CS-110 Notes:

* Skills needed for Comp Sci
  + Problem Solving
  + Algorithms
* Files on your computer
  + Binary File
  + Text File
* Text file 🡪 binary file (machine code)
  + Compiled
    - Turns entire text file into machine code
    - Faster
* Interpreted
  + - Doesn’t require that your whole code works
    - Python is interpreted language
      * Aka scripting language
* Python
  + Two Modes
    - Shell Mode
    - Programmatic mode
* Computer Functions
  + Arithmetic/ Logic
  + Output
    - To the screen
  + Input
  + Conditional
    - Only under certain conditions
  + Repetition (Jump)
    - Loop is an example
    - Repetition
* Semantic Errors

🡪 making a logic error

* + Writing code that the computer doesn’t understand
  + Syntax error

🡪making an error in language

* + Runtime error
    - Most difficult to debug
* Data
  + Referred to as an “object”
    - Must have data type
      * Whole number = integer
      * Text = string
      * Anything with a decimal point = float
* String
  + Can be single quoted or double quoted or triple quotes
  + Double quote when you have apostrophe
  + Single quote when you have double quotes
  + Triple quote when you have both in the string
  + “1”+”1” = “11” 🡪 adds two strings together
  + To add string with space you must add space in one string
* Integer
  + Do not need comma 🡪 will result in two different values
    - Ex. 42,000 🡪 (42,0)
  + Chops off decimals
    - Ex. int(3.6) 🡪 3
  + Always get integer back when you use two integers except in division
    - Can use integer division to not get float
  + Exponents you use \*\*
    - Ex. 2\*\*3 = 8
  + To get remainder you use %
    - Ex. 2%5 = 2
      * 5 goes into 2 zero times and leaves you with a remainder of 2
* Store Data
  + = is for assignment not equality
    - Ex. My\_var = 2
      * Can’t start with number
      * Can’t have a space
      * Have a few words you can’t use for variable names s/a
        + True
  + Statement
    - Assigning value to variable
      * Ex. 3+5\*a = b
  + Call
    - Calls up data
      * Ex. Print(b) = 28
* Save Input
  + Ex. My\_var = input(“Please enter a value”)
  + Override the last entry
    - Ex. My\_var = input(“Please enter a value”)

Print (my\_var) #will print first input value

My\_var = input(“Please enter another a value”)

Print(my\_var)#will print newest input value

* Convert string to float
  + Float(input(\*\*\*insert\_text\*\*\*))
* Open File
  + Cd Documents
  + Cd CS-110
  + \*\*Run Program\*\*
    - python3 program\_name
* Define start of program
  + Def main():
    - Everything after must be indented 4 spaces or one tab
* To end function
  + Main()
* Basic Rules of Codes
  + DRY= Don’t repeat yourself
  + Make it reusable
* Modules
  + Code libraries
  + To import modules
    - Ex. Import turtle
      * Go at the top before def main():
* Tutle Example

Def main():

Window = turtle.Screen()

Leonardo = turtle.Turtle()

Leonardo.shape(‘Turtle’)

Leonardo.color(‘blue’)

Leonardo.forward(100)

Window.exitonclick()

* Python is sequential

Ex. Turtle Documentation:

3 actions

1) circle(); drawing, diameter

2)towards(); state, distance

3)home(); move, no parameters

4) set undo buffer; set number of undo’s before the turtle forgets it

Goto(0,0)

* Loops
  + Val = range (num)
  + For loop
    - For num in range(4)

Michelangelo.forward(50)

Michelangelo.left(90)

* + - For num in range (0,8,2)
      * Runs 4 times; goes to 8 counting by twos
      * Print(i, end=” ”)
    - Define function

Def foo(): #function header

X=5+3 #function body

Print(x)

Def main()

…

foo()

…

foo()

…

foo()

* By calling foo() it prints function
* Function should always be lower case

\*\*\*\* to exit function use “return”\*\*\*\*

\*\*\*\* to exit loop use “break”\*\*\*\*

\*\*\* “Continue” jumps back to top of loop\*\*\*

* Def foo(param1,param2,param3)
  + If function takes three parameters, you must give it three
* Rule of thumb
  + Valid input and valid function yields valid results

-------------------------------------------------------------------------------------------------------------------------------

How to get turtle to draw square:

import turtle

#create a function to draw a square

def drawSquare(turtle, size):

for i in range(4):

turtle.forward(size)

 turtle.left(90)

def  main():

wn = turtle.Screen()              # Set up the window and its attributes

wn.bgcolor("lightgreen")

alex = turtle.Turtle()            # create alex

drawSquare(alex, 50)             # Call the function to draw the square passing

wn.exitonclick()

main()

Programming Patterns:

* Accumulator
* If statement
  + elif statement will only execute first if statement
  + normal if statements will execute all if statements the argument statisfies

Debugging

* Flush statements
  + Used primarily for debugging purposes

Accumulator

* Pattern is the same
  + Set accumulation value
  + Alter accum value

Return Statement

* What happens in the function stays in the function
* Return Statement is like passing off the ingredients in cookies analogy
  + Functions should always return something

Print (5+2)

* Prints 7

Booleans

* False and…. Always = False
* True or….. Always = True
* Always start with inner most parentheses in not statements

Import Random Number

* Import random module
* Random.randrange(put range here)

Set Screen Size

* Different from setworld coordinates
  + Set world changes canvas size
  + Set screen size changes window size

---------------------------------------------------------------------------------------------------------------------

While Loop

* Continues to loop until some condition is met
* Scope of i is outside while loop
  + In for loop it is recreated every iteration

---------------------------------------------------------------------------------------------------------------------

List:

In order to add range(5) and [1,2] you must do:

* List(range(5)) + [1,2]sss

Class Point:

#xvalue

#yvalue

#distance from (0,0)

def \_\_init\_\_(self, ):

self.x=0

self.y=0 🡨 adds attributes to class

---------------------------------------------------------------------------------------------------------------------

GUI(Graphical User Interface):

* GUI
  + Screen/Window
  + Components
    - Interactable things that the everyday person knows how to use
* What we’ve been doing is procedural programming
  + It’s all sequential
* Object oriented programming
  + Things are done based on the states of objects
* Event Driven Programming
  + State
  + Behaviours
* Aspects of gui
  + Controller
    - Models and directs events
  + Views
    - Displays actions
  + Models
    - Data/state
* Main Loop

Loop

Events

If event 1

Elif event 2

Alter State

Update View