Introduction to Computer Programming

Software Development, Variables, Working with User Input, Math Operators & Data Types
Software Development

- Design
- Write
- Correct logic
- Test
- Correct syntax
BEFORE you start writing code you must first **Design** the program.

As programmers we must first establish a solid foundation before we begin coding.

That foundation involves us understanding the task that the program must perform.

Once we are clear on the task, we determine the steps that need to be taken in order to perform the task.
Well what is the Task?

We begin answering this by asking/interviewing our client/end user.

We, the programmers, must ask lots of questions and get as many details as possible about the task.

We might forget questions… and that is okay. We tend to have follow up sessions.

Once we have met with the client we generally construct a “software requirement” document.

This agreement between us and the client establishes what the program should actually do.
Understanding the Task
Understanding 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
Understanding the Task

Amtrak Case Study
Understanding the Task

Amtrak Case Study
Understanding the Task

Amtrak Case Study
Understanding the Task

Amtrak Case Study
Understanding the Task

Amtrak Case Study

Here's something to make your travel even more rewarding:

750 BONUS POINTS

Thanks for your continued loyalty. And feel free to keep shaking.
We're always keeping something fun under our hats.
Understanding the Task

Company will send out an e-mail that has a link to a specially designed e-card.

This e-card is designed to give only one type of prize, and this prize is set by the company. There can be multiple prizes.

When the card opens the user will be able to shake a hat.

When the hat shakes enough a prize will appear.
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!
Boiling Water... How do I tell a computer to do that?
Boiling Water... How do I tell a computer to do that?

- 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
What is an algorithm?

… hips don’t lie… and algorithms don’t either…

“Algorithms are a series of well defined, logical steps that must be taken in order to perform a task”

Algorithms serve as a necessary step between understanding the task at hand and translating that into computer code
Ahem... Pseudocode...

aka

Fake CODE

Pseudocode is a useful technique that we use to break down an algorithm into meaningful chunks and aligning these with the toolset of a language.

When we write pseudocode we don’t worry about syntax or spelling.

Instead we write pseudocode to create models or mock ups of the program. We focus on the design of the program when we write pseudocode.
Ahem... Pseudocode...

Amtrak Case Study

1. Start Program. You can assume you will be told what prize the user has won when the program begins.

2. Display a floating hat on the screen

3. Allow the user to click his or her mouse on the hat
   a. When clicked, drag the hat around with the user
   b. If the user releases the mouse button, snap the hat back into place and go back to step #3
   c. If the user vigorously moves the hat around, proceed to step #4

4. Stop dragging the hat around the screen

5. Display an animation that contains the prize amount specified in step #1 on the screen
Flowcharts are a graphical model that helps programmers understand the task at hand.
Flowcharts

Start

Input: Company provides prize level

Output: Display Floating Hat on Screen

Is user clicking on hat?

Did user release hat?

Did user shake hat?

Output: Display Prize

End
Understanding the Task

Sculpting With Data
Understanding the Task

Sculping With Data
Ahem... Pseudocode...

Sculpting With Data

1. Get the data
2. Clean the data
3. Decide what data streams to use vs. those you won’t
4. Define a material work area
5. Randomly map points from data stream 1 to an area that you predefined
6. Use numeric data from data stream 2 to copy and translate points from data stream 1 in the z axis
7. Generate circles that respond to data stream 3
8. Set the centroid of those circles to points from data stream 1
9. etc. etc. etc
What are a program's typical set of steps to perform?

INPUT - was it received?

PROCESSING - what is being performed?

OUTPUT - something that happens/something that is produced...
Inputs

Sources:

User - Keyboard, mouse, etc.

Hardware - Scanner, camera, etc.

Data - File, the Internet, etc.
Processing

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
Questions???
Python: Getting Started

Integrated DeveLopment Environment AKA IDLE *(I’m not pointless, Just easy to remember)*

Has two modes:

- Interactive Mode - Commands are written into IDLE and immediately processed as these are received by the shell.
- Script Mode - Allows us to write programs that we can reuse and process whenever we want - Yeah we do what we want...

We will mostly stick to using the script mode.

Side Note: Memorize the f5 key. I’ll likely use this to run our scripts in class. I use lots of shortcuts. I’ll try to mention these as I go along but if I forget feel free to ask me what I just did...
Python: A New Program

Open IDLE

Use keyboard shortcuts:
Command + N (mac) or Control + N (pc) OR move to/hover mouse on menu item labeled File and select New Window.

Go to File again and select save OR use keyboard shortcuts (SAVE TIME with shortcuts): Command + S/Control+S
Python: A New Program

OR you can slowly mouse over to the menu and select Run/Run Module

If your file is not open Command+O/CTRL+O etc.
Save your file somewhere on your computer. Preferably in a folder named after this class with a folder structure that captures a week by week list of folders. (Keep things tidy and organized - This will help me and our awesome tutors help you… Not to mention it will make your life easier… really it will)

When saving the file ensure that the file extension is set to .py
I can’t function in the mornings but Functions know what to do…

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 - “I’m so excited and I just can’t hide it”)

Functions always begin with a keyword followed by a series of parenthesis.

Ex: print ()
I can’t function in the mornings but Functions know what to do...

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 (‘‘‘‘)
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:

```
print ("Hello! My name is", "Ada")
```

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

Example:

print(‘a’, ‘b’, ‘c’, sep=‘∗’, end=‘’)
Variables

variables are moody... sometimes they are one thing other times they are another... they can hold numbers, strings, lots of different things... but boy are they variable
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:

```python
variablename = somedata
```

Examples:
```python
speed = 5
myName = 'Michell'
```

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

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
Variables

you CAN’T use the following reserved words/keywords 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'
Variables

```
variblename = 'Hello World'
```
Variables: Use Underscore

variable_name = ‘Hello World’
Variables: Use CamelCasing

variableName = 'Hello World'
... infringing on my patent
Legal or Illegal Variable Names

class = 2
class_avg = 125
classAvg = 99
_class_avg = 99
2ndclassavg = 125
classavg! = 99
Legal or Illegal Variable Names

- class = 2 - Illegal: “class” is a reserved word
- class_avg = 125 - Legal
- classAvg = 99 - Legal
- _class_avg = 99 - Legal
- 2ndclassavg = 125 - Illegal: can’t start with a number
- classavg! = 99 - Illegal: can only use alphanumeric values and the “_” symbol in variable names
Displaying Variables with Print()

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 = "Harry"
print ("Hello, my name is", name_var)
```

```
>> Hello, my name is Harry
```
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 = 10.99
print ('I have', dollars, 'in my account')
dollars = 29.99
print ('Now I have', dollars, 'in my account')
```
Multiple Variables

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:

\[
# a, b \text{ and } c \text{ will all contain the integer 100}
\]
\[
a = b = c = 100
\]
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
Reading input from Keyboard

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:

my_name = ‘Harry Potter’

print (‘Welcome to my program’, my_name)
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
user_name = input('What is your name?')

# welcome them!
print('Welcome, ', user_name)
```
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 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.
Note that these two lines 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.
Challenge

**Apples!**

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 allot.

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 ______ in it. That night we had ______ applesauce!
Our school cafeteria has really sleepy food. Just thinking about it makes my stomach squish. The spaghetti is red and tastes like sloshes. One day, I swear one of my meatballs started to rub! The turkey tacos are totally squishy and look kind of like old mosquitoes. 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 meatballs. 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!
Challenge

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.
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”
## Performing Calculations

<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>
We use operators along with numeric data to create “math expressions” that Python can “evaluate” on our behalf.
Outputting Math Expressions

```
print (5+2)

print (100 * 5 – 12)
```
Storing Results of Math Expressions

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
next steps:

begin “Self Paced Learning Module # 2”

work on Assignment #1: ‘Hello World!’