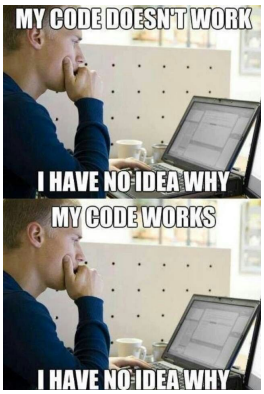
WEEK 1 & 2 REVIEW:



How to Submit Assignments (live demo)

1. Always ALWAYS put all your code in a .py file. You will get zero points if the file type is anything else because the TA cannot compile and run anything that's not a .py file
2. Find the .py file(s) in your computer directory and COPY (do NOT move) them into a new folder, properly named.
3. ZIP the folder
4. Submit ZIPped folder on BlackBoard

Variables

- Variables in python work the same way as they do in algebra
- A variable is a placeholder for a value
- All variables have "types." If a variable stores a number without a decimal, that variable's type is an int. If a variable stores letters, that variable's type is a string. If a variable stores a decimal, that variable's type is a float.
- You can name a variable anything you want, however, it is good coding practice to name the variable after what it actually stores (eg numOfPotatoes, userName, etc) and use camelCase (ie the first letter is lowercase, and then the subsequent first letters of each new word within the name is capitalized)
- A variable in python can store anything, however once you have given it an initial value of a certain type, it's best practice to always keep the variable that type. However, Python will allow you to change types freely

Exercises:

1. myValue = 50. What type is myValue? Is it good naming practice?
2. myName = "Brandon" What type is myName? Is it good naming practice?
3. Cost = 50.25. What type is cost? Is it good naming practice?

Input/Output

- How to get string input from user: userString = input("What is your string?") So, format: <var_name> = input(<prompt_for_user>)
- How to get integer input from user: userInt = int(input("What is your integer?")) {This is called integer casting}
- You ALWAYS need to store any input received from the user inside a variable. So any code that is just input("What is your string?") is invalid.

Exercises:

1. Ask the user for their name
2. Ask the user for their age using integer casting.

Type Conversions

If you are using the print statement such that you are using the "+" sign to concatenate your information, you **need** to make sure that any variables you are concatenating are **strings**. If the variable is already of type string, you're solid. Otherwise, you need to cast that variable as a string.

myName = "Brandon"
myAge = 22

Will this work?
print("Hello! My name is " + myName + " and my age is " + myAge)

No, the variable myAge is trying to mathematically add the string " and my age is " which is going to cause a type conversion error (more on this later). In order to avoid this, always ALWAYS cast any non-string variables to strings with the str() syntax.

Exercises:

Using the same code from the Input/Output section, output a sentence that uses both the user inputted name and age.

If Statements: The Review of the Review

Operator	Meaning	Sample Condition	Evaluates To
==	equal to	5 == 5	True
!=	not equal to	8 != 5	True
>	greater than	3 > 10	False
<	less than	5 < 8	True
>=	greater than or equal to	5 >= 10	False
<=	less than or equal to	5 <= 5	True

- 3 main different ways of doing an if statement
  - Just an "if" by itself. Typically used when the segment of code that changes is small and then the rest of the code continues as normal. Good for setting a default value (e.g. you ask the user for input, and they give invalid input, you can use a single if statement to set a default value)
  - If/else: only two choices. Different from single if in that each code block is significantly large. Mostly seen when a user is given a choice (yes or no, 1 or 2, etc)
  - If/elif/else: multiple choices. Most like the "Choose your Own Adventure Stories." A different piece of code executes for each option
- If a piece of code is not unique to an option, DO NOT put it in an if/else statement
- If/elif statements are **mutually exclusive**
- Logic in Python does not work the same way as logic in English language.
  - if a **and** b: The presence of the **and** indicates that BOTH a and b need to evaluate to true in order for the WHOLE if statement to be true. Example: if userInput1 == 5 and userInput2 != 6. In order for the entire if statement to be true, userInput1 needs to be 5 and userInput2 needs to be anything but 6.
  - if a **or** b: The presence of the **or** indicates that EITHER a or b needs to be true for the whole statement to be true.
    Example: if userInput1 == 5 or userInput2 != 6. In order for the entire if statement to be true, either userInput1 needs to be 5 OR userInput2 needs to be anything but 6. Both can true, but in order for the whole statement to be true, only one of them needs to be true

Exercise:

1. Ask the user for a number and store it in a variable named userNumber. If the user inputs a number less than zero, set userNumber equal to 0.
2. Print the menu "1) Print 'Hello World' 2) Print 'Python is awesome'" and ask the user for a number. Depending on the number they enter, print the corresponding message

3. Do the same as above but if they enter an invalid number print “Sorry, you didn’t enter a valid choice”

Math

Operator	Description	Example	Evaluates To
+	Addition	`7.0 + 3.0`	`10.0`
-	Subtraction	`7.0 - 3.0`	`4.0`
*	Multiplication	`7.0 * 3.0`	`21.0`
/	Division (True)	`7.0 / 3.0`	`2.333333`
//	Division (Integer)	`7.0 // 3.0`	`2.0`
%	Modulus	`7.0 % 3.0`	`1.0`
**	Exponent	`7.0 ** 3.0`	`2401.0`

Exercises:

For the following operations, write down your prediction for the result then code up the operation and see if you were right.

1. 5+4
2. 3-6
3. 10*50
4. 10**50
5. 5/2
6. 5.0/2
7. 5/2.0
8. 5//2
9. 5.0//2
10. 5//2.0
11. 5%2
12. 5.0%2
13. 5%2.0

Conditions

Operator	Meaning	Sample Condition	Evaluates To
==	equal to	5 == 5	True
!=	not equal to	8 != 5	True
>	greater than	3 > 10	False
<	less than	5 < 8	True
>=	greater than or equal to	5 >= 10	False
<=	less than or equal to	5 <= 5	True

A condition is an if/else statement. This is called branching. This is a useful coding technique if you want to change the actions of the program depending on the value of a variable. Comparisons can be made on both strings and ints/floats/doubles.

Exercise:
Ask the user for their name. If the name is the same as yours, print the statement "Yay! #twinning." Otherwise, print the statement "It's a trap!" (extra karma points if you get the reference)

Code Blocks
Code blocks refer to any code that is INDENTED beyond the normal indentation. Since we have just gone over conditionals, a code block refers to the code within an if statement and within and else statement.

Debugging demo time permitting