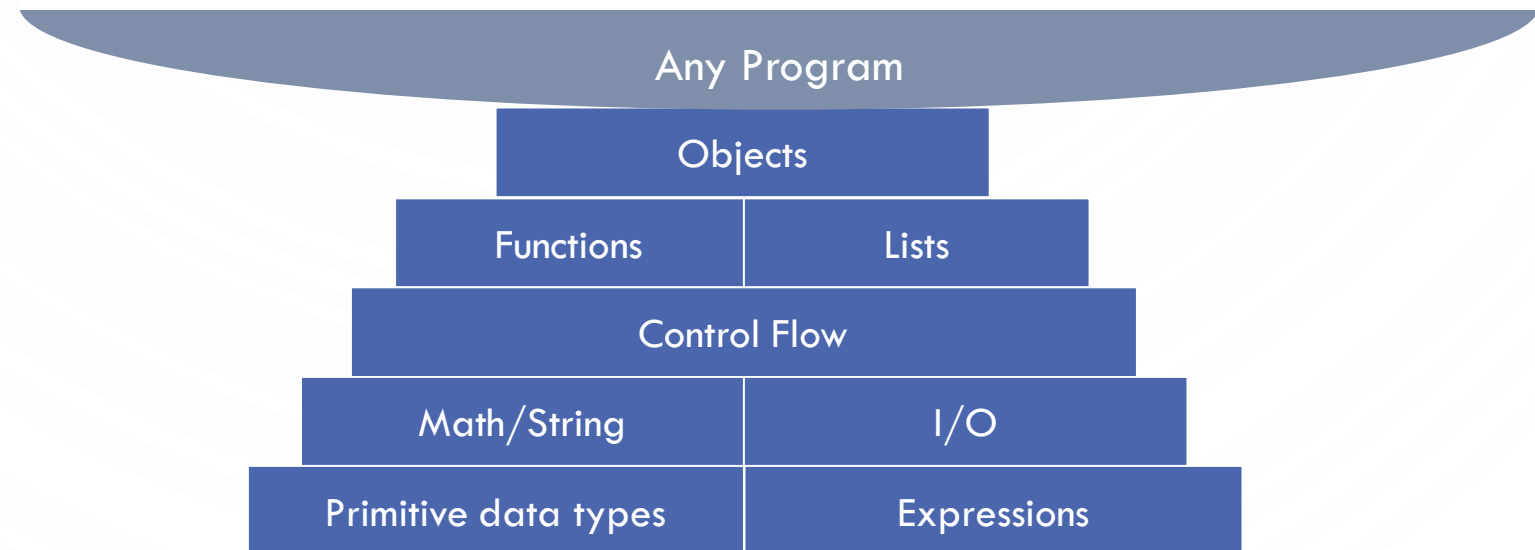
Using a compiler



Using an interpreter

HOW DO WE PROGRAM THE COMPUTER?

- We will use Python
 - NOTE – This is an arbitrary choice. All languages build on the same basic building blocks discussed in the course. So Python is merely the vessel to our exploration of computing!
- Major concepts:



WHY PYTHON?

- Python
 - Widely used.
 - Widely available.
 - Embraces full set of modern abstractions.
 - Variety of automatic checks for mistakes in programs.
- Our study will
 - Use a minimal subset of Python.
 - Develop general programming skills that are applicable to many languages.
 - IT IS NOT ABOUT THE LANGUAGE!!!

“ There are only two kinds of programming languages: those people always [gripe] about and those nobody uses.”

– Bjarne Stroustrup



PYTHON2 VS PYTHON3

- We will specifically use Python3 in this class
- Many resources online teach/use Python2
- Python3 is not backwards compatible, so be careful with using online resources





WALK THROUGH PROGRAMMING

1. PROGRAM HELLOWORLD
 2. PROGRAM HELLOGRAPHICS
 3. PROGRAM HELLOROBOTICS
 4. TURN THIS SET OF PROGRAMS IN THROUGH GITHUB
- 
- 

HELLO WORLD

HelloWorld.py

1. #Print two messages
2. `print("Hello class")`
3. `print("Welcome to Robotics")`

- Run: `python3 HelloWorld.py`

TRACING

HelloWorld.py

```
1. #Print two messages  
2. print("Hello class")  
3. print("Welcome to Robotics")
```

Output

```
Hello Class  
Welcome to Robotics
```

Memory

- Tracing is the activity of following a computation by hand. We will regularly do this in and out of class
- Not a classroom activity! Professionals do this regularly on sections of programs
 - Typically to determine when something goes wrong

VS CODE AND TERMINAL

- In this class, we will exclusively use Visual Studio Code (VS Code) text editor to write programs, its terminal to run our programs, and GIT to turn in assignments
- Download and setup VS Code for activities tomorrow
 - Follow course website to setup python and GIT



TERMINAL REFERENCE GUIDE

- A **terminal** is a window to interact with your operating system through commands. Things to know:
 - You are always in a specific directory, called the **current (or working) directory**
 - Filenames are specified “relative”ly – this means you have to be in the same directory or refer to the location relative to your current directory
- Common commands (to move through folders and create them (may be different on Windows machines))
 - **pwd** – print the current directory
 - **cd** – change directory, e.g., **cd** Desktop
 - **ls** – print everything in a directory
 - **mkdir** – make a new directory, e.g., **mkdir** HelloWorldProject



ANATOMY OF A PYTHON PROGRAM

- Statements
 - Comments
 - Indentation
- 
- 

STATEMENT

- A **statement** represents an action or a sequence of actions.
- The statement `print("Hello class")` in the program is a statement to display the message "Hello class".

HelloWorld.py

```
1. #Print two messages
2. print("Hello class")
3. print("Welcome to Robotics")
```


INDENTATION

- The indentation matters in Python. Note that the statements are entered from the first column in the new line. It would cause an error if the program is typed as follows, for example:

HelloWorld.py

```
1. #Print two messages
2.  print("Hello class")
3.  print("Welcome to Robotics")
```

SPECIAL SYMBOLS

Character Name	Description
<code>()</code>	Opening and closing parentheses Used with functions.
<code>#</code>	Pound sign Precedes a comment line.
<code>" "</code>	Opening and closing quotation marks Enclosing a string (i.e., sequence of characters).
<code>''' '''</code>	Opening and closing quotation marks Enclosing a paragraph comment.

RESERVED WORDS

- **Reserved words** or **keywords** are words that have a specific meaning to the compiler and cannot be used for other purposes in the program. We will see many of these during the course of the semester. The previous program doesn't have any specifically.



PROGRAMMING STYLE AND DOCUMENTATION

- Appropriate Comments
 - Naming Conventions
 - Proper Indentation and Spacing Lines
- 
- 

APPROPRIATE COMMENTS

- Include a summary at the beginning of the program to explain what the program does, its key features, its supporting data structures, and any unique techniques it uses.
- Document each variable, function, and class
- Include your name and a brief description at the beginning of the program.


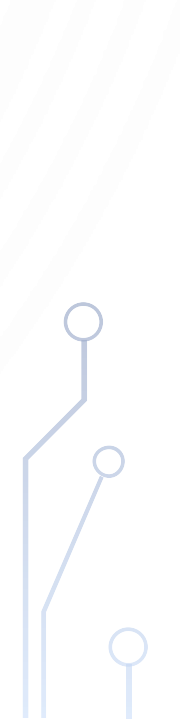


NAMING CONVENTIONS

- Choose meaningful and descriptive names.
- 


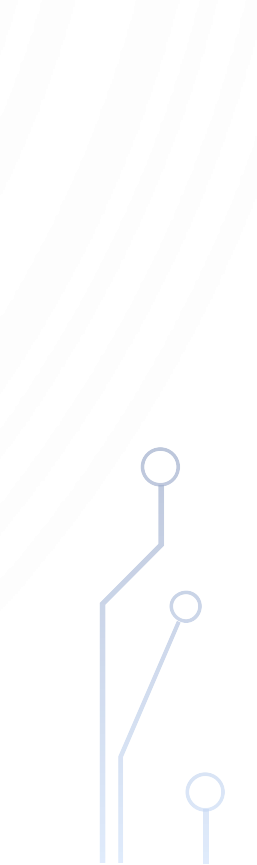


PROPER INDENTATION AND SPACING

- Indentation
 - Indent two spaces.
 - A consistent spacing style makes programs clear and easy to read, debug, and maintain.
 - Spacing
 - Use blank line to separate segments of the code.
- 
- 



PROGRAMMING ERRORS

- **Syntax Errors**
 - Error in writing python syntax
 - **Runtime Errors also called Exceptions**
 - Causes the program to abort
 - **Logic Errors**
 - Produces incorrect result
- 
- 

SYNTAX ERRORS

- **Syntax errors** are errors from incorrectly written Python code.
- Anatomy of a compiler error:
File "filename.py", line num
ErrorType: Confusing description of error including code where it occurs.
- Deal with errors by experience, google, stack overflow, etc. After you have exhausted these resources...piazza/ask me. Advice, always handle the first error...not the last one.

HelloWorld.py

```
1. //Print two messages
2.   print("Hello class")
3.   print(Welcome to Robotics")
```

Can anyone spot the syntax errors?

RUNTIME ERRORS

- Runtime errors occur from impossible commands encountered while executing the program
- Error message shows a "traceback" of the program execution. Right now, just know that this tells where/why the error occurs.

HelloWorld.py

```
1. print(1/0)
```

LOGIC ERRORS

- Logic errors are incorrect computations that run without exceptions but produce the incorrect output

HelloWorld.py

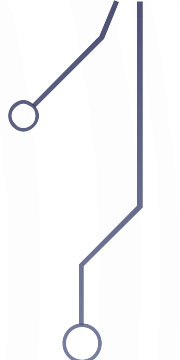

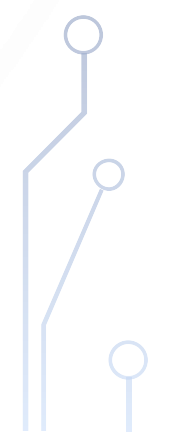
```
1. #Celcius conversion
```

```
2. print("Celcius 35 is Fahrenheit", (9//5)*35+32)
```

Can anyone spot the logic error?



HELLO GRAPHICS

- Python makes programming with graphics and user interfaces (GUI) easy
 - Turtle – Simple turtle graphics library
 - Tkinter – Simple GUI library
 - We will occasionally operate with graphics and GUIs
- 
- 
- 

HELLO GRAPHICS

HelloGraphics.py

```
1. # Grab the correct python
2. # component
3. import turtle
4.
5. # Draw text
6. turtle.showturtle()
7. turtle.write("Welcome")
8.
9. # Movement
10. turtle.forward(100)
11. turtle.right(90)
12.
13. # Color
14. turtle.color("red")
15. turtle.forward(50)
16. # More advanced options
17. turtle.penup()
18. turtle.goto(-100, -50)
19. turtle.pendown()
20. turtle.goto(-100, 0)
21. turtle.color("green")
22. turtle.circle(25)
23.
24. # Keep the window open
25. turtle.Screen().exitonclick()
```

HELLO ROBOTICS

- We will use both the GoPiGo platform and the CoDrone platforms
- Description of the robots found on my webpage
- We must run our programs remotely (use **type** instead of **cat** on windows)

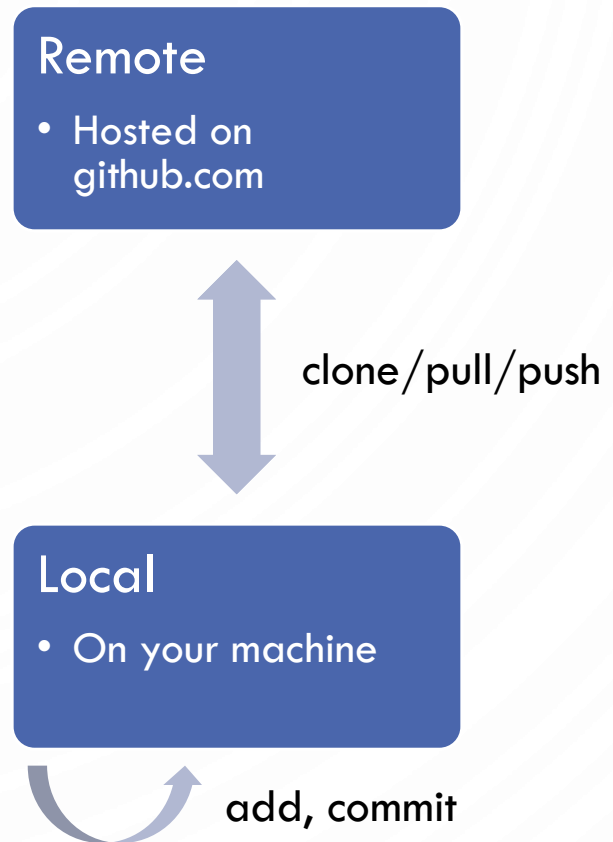
```
cat HelloRobotics.py |  
ssh pi@gopigo00 python3 -
```

HelloRobotics.py

```
1. import time  
2. from easygopigo3 import EasyGoPiGo3  
3. # Make a robot  
4. robot = EasyGoPiGo3()  
5.  
6. # Use the robot  
7. robot.forward()  
8. time.sleep(1)  
9. robot.right()  
10. time.sleep(2)  
11. robot.stop()
```

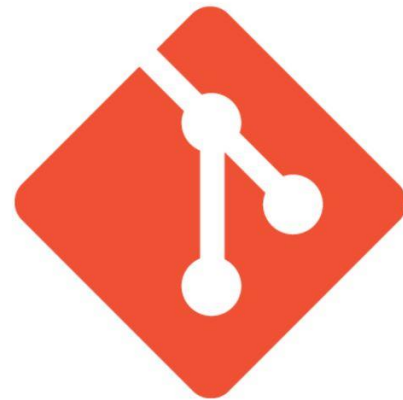
GIT

- GIT is a version control software that is a professional tool
- We will use it for turning in assignments
- GIT is a distributed repository management, so basic items
 - remote – the remote server
 - clone/pull – copy/update the local copy from the remote copy
 - push – send changes to the remote copy
- Changes are managed in the following phases
 - Unstaged – changes you did without notifying GIT
 - Stages – changes that you have notified GIT of (with add)
 - Committed – changes saved by GIT locally (with commit)
 - Push – changes shared by GIT remotely



FOLLOW ALONG WITH TURNING IN ON GIT

- Accept assignment invitation
- Clone your repository
- Make changes and edit the Coverpage
- Stage, Commit, and Push
 - Can be done as much as you need before the deadlines



git

EXERCISES

1. Create a program to share three things about yourself.
2. Write a program to show a pentagon. Try to use different colors for each edge.
3. Write a program to make the GoPiGo move in a simple pattern (like a pentagon)
4. Work on Programming Assignment 0

