SOFTWARE DEVELOPMENT, VARIABLES, WORKING WITH USER INPUT, MATH OPERATORS & DATA TYPES
Software Development
Software Development Loop

1. Design
2. Write
3. Correct Syntax
4. Test
5. Correct Logic

The process flows in a loop, with each step leading back to Design.
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.
- You may have heard the terms ‘user-centered design’ and ‘UX designer’ which are jobs and design methodologies that are formed out of this process.
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
Flowcharts

- A graphical model that helps programmers conceptualize the task at hand
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)
Some kind of tangible / visible / readable product is constructed:
- Printout
- Screen display
- 3D fabrication
Boiling a Pot of Water
Boiling a Pot of Water

- Find a measuring cup
- Pick up measuring cup
- Find a sink
- Walk to sink
- Turn on water
- Fill the measuring cup with 2 cups of water
- Turn off water
- Find a pot
- Pick up pot
- Find a stove
- Walk to stove
- Place the pot on the stove
- Pour the water into the pot
- Put measuring cup down on countertop
- Turn the heat on the stove to Medium-High
- Wait until the water begins to rapidly bubble
- Turn off stove
Understanding a Task

- Turning on Netflix

Netflix and chill?
Algorithms

“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

These aren’t necessarily overly complicated!
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
Pseudocode/Algorithm Practice

What to do in an awkward situation

```
Awkward?
  No  Yes
    Yes
    No
      Free food?
        Yes
        No
          Stay
          Leave
```
Any questions?

…. then on to Python!
Python: Getting Started

- IDLE: Integrated Development 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 extension)
Functions

- A “function” is a pre-written piece of computer code that will perform a specific action or set of actions.
- Python comes with a number of built-in functions, and you can also write your own (more on that later in the semester).
- Functions always begin with a keyword followed by a series of parenthesis.
  - Ex: `print ()`
- You can “pass” one or more “arguments” into a function by placing data inside the parenthesis.
  - Ex: `print (‘Hello, World!’)`
- Different functions “expect” different arguments. The print function, for example, expects printed text as an argument.
- When you ask Python to run a function we say that you have “called” the function.
Strings

- Data that is textual in nature (i.e. “Hello, World!”) is called a “String”
- Strings can contain 0 or more printed characters
- “String Literals” are strings that you define inside your program. They are “hard coded” values and must be “delimited” using a special character so that Python knows that the text you’ve typed in should be treated as printed text (and not a function call)
  - Ex: print (‘hello, world!’)
- Python supports three different delimiters
  - The single “tick” (‘ )
  - The single quote (“ ”)
  - The triple quote (‘”’ )
Printing multiple arguments

- The print() function can accept zero or more arguments.

- If you decide to pass multiple arguments to the print() function, you should separate them by a comma. Example:

  ```python
  print(“Hello! My name is”, “Craig”)
  ```

- Note that when Python executes the function call above, it will insert a space between the two arguments for you automatically.

- Also note that the print() function will automatically add a line break after it prints the last argument it was passed. We will discuss how to override this behavior in our next class.
Line endings

- When using the print() function you probably have noticed that Python automatically places a newline character at the end of each line.
- You can override this behavior and have Python use a character of your choice by using the optional ‘end’ argument when using the print() function.
- Example:

```python
print ('one', end='')
print ('two', end='')
```
Separating print() function arguments

- By default, Python will place a space between arguments that you use in print() function calls.
- You can override this behavior by using the optional ‘sep’ argument.
- Example:

```python
print (‘one’, ‘two’, sep=‘*’)
```

# output: one*two
Combing both line endings and separators

- You can use both the ‘sep’ and the ‘end’ arguments at the same time in the same `print()` function call.

- Example:

  ```python
  print ('a', 'b', 'c', sep='*', end='')
  ```
Variables
Variables

- Variables are like little “buckets” that can store information in your computer’s memory.
- You will be using variables constantly when writing your programs in order to keep track of various pieces of information.
- You can create a variable by using the following syntax:
  - variablename = somedata
- Examples:
  - speed = 5
  - myname = ‘Batman’
- The ‘=’ symbol is called the ‘assignment operator’ and will cause Python to store the data on the right side of the statement into the variable name printed on the left side.
Variables

variable_name = 'Hello, World'
Variable Naming Rules

- You can’t use one of Python’s “reserved words” (see the next slide for a list)
- Variables can’t contain spaces (though you can use the underscore character (“_”) in place of a space)
- The first character of a variable name must be a letter or the underscore character. Any character after the first can be any valid alphanumeric character (or the underscore character)
- Python is case sensitive
Python Reserved Words

(These words can’t be used when declaring a variable in your program)

'False', 'None', 'True', 'and', 'as', 'assert', 'break', 'class',
'continue', 'def', 'del', 'elif', 'else', 'except', 'finally', 'for',
'from',
'global', 'if', 'import', 'in', 'is', 'lambda', 'nonlocal', 'not',
'or',
'pass', 'raise', 'return', 'try', 'while', 'with', 'yield'
Legal or Illegal variable name?

class = 2
class_avg = 125
classAvg = 99
_class_avg = 99
2ndclassavg = 125
classavg! = 99
Legal or Illegal variable name?

```
class = 2  Illegal: “class” is a reserved word
class_avg = 125
classAvg = 99  Legal
_class_avg = 99  Legal
2ndclassavg = 125  Legal
classavg! = 99  Illegal: can’t start with a number

Illegal: can only use alphanumeric values and the “_” symbol in variable names
```
Common Variable Naming Conventions

rocketTopSpeed = 1000  
rocket_top_speed = 1000  
rocketTopSpeed = 1000  

# can be hard to read  
# underscored  
# “camel case”
Displaying Variables with the print function

- You can print the data that is currently being held by a variable by passing the variable name to the print() function as an argument. Example:

  ```python
  print (myvar)
  ```

- As with string literals, you can tell the print() function to display more than one item in the same function call by separating your arguments with commas. Example:

  ```python
  name_var = "Slim Shady"
  print ("Hi, my name is", name_var)
  ```

  ```
  >> Hi, my name is Slim Shady
  ```
Reassigning Variables

- Variables are called variables because they can “vary” the data that they store.
- You can re-assign the value that a variable stores by simply using a second assignment statement. Example:

```python
dollars = 10000000.99
print ('I have', dollars, 'in my account')
dollars = 9.99
print ('Now I have', dollars, 'in my account')
```
Multiple Assignments

- It's possible to set the value of multiple variables on the same line. For example:

  ```
  x, y, z = 'a', 'b', 'c'
  ```

- In this case the variables x, y and z will assume the values ‘a’, ‘b’, ‘c’ in that order.

- You can also assign the same value to multiple variables at the same time by doing the following:

  ```
  # a, b and c will all contain the integer 100
  a = b = c = 100
  ```
You’re working on a simple inventory management system for a small store. You’d like to store the name and price of two different products and print them out in the following format.

Item: Bread, Price: $ 2.99
Item: Eggs, Price: $ 1.99
Data Input
So far we have learned how to:
- OUTPUT information (via the print function)
- STORE information (via variables)

However, all of our programs have been “hard coded,” meaning that we have predefined the operating environment using information that we ourselves have defined. Example:

```python
mynname = 'Harry Potter'
print ('Welcome to my program', myname)
```
You can make your programs more interactive by involving the user in the process.

One of the simplest ways to do this is to request information from the keyboard using the `input()` function.

Example:

```python
# ask the user their name
username = input ('What is your name?')

# welcome them!
print ('Welcome,', username)
```
The Input function

- Input is a built-in function
- It accepts one argument – a String
- It then prompts the user with that string and waits for them to type in a series of characters. Your program will resume when the user hits the ENTER key
- Whatever the user typed in during that time is sent back to your program as a string. You can store that string in a variable. Example:

```python
user_age = input('How old are you?')
```
The Input function

- The `input()` function always “returns” a `String`.
- This means that the output it generates is a string, and when you store that output in a variable it will be treated as though it is a string.
The Input function

- Note that these two line work using the same mechanics:

```python
var1 = "Blue"
var2 = input("What is your favorite color?")
```

- In the first line of code we are assigning the String Literal into the variable ‘var1’
- In the second line of code we are assigning the RESULT of the input function into the variable ‘var2’
- Both ‘var1’ and ‘var2’ will be filled with data after these lines execute
Let’s give it a try …
Red, ______ apples! Today we are going to ______ apples. I am going to ______ the most.

My ______ and I are having an ______ picking contest this year. Every ______ we go to ______ farm to pick a ______ of apples. This year ______ wants to make ____________, so we need alot.

When we arrive ______, ______ counts out our apples. We anxiously await the final count.

My ______ and I ______! Well actually I had one more then him, but it had a ______ slimy worm.
_______ you____ and tastes like _______ubles_____. One day, I swear one of my meatballs started to _____rub______! The turkey tacos are totally____ squishy____ and look kind of like old ______mosquitos____. My friend Dana actually likes the meatloaf, even though it's _______funky____ and ______scary_____. I call it “mystery meatloaf” and think it's really made out of ______mothballs_____. My dad said he’d make my lunches, but the first day, he made me a sandwich out of ______eyelashes____ and peanut butter! I think I'd rather take my chances with the cafeteria!
Write a program that asks the user to type in 4 different words using the following prompts:

- enter a noun
- enter a verb
- enter an adjective
- enter an adverb

Use the input to output a “Mad Libs” paragraph using the following text:

The [adjective] [noun] was very hungry, so it decided to [adverb] [verb] to the nearest restaurant.
Performing Calculations
Performing Calculations

- Algorithms generally require some kind of calculation to be performed
- All programming languages have tools for manipulating numeric data – we generally call these “math operations”
<table>
<thead>
<tr>
<th>Operation</th>
<th>Operator</th>
</tr>
</thead>
<tbody>
<tr>
<td>Addition</td>
<td>+</td>
</tr>
<tr>
<td>Subtraction</td>
<td>-</td>
</tr>
<tr>
<td>Multiplication</td>
<td>*</td>
</tr>
<tr>
<td>Division (floating point)</td>
<td>/</td>
</tr>
<tr>
<td>Division (integer)</td>
<td>//</td>
</tr>
</tbody>
</table>
Expressions

- We use operators along with numeric data to create “math expressions” that Python can “evaluate” on our behalf
Expressions

5 + 2

7 - 6
2 * 3
6 / 2
6 + 2 - 12
Outputting math expressions

print (5+2)
print (100 * 5 - 12)
Storing the results of an expression

```python
answer = 5 + 2
print ('the answer to 5 + 2 is', answer)
```
Using variables in math expressions

- Math expressions don’t necessarily need to involve only numeric literals
- Example:

  ```
  price = 100.00
  sales_tax = 0.07
  
  total = price + sales_tax*price
  ```
Comments

- Commenting your code is EXTREMELY IMPORTANT.
- In the real world, multiple people work on projects, so programmers need to know what other team members were thinking when they were writing code a certain way.
- Scripts also get very large very quickly, and I swear you forget what you were thinking when you wrote some lines of code.
Starting now, the the first assignment, and for everything you ever do after, comment, comment, comment

These lines of code are not executed but seen in the source code explaining the code. A line of code is commented out with a ‘#’

#Variable to get users name
user_name = input("What’s your name?")
Comments

- If you want to comment multiple lines of code, you can use three apostrophes on each side

```
'''Below is a variable to get the user’s name so we can make mean statements to them'''
user_name = input("What’s your name?")
print(‘You smell, ‘, username, ‘!’)
```
Comments

- You NEED to comment your work for full marks this semester. The graders need to know what’s going on in your code. You can never over-comment (well, you can, if you did a paragraph for each line of code).

- As our programs get more complicated, logic, especially, absolutely needs to be commented
Work on Assignment #1