Module 3 - Conditionals

**1. Conditionals: Boolean Strings**

**1.1 Intro Python**

Jupyter Notebook: MOD03\_1-4.1\_Intro\_Python.ipynb

Conditionals

* **if, else, pass**
  + **Conditionals using Boolean String Methods**
  + Comparison operators
  + String comparisons

**Student will be able to**

* **control code flow with if... else conditional logic**
* **using Boolean string methods (.isupper(), .isalpha(), startswith()...)**

**1.2 if, else, pass**

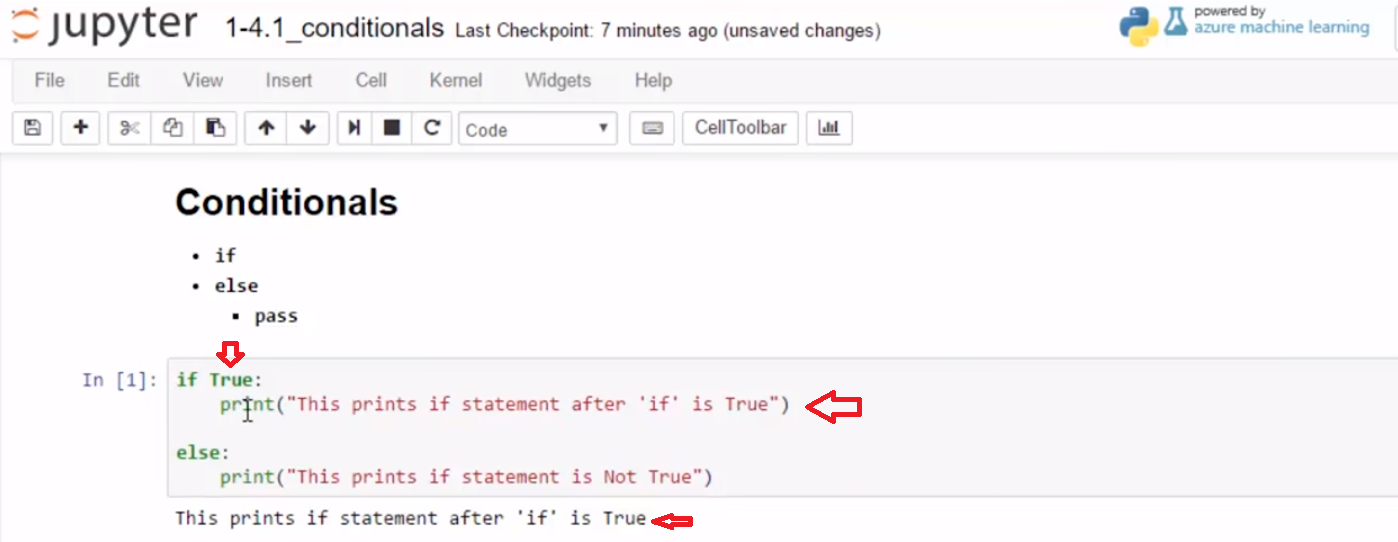
**Video:** **ifElsePassV1.mp4**

Concept: Boolean Conditional

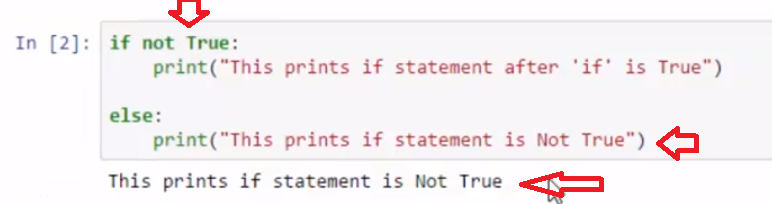
Conditionals use True or False

* **if**
* **else**
  + **pass**

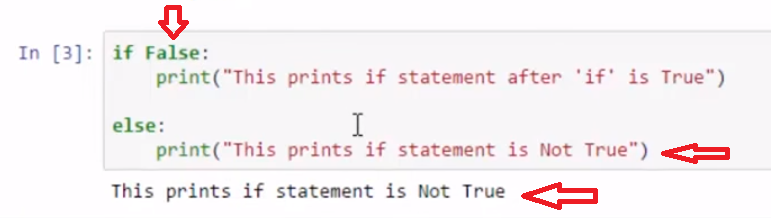
Conditionals help us run the flow of our code using conditional statements that resolve to true or false. **We will look at if, else, and pass statements**. >> The example has an if True statement. True is a Boolean keyword. **So if True will do whatever code is indented below the if statement. If it's not true, and in this case it's always true, so unless I write false, then we will do the else statement.** So let's just quickly look at how this works out when I run it. So if True, it never gets past if True. It prints this if statement after if is True. And the statement after if is True.



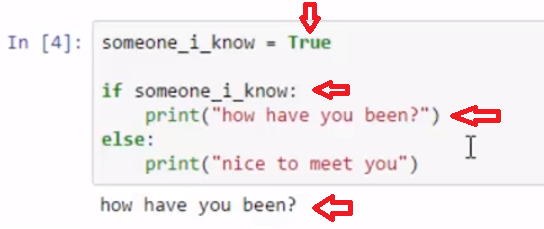
**To get to the else statement, we would have to have whatever's here be not True.** **And so we can actually use these keywords, not and True**. And then we see that the else statement will print when something's Not True.



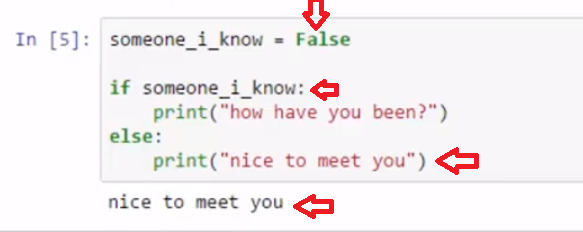
**The more typical way we think of Not True, is we think of it as False**. **And this is also the Boolean keyword, False.** So as different statements evaluate to True or False, we can execute either the if statement or the else.



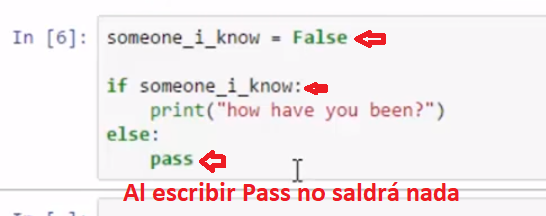
**A way to understand that a little more easily is we can set right here a variable to True or False.** And so, I have someone I know is True. And if someone I know is True then I'll say, how have you been? Else, if it's Not True, I'll print nice to meet you. And so I say how have you been?



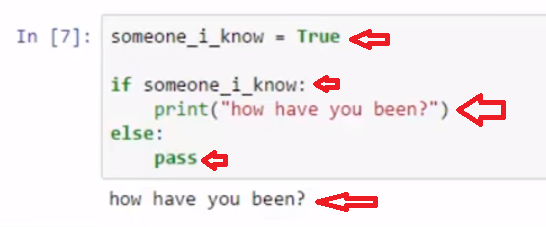
And then I could have something later where I determine that I don't know who that person is, and so is it someone I know? That is False. So I look here, if someone I know, if it's True or False, and I should see that nice to meet you. And it says nice to meet you.



**Sometimes though, when I run a if statement or an else statement, I really want only one of the conditions to execute.** What that means is I would only wanna print, how have you been or nice to meet you. **And so if I do not wanna do anything, I can just write the word pass. And this is a keyword, it turns green. And so nothing happens in this case.**



And then if I write True. Then it will run the how have you been path. So, if statements evaluate if they are True, and if Not True they will move to the else statement.



# Example

if True:

print("True means do something")

else:

print("Not True means do something else")

hot\_tea = True

if hot\_tea:

print("enjoy some hot tea!")

else:

print("enjoy some tea, and perhaps try hot tea next time.")

someone\_i\_know = False

if someone\_i\_know:

print("how have you been?")

else:

# use pass if there is no need to execute code

pass

# changed the value of someone\_i\_know

someone\_i\_know = True

if someone\_i\_know:

print("how have you been?")

else:

pass

# Task 1

## Conditionals

### Using Boolean with if, else

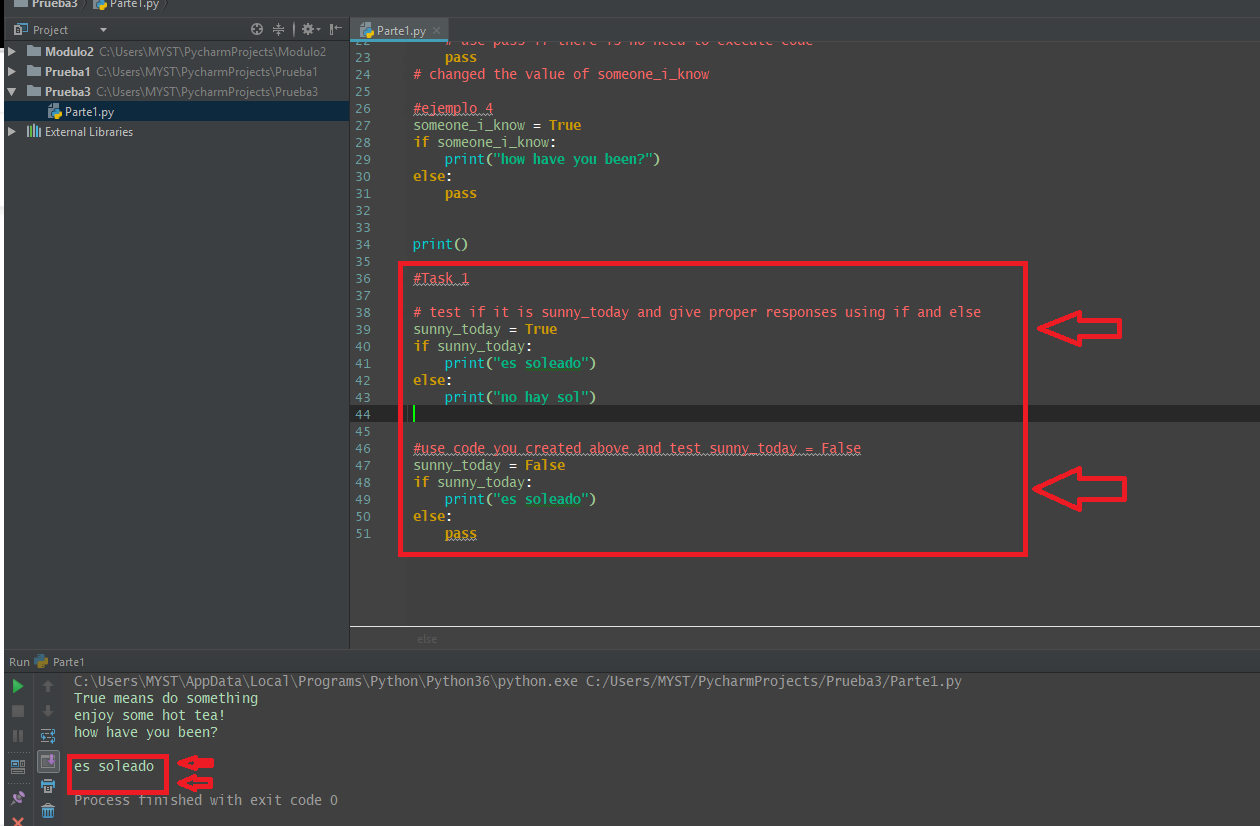
* **Give a weather report using if, else**

sunny\_today = True

# [ ] test if it is sunny\_today and give proper responses using if and else

sunny\_today = False

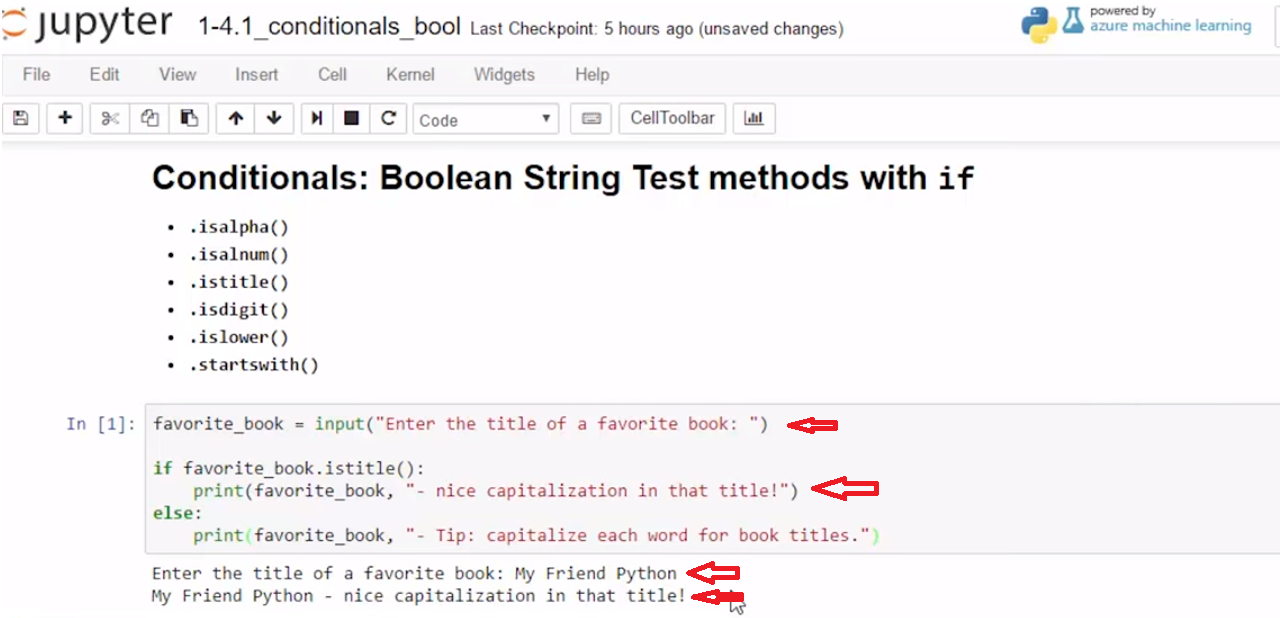
# [ ] use code you created above and test sunny\_today = False



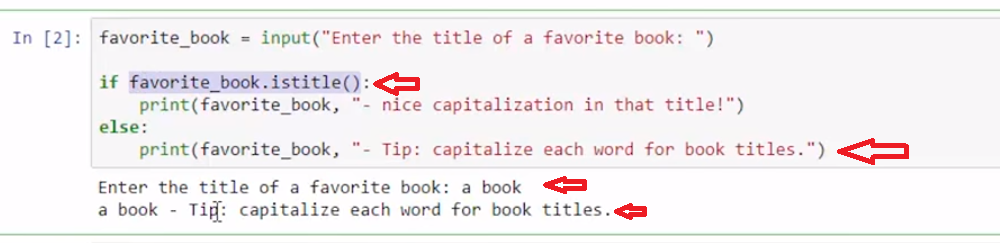
**1.3 Conditionals: Boolean String methods**

**Video:** **ConditionalsBooleanStringMethodsV2.mp4**

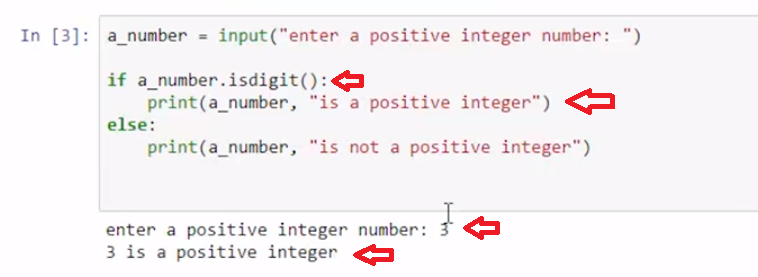
Boolean String Methods such as isalpha or isnum can be used to control our flow of code in an if statement. **We can use any Boolean String Method in an if statement because they all resolve to either true or false.** In our example, we get input for the title of a book and save that in favorite\_book. **Then we use the istitle method to see if it's a title case where every word is capitalized.** If that result is true, then we'll print nice capitalization. If not true, we'll go to the else statement and we'll give them a tip to capitalize each word. We run the code and I'll put in a title. **It says nice capitalization because I put in each word capitalized.**



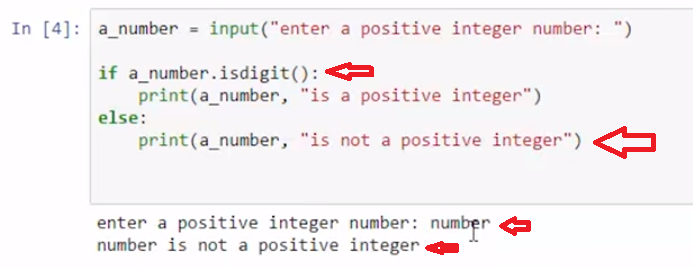
If I'll just put in **something without capitalization**, it gives me the tip. So this resolve do **not true**.



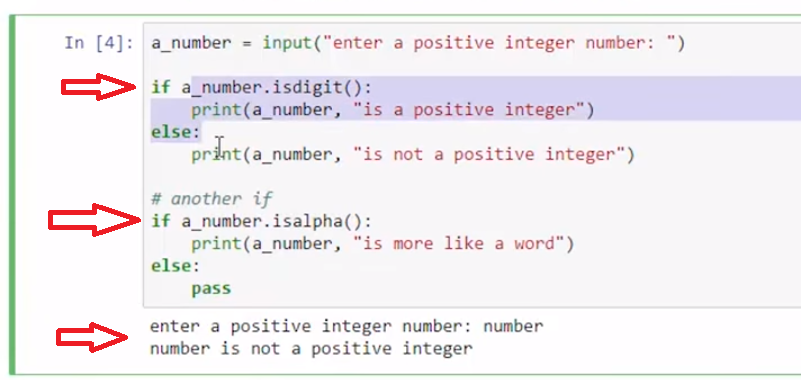
**In the next example, we want to enter a positive integer and then we run the check, is it a digit.** This is a good check because if we're **looking for a number input**, then if someone puts in something that's not a number or the positive digit we want,



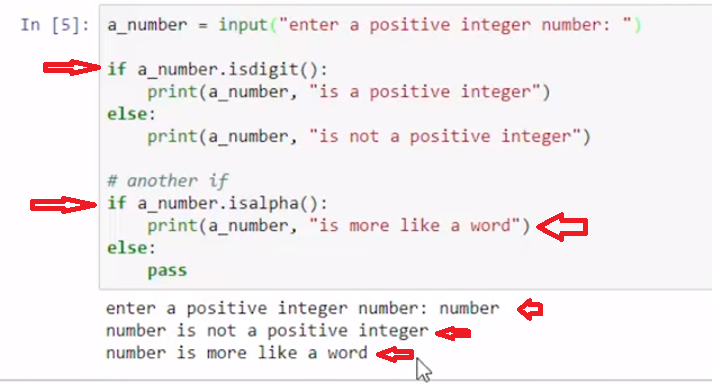
Then we see that a text string is not an integer.



**We can also run more than one if statement at a time. After this first if statement runs, I can run a second one to check if it's a alphabetical character.** And if it is, we can tell people that that's a word and that we're looking for a number input.



So I can try again. **And that it says number is not a positive integer, number is more like a word.**



So we can use our Boolean string test to evaluate if statements and control the flow of our code.

# Concept: Conditional with Boolean String Methods

## Using Conditionals with Boolean String test methods

**if student\_name.isalpha():**

* **.isalnum()**
* **.istitle()**
* **.isdigit()**
* **.islower()**
* **.startswith()**

# Example

# review code and run cell

favorite\_book = input("Enter the title of a favorite book: ")

if favorite\_book.istitle():

print(favorite\_book, "- nice capitalization in that title!")

else:

print(favorite\_book, "- consider capitalization throughout for book titles.")

# review code and run cell

a\_number = input("enter a positive integer number: ")

if a\_number.isdigit():

print(a\_number, "is a positive integer")

else:

print(a\_number, "is not a positive integer")

# another if

if a\_number.isalpha():

print(a\_number, "is more like a word")

else:

pass

# review code and run cell

vehicle\_type = input('"enter a type of vehicle that starts with "P": ')

if vehicle\_type.upper().startswith("P"):

print(vehicle\_type, 'starts with "P"')

else:

print(vehicle\_type, 'does not start with "P"')

# Task 2: multi-part

## Evaluating Boolean Conditionals

### create evaluations for .islower()

* print output describing **if** each of the 2 strings is all lower or not

test\_string\_1 = "welcome"

test\_string\_2 = "I have $3"

# [ ] use if, else to test for islower() for the 2 strings

# Task 2 continued...

### create a functions using startswith('w')

* w\_start\_test() tests if starts with "w"  
  **function should have a parameter for test\_string and print the test result**

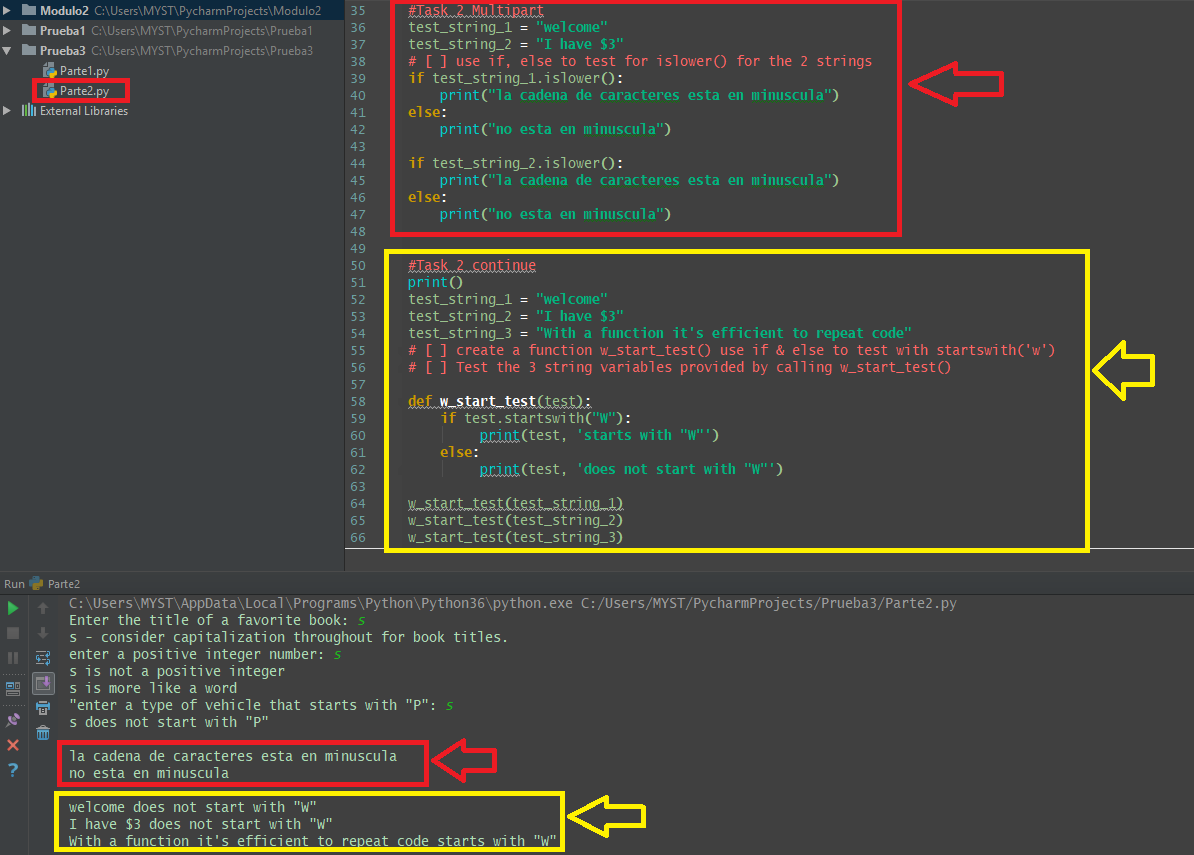
test\_string\_1 = "welcome"

test\_string\_2 = "I have $3"

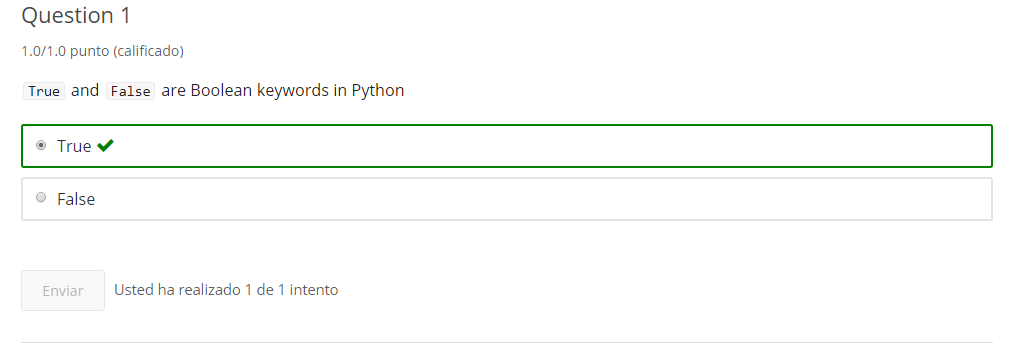
test\_string\_3 = "With a function it's efficient to repeat code"

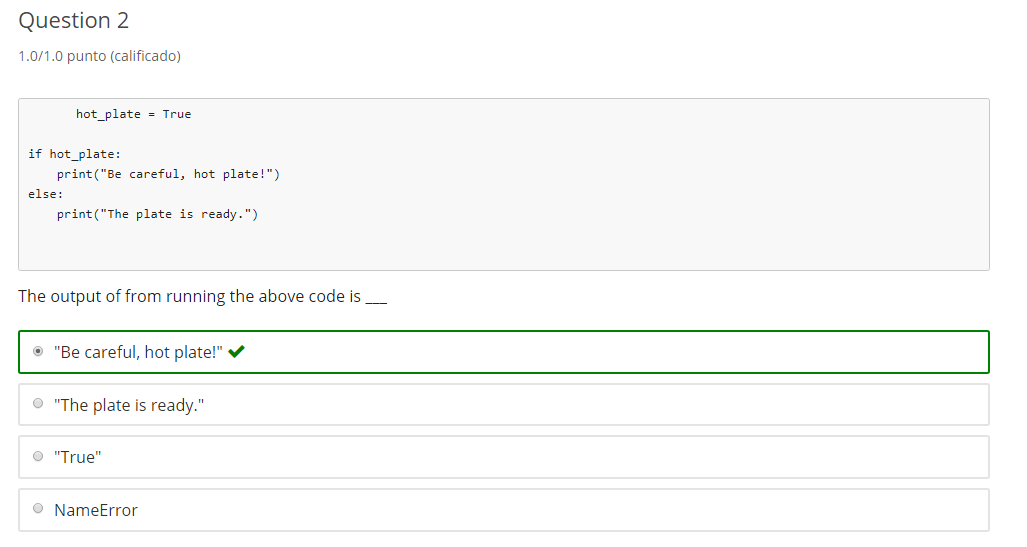
# [ ] create a function w\_start\_test() use if & else to test with startswith('w')

# [ ] Test the 3 string variables provided by calling w\_start\_test()



**1.4 Self-Check: Module 3 - Section 1**









**2. Conditionals: Comparison Operators**

**2.1 Intro Python**

Jupyter Notebook: MOD03\_1-4.2\_Intro\_Python.ipynb

Conditionals

* **if, else, pass**
  + Conditionals using Boolean String Methods
  + **Comparison operators**
  + String comparisons

**Student will be able to**

* **control code flow with if... else conditional logic**
* **using comparison (>, <, >=, <=, ==, !=)**

**2.2 Comparison Operators**

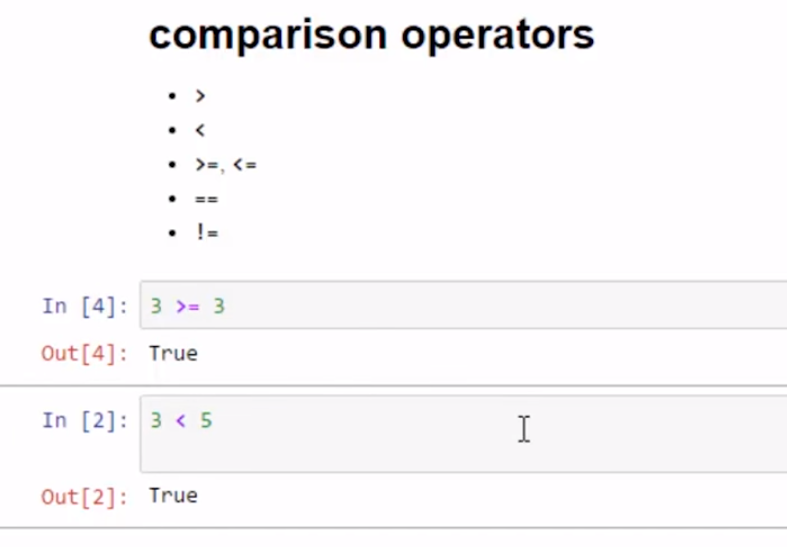
**Video:** **ComparisonOperatorsV3.mp4**

**Concept: Comparison Operators**

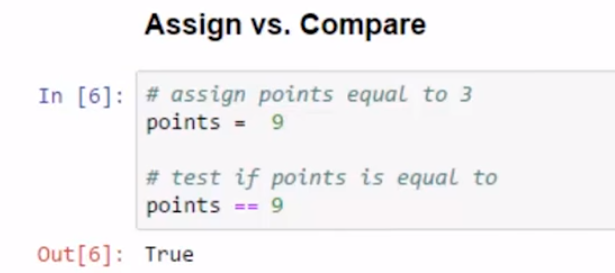
Comparison Operators

* **>**
* **<**
* **>=**, **<=**
* **==**
* Assign **=** vs compare **==**
* **!=**

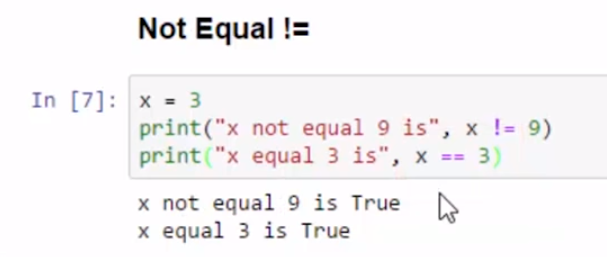
Comparison operators such as greater than, less than or equal to, can be used to evaluate to a true/false boolean. Let's take a look at each of the comparison operators Starting with >, 3 > 5 is False. 3 < 5 is True. 3 > 3 is False. So we can use a > or =. 3 > or = 3 is True. These work just like we've seen in our math class.



Assignment versus comparing. Here we're going to assign points the value of 0. The single equals sign is used for assignment. To compare, we use two equals signs. These points = to 9. That is false. **If we set the value to 9, we can see the comparison is now true.**



Not equal can be confusing when used in comparison. Here we're gonna assign x the value of 3. So the not equals sign is used here in this comparison in the print statement and it returns a boolean true or false. Is x != 9? Is x Equal to 3? Here we see x not equal to 9 is true because x is 3. And x equal to 3 is true.



So now we can run different comparison operators that return our boolean values.

# Example

# [ ] review and run code to see if 3 greater than 5

3 > 5

# [ ] review and run code to see if 3 less than or equal to 5

3 <= 5

# [ ] review and run code

# assign x equal to 3

x = 3

# test if x is equal to

x == 9

# [ ] review and run code

x = 3

print("x not equal 9 is", x != 9)

print("x equal 3 is", x == 3)

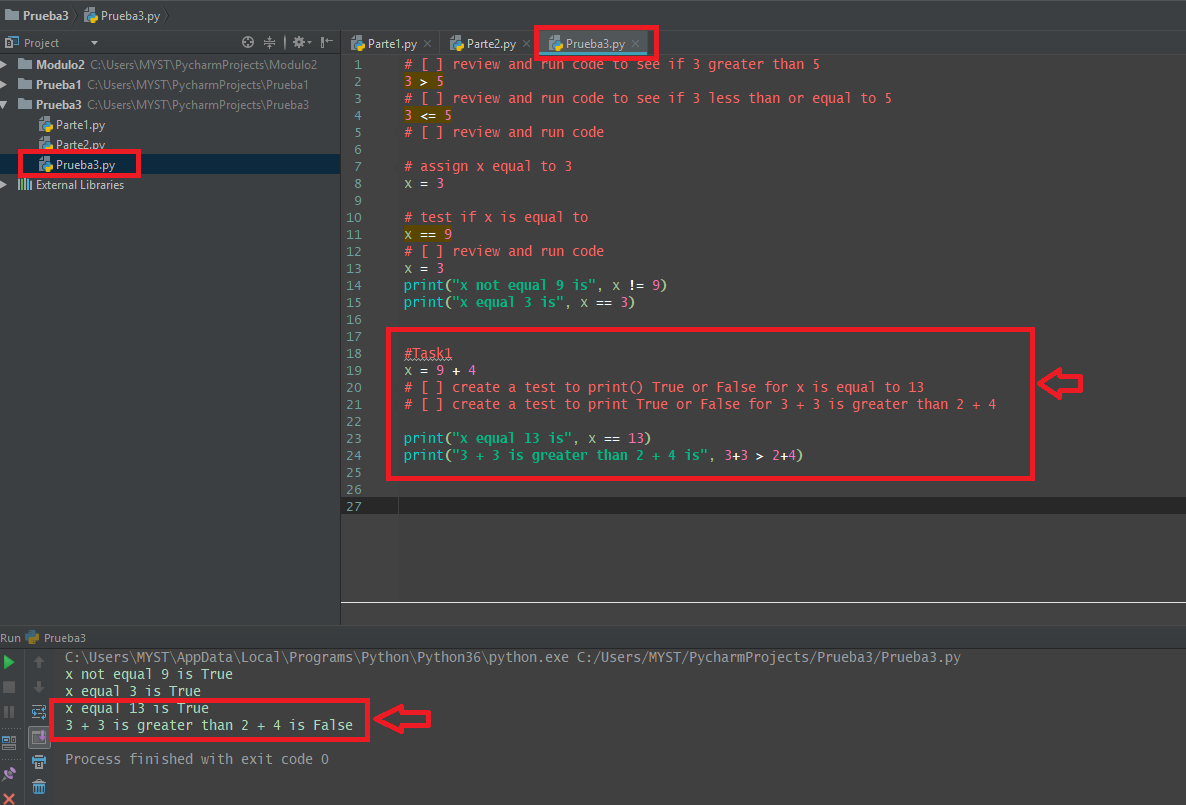
# Task 1

## comparison operators

x = 9 + 4

# [ ] create a test to print() True or False for x is equal to 13

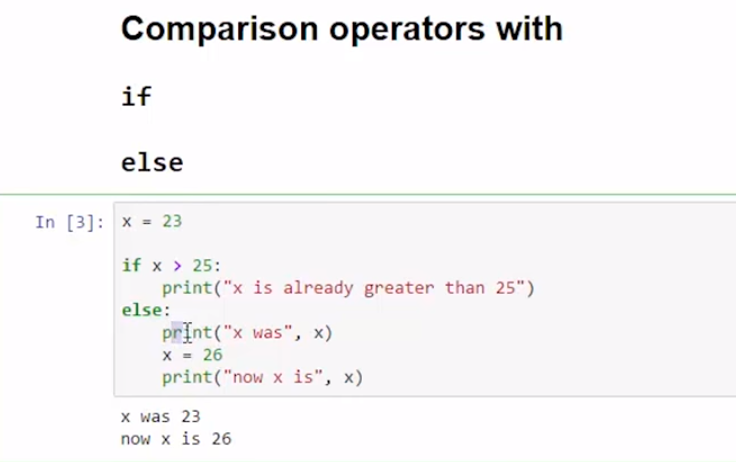
# [ ] create a test to print True or False for 3 + 3 is greater than 2 + 4



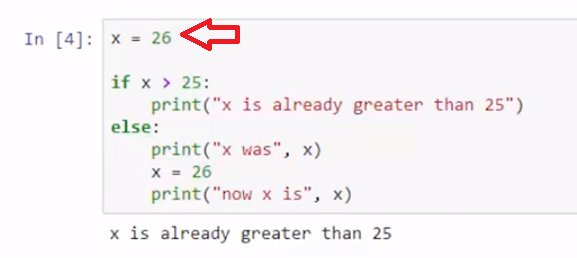
**2.3 if Comparison**

**Video:** **ifComparisonV4.mp4**

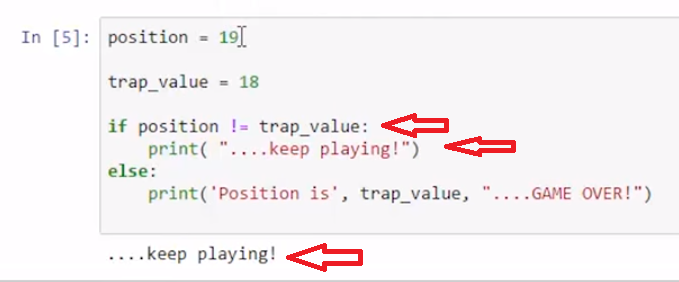
We use comparison operators with if else statements to help make decisions in code. Here, we have an example where we use a x greater than 25 test. If x is greater than 25, then we print x is already greater than 25. If it's not true, if x is less or equal to 25, then we'll execute our else statement and print what the value of x was. And then assign it a new value of 26, and inform the user. Let's run that code. So with x at 23, this evaluates to false or not true, else statement runs.



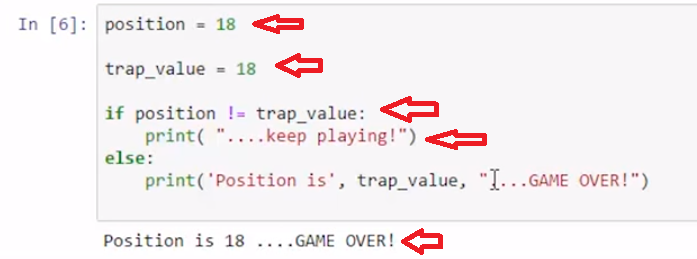
If we increase the value to larger than 25, sorry 26, then we see x is already greater than 25. We are using decisions in our code.



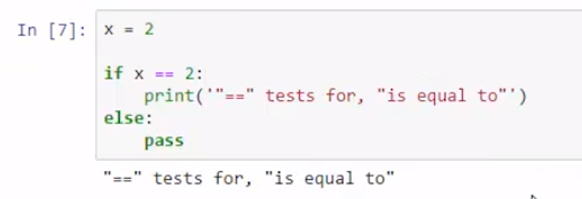
In this example, we have a gameboard where the player position and a trap position are set in the code. We do an if statement checking if we're not in the trap. **If we're not in the trap and that's true, we keep playing.** If we're not in the trap false, that means we're in the trap and then the game is over. So let's start with position of 19 and the trap at 18, and we can keep playing.



And then if I put myself in the trap at position 18, then my game is over.



The last example uses the equal comparison operator. First, I use the single equals sign to set the value of my variable x to 2. In my comparison statement, I just test if the value is equal to 2, and then I give the message about the double equals sign is the test for is equal to. I use a pass statement if this value is not true.



So we have seen how we can use boolean comparison operators in our if else statements, and it really does help us make decisions and make more interesting code.

# Concept: Comparison Operators in Conditionals

## Conditionals: comparison operators with if

Comparison operators evaluate to Boolean **True** and **False** to direct the flow of **if** conditionals

# Example

# review code and run cell

x = 21

if x > 25:

print("x is already bigger than 25")

else:

print("x was", x)

x = 25

print("now x is", x)

# review code and run cell

x = 18

if x + 18 == x + x:

print("Pass: x + 18 is equal to", x + x)

else:

print("Fail: x + 18 is not equal to", x + x)

# review code and run cell. "!" means "not"

x = 18

test\_value = 18

if x != test\_value:

print('x is not', test\_value)

else:

print('x is', test\_value)

# review code and run cell

# DON'T ASSIGN (x = 2) when you mean to COMPARE (x == 2)

x = 2

if x = 2:

print('"==" tests for, is equal to')

else:

pass

# Task 2

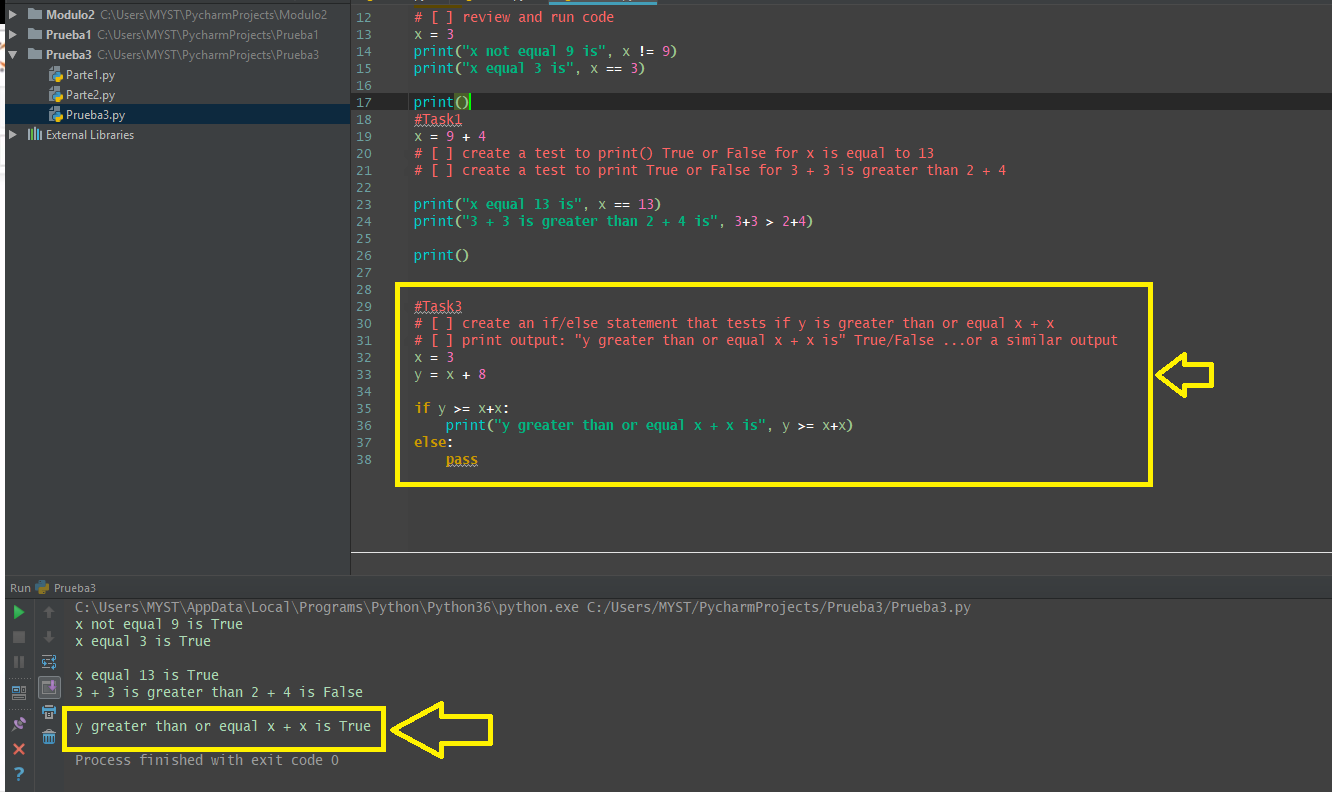
## Evaluating a comparison operator in if

# [ ] create an if/else statement that tests if y is greater than or equal x + x

# [ ] print output: "y greater than or equal x + x is" True/False ...or a similar output

x = 3

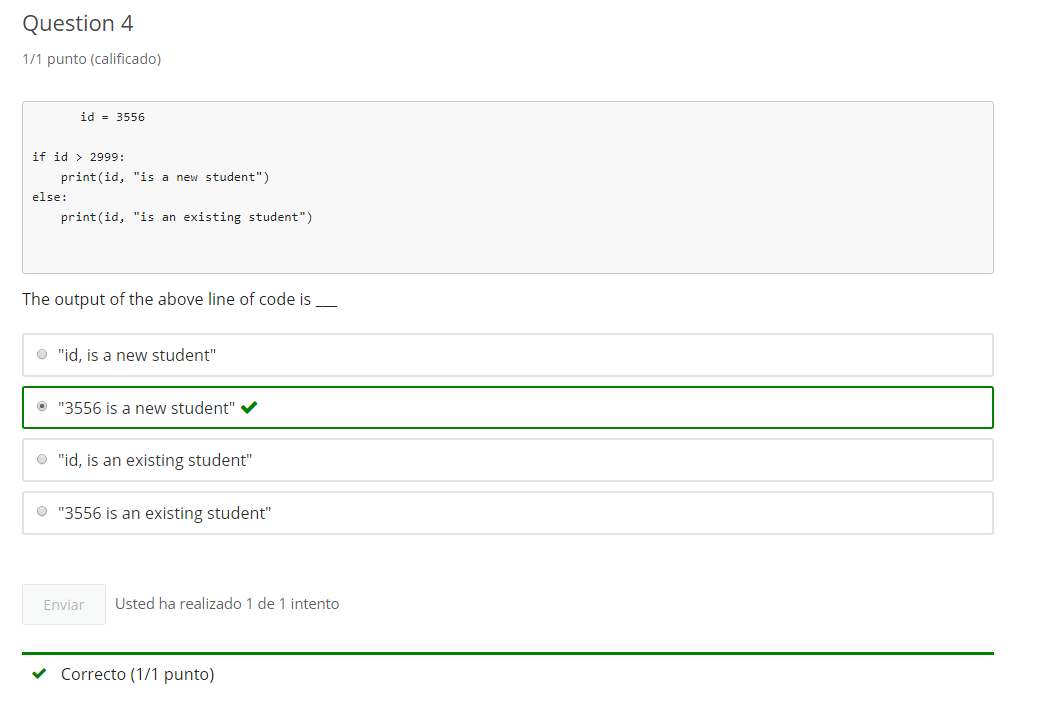
y = x + 8



**2.4 Self-Check: Module 3 - Section 2**







**3. String Comparison**

**3.1 Intro Python**

Jupyter Notebook: MOD03\_1-4.3\_Intro\_Python.ipynb

Conditionals

* **if, else, pass**
  + Conditionals using Boolean String Methods
  + Comparison operators
  + **String comparisons**

**Student will be able to**

* **control code flow with if... else conditional logic**
* **using Strings in comparisons**

**3.2 String Comparisons**

# Concept: String Comparison

## String Comparisons

### Strings can be equal == or unequal !=

### Strings can be greater than > or less than <

### alphabetically "A" is less than "B"

### lower case "a" is greater than upper case "A"

# Example

# review and run code

"hello" < "Hello"

# review and run code

"Aardvark" > "Zebra"

# review and run code

'student' != 'Student'

# review and run code

print("'student' >= 'Student' is", 'student' >= 'Student')

print("'student' != 'Student' is", 'student' != 'Student')

# review and run code

"Hello " + "World!" == "Hello World!"

# Task 1

## String Comparisons

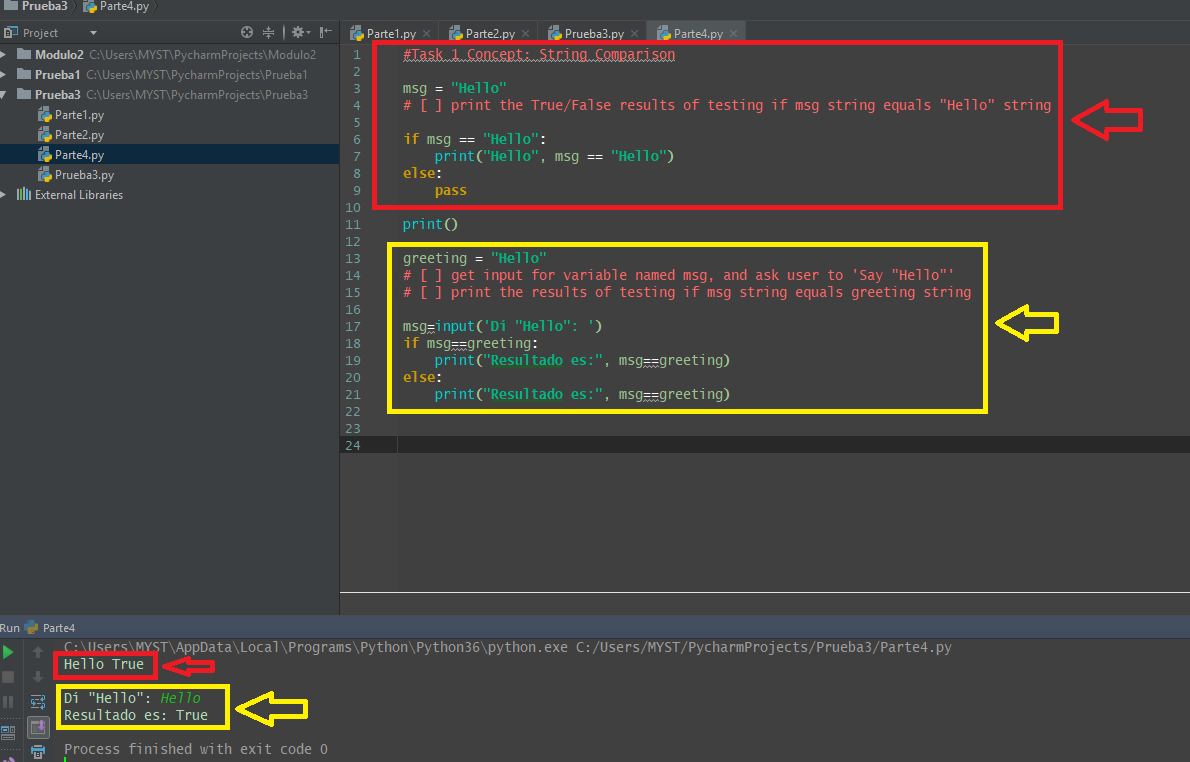
msg = "Hello"

# [ ] print the True/False results of testing if msg string equals "Hello" string

greeting = "Hello"

# [ ] get input for variable named msg, and ask user to 'Say "Hello"'

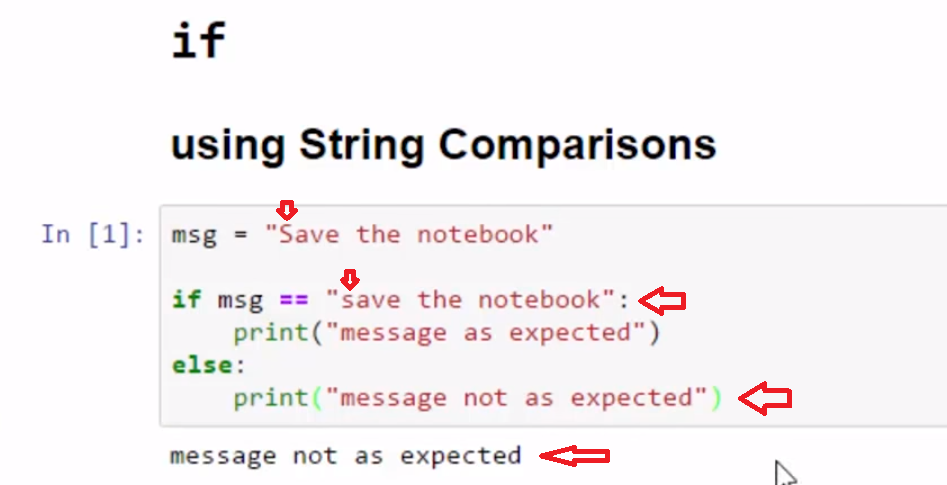
# [ ] print the results of testing if msg string equals greeting string



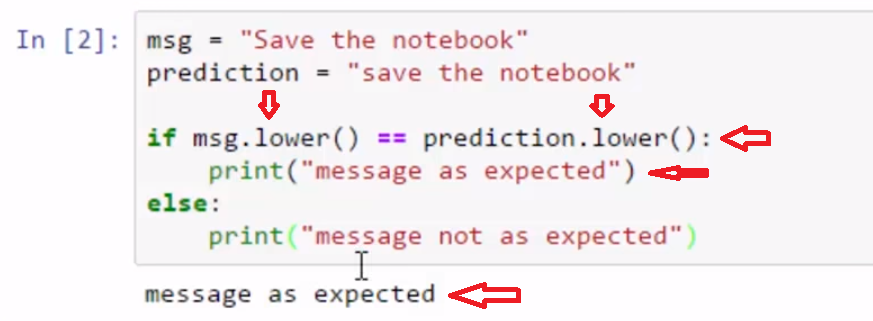
**3.3 String compare with "if"**

**Video:** **StringCompareWithIfV5.mp4**

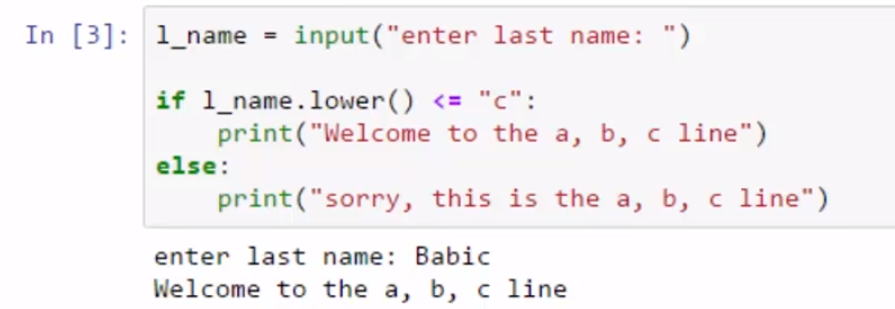
**String comparisons will resolve to true and false and therefore we can use them in an if statement to control the flow of our code.** In our first example, we have a string called msg and that contains Save the notebook. And we want to check if that message string is equal to this string literal Save the notebook. If that resolves to true, we're going to print the message as expected. If it resolves to false, then we'll go to our else statement. Print the message was not as expected. Let's run the code. So the message was not as expected, so that means this must have resolved to not true or false. **And we see that's because the capitalization of the two strings is different.**



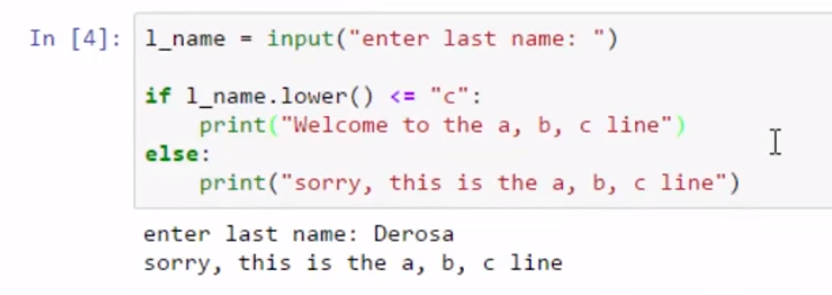
A more elegant way to write this code would be to test our message against a prediction. So both of these can use variables. And we're going to ignore the case. **So here we're going to put both sides of this comparison to lower. Regardless of what are entered for the message and the prediction the case will be ignored.** And so the same as before, we will print message as expected if the statement resolves to **true**. And if doesn't, we'll go to our else. If this is not true, and do message not as expected. **And this time, the message is as expected because we ignore the case.**



In this example, we're gonna input a name, a last name and check if it is alphabetically less than or equal to c. If you remember from our ASCII discussions before that capital C ends up being smaller than lowercase c, it's just the way ASCII works. So in order to do the check, we have to make sure the name is entered to lower. To do this check cuz otherwise a Z uppercase would be less than c and we would not be able to do this check. So we're gonna check against those and if this resolves to true we're gonna print Welcome to the a, b, c line. If the statement resolves to false, then we're going to tell the people sorry, this is only for a, b, c. So let's go ahead and run that code. And they ask us to enter the last name, we'll put in Babic and it says Welcome the the a, b, c line.



We could run that again. And we could put in Derosa.



We've seen using string comparisons with our if statements, we can control the flow of our programs.

**Concept: String Comparison with if**

Conditionals: String comparisons with if

**Examples**

# [ ] review and run code

msg = "Save the notebook"

if msg.lower() == "save the notebook":

print("message as expected")

else:

print("message not as expected")

# [ ] review and run code

msg = "Save the notebook"

prediction = "save the notebook"

if msg.lower() == prediction.lower():

print("message as expected")

else:

print("message not as expected")

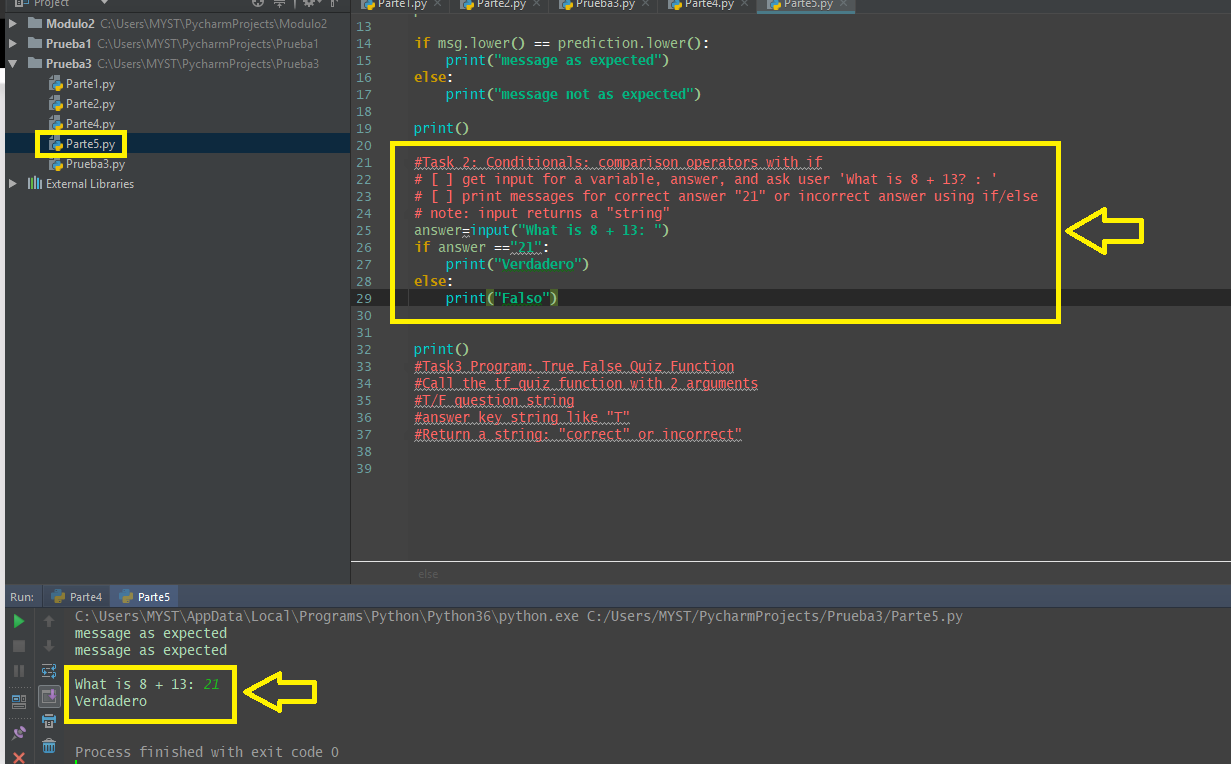
**Task 2**

Conditionals: comparison operators with if

# [ ] get input for a variable, answer, and ask user 'What is 8 + 13? : '

# [ ] print messages for correct answer "21" or incorrect answer using if/else

# note: input returns a "string"



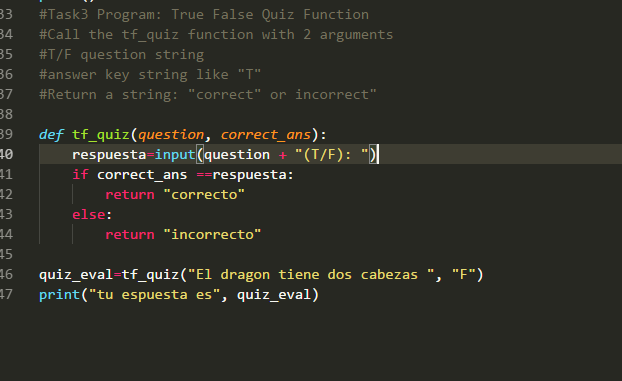
**Task 3**

Program: True False Quiz Function

Call the tf\_quiz function with 2 arguments

* T/F question string
* answer key string like "T"

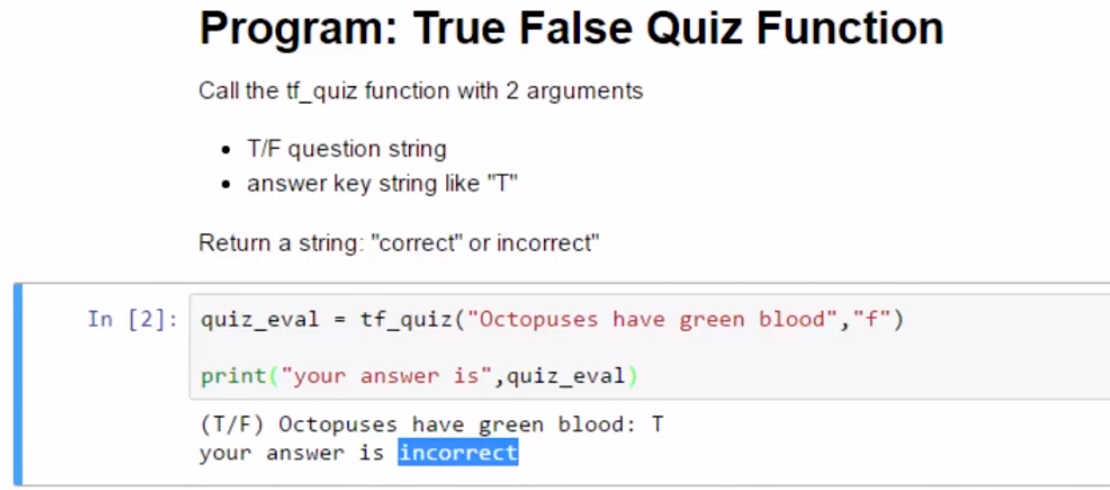
Return a string: "correct" or incorrect"



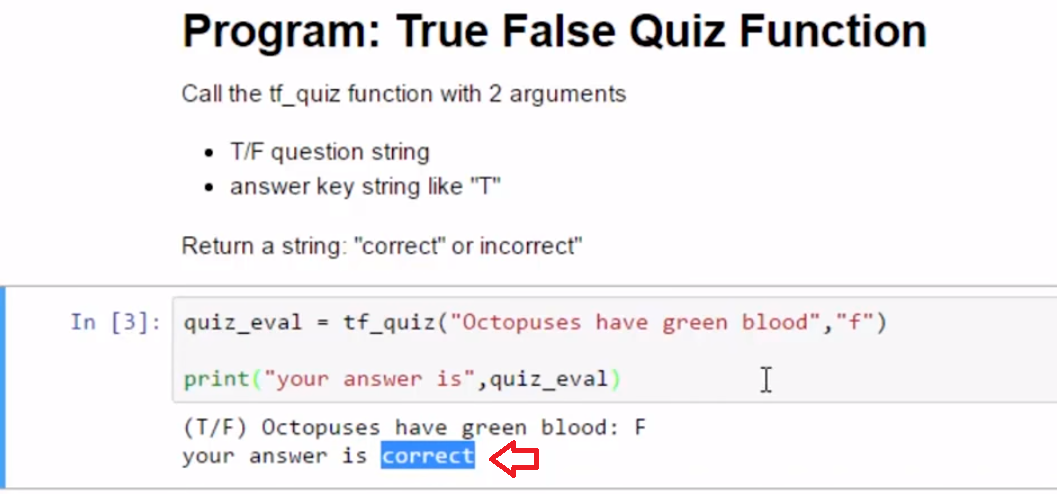
### Return a string: "correct" or incorrect"

**Video:** **ReturnStingCorrectOrIncorrectV6.mp4**

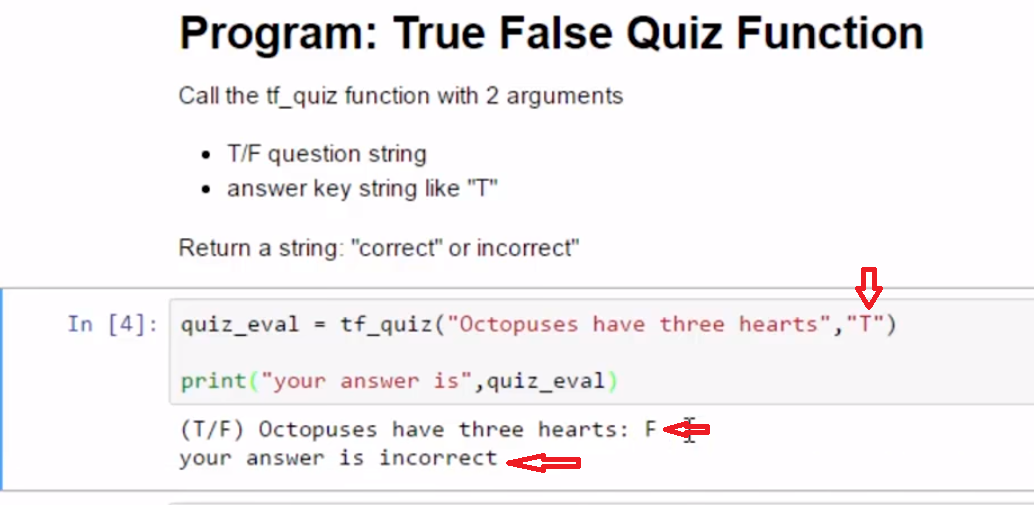
Here are some tips on the true and false quiz. You should try to complete it without watching first. The true false quiz has two parts. We're gonna create the function, and we're going to call the function with a true false question statement and the answer. So I'm going to show you how to call the true false quiz and you will have to write the function still. Calling true false quiz or tf\_quiz, I'm going to send two arguments. And so you'll have to create these parameters in the function. The first is a statement, octopuses have green blood, is my question statement, and the second argument is the answer which is that is not true because octopuses have blue blood. And then I'm just going to print out the answer that comes back so let's run that code. And we see, I can put in there that it's true and it says, your answer is you can see it's gonna send back correct or incorrect from the function call.



So we can try that again. You can test it with false, and it says my answer is correct.



I could give it another question. Octopuses have three hearts. And we know that's true. So let's go ahead and call the function again. And then you see, it's echoing out the new statement octopuses have three hearts. And maybe I think they have five and so I put in false and it tells me my answer is incorrect.

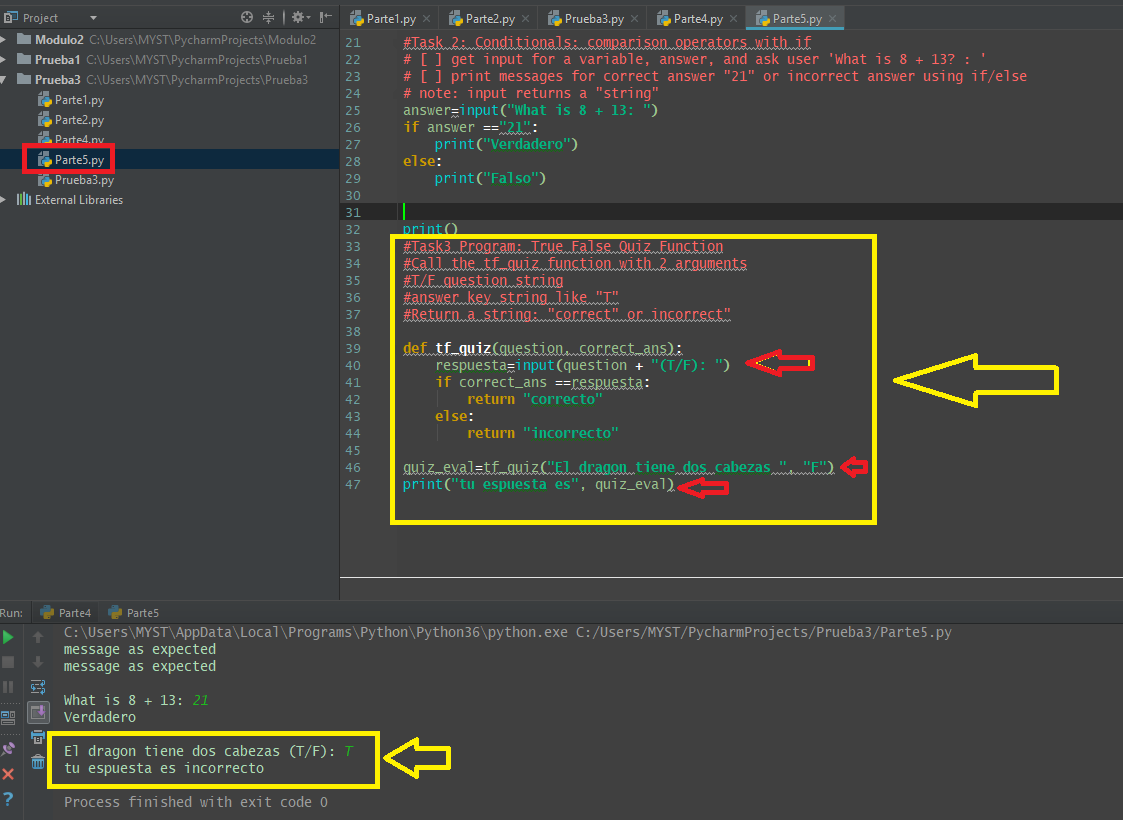


So now go ahead and write that function.

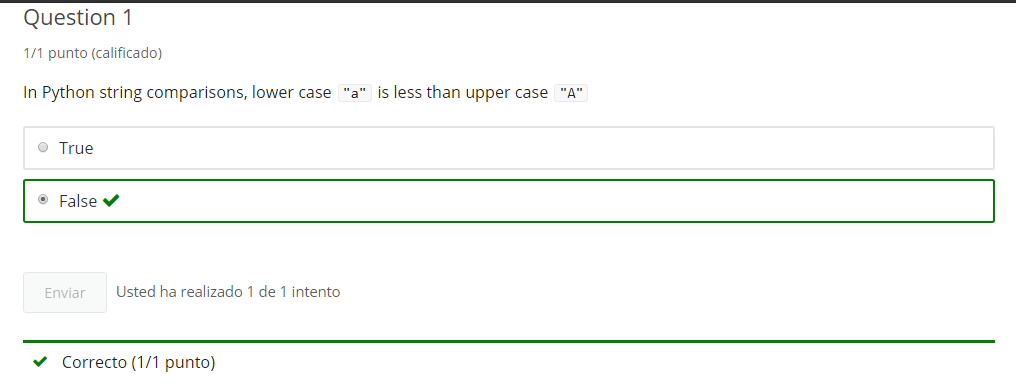
### Define and use tf\_quiz() function

* **tf\_quiz()** has **2 parameters** which are both string arguments
  + **question**: a string containg a T/F question like "Should save your notebook after edit?(T/F): "
  + **correct\_ans**: a string indicating the correct answer, either **"T"** or **"F"**
* **tf\_quiz()** returns a string: "correct" or "incorrect"
* Test tf\_quiz(): **create a T/F question** (or several!) to **call tf\_quiz()**

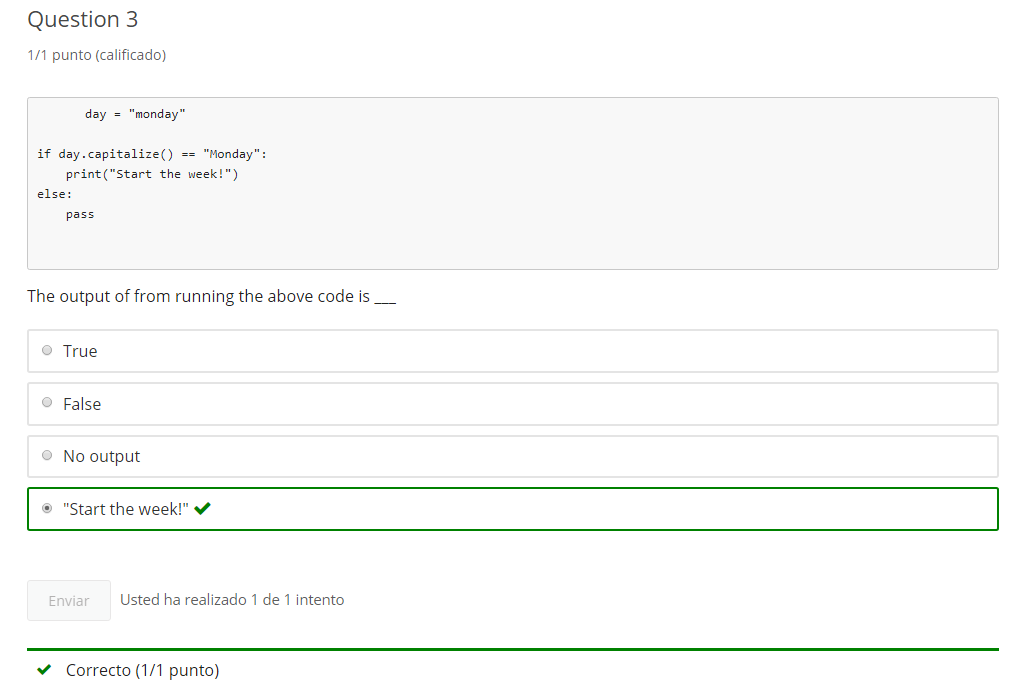
# [ ] Create the program, run tests



**3.4 Self-Check: Module 3 - Section 3**









**4. Conditions elif and casting**

**4.1 Intro Python**

Jupyter Notebook: MOD03\_1-5.1\_Intro\_Python.ipynb

conditionals, type, and mathematics extended

* **conditionals: elif**
* **casting**
* basic math operators

**Student will be able to**

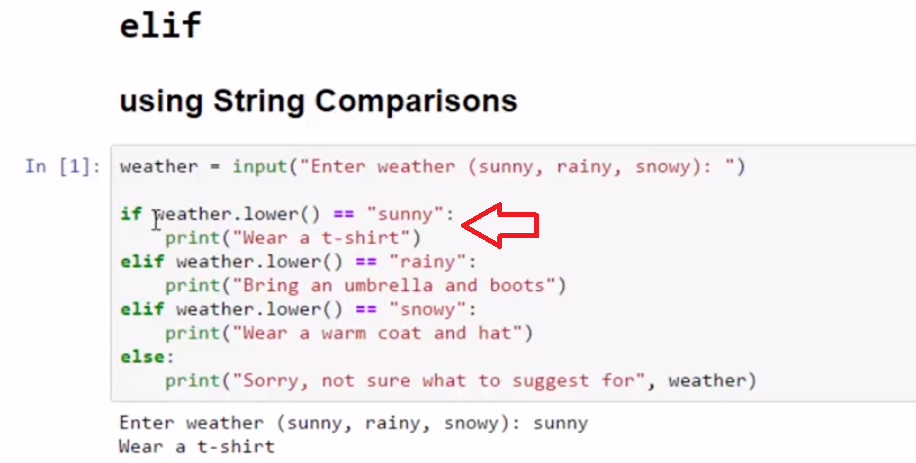
* **code more than two choices using elif**
* **gather numeric input using type casting**

**4.2 elif**

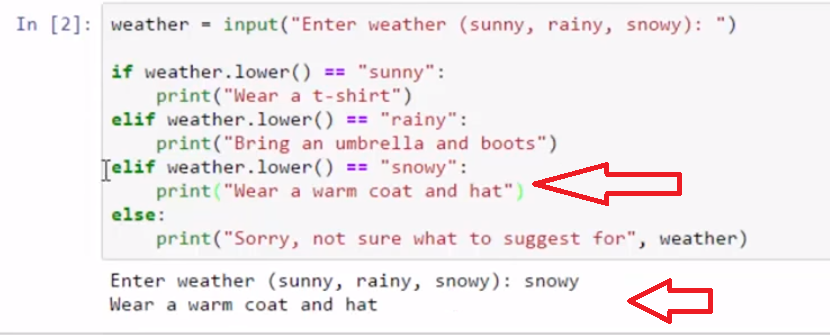
**Video:** **elifV7.mp4**

**Do you know what an elif statement is? Elif in Python is an else if statement and it allows us to test for multiple conditions.**

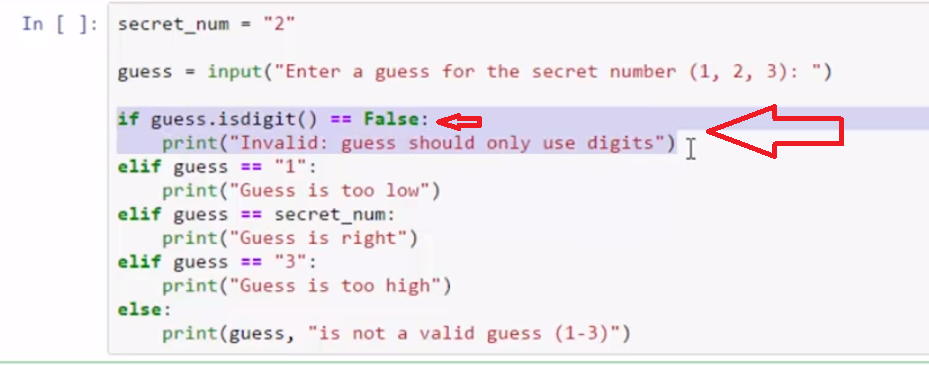
**Elif in this example will allow us to check for four different cases.** Whether the weather is sunny, rainy, snowy, or if we put in another entry that we don't understand. Let's look at the example in the very simplest case with just the if statement. If the weather, this input that we got right here, is equal to sunny, then we're gonna print wear a t-shirt. If we wanna check for rainy, else would not work for us, cuz it could also be snowy entered, or some invalid input. So we use the elif statement. This first elif checks if it's raining. The second elif checks if its snowy. Then finally we get to our else statement, where its none of the three cases we checked for and we just say we don't understand the message. Let's run the code. I'll try sunny first. It says to wear a t-shirt, so we went through this path.



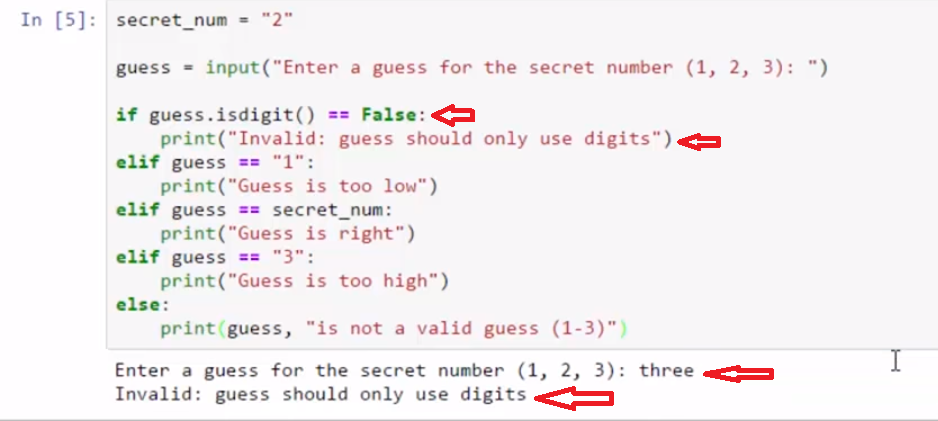
Let's run it again. It's snowy, we went through this path. This was not true, if statement. The elif statement was not true, rainy. **The elif statement for snowy was true, and we printed our message.**



So we can control the flow through many paths using elif. The other example's very similar. It just uses the elif statement to see if we're guessing a number properly. So we put in a secret number, and then we tell the individuals to guess 1, 2, or 3. **If its not a digit, then we say, they need to put in a digit. So this case checks if it's a letter or some other non digit. If they guess 1, we tell them they guessed too low.**



If they guess the secret number, we tell them that it's right. If they guess 3, we tell them too high. And otherwise, we say it's not a valid guess in the range. Let's run the code. **If we enter a typed, "Three" in letters, we should get the message that it's a invalid guess. And we do. And that means this statement here executed. It checked if it's a digit, and because if .isdigit was false, then that executes it matched. .isdigit()== False;. Yes, then run this code.**



Let's try another test. **Here we gotta to run the third elif statement**.



Using the elif statement, we can check for multiple conditions.

# Concept: Conditional elif

## Using the Conditional elif

**Review:** if and else

* **if** means "**if** a condition exists then do some task." **if** is usually followed by **else**
* **else** means "**or else** after we have tested **if**, then do an alternative task"

When there is a need to test for multiple conditions there is **elif**

* **elif  statement follows  if**, and means **"else, if "** another condition exists do something else
* **elif**  can be used many times
* **else  is used after the last test condition (if** or **elif**)

#### in pseudo code

**If** it is raining bring an umbrella  
or **Else If**  (elif) it is snowing bring a warm coat  
or **Else** go as usual

Like **else**, the **elif** only executes when the previous **if** conditional is False

# Example

# [ ] review the code then run testing different inputs

# WHAT TO WEAR

weather = input("Enter weather (sunny, rainy, snowy): ")

if weather.lower() == "sunny":

print("Wear a t-shirt")

elif weather.lower() == "rainy":

print("Bring an umbrella and boots")

elif weather.lower() == "snowy":

print("Wear a warm coat and hat")

else:

print("Sorry, not sure what to suggest for", weather)

# [ ] review the code then run testing different inputs

# SECRET NUMBER GUESS

secret\_num = "2"

guess = input("Enter a guess for the secret number (1-3): ")

if guess.isdigit() == False:

print("Invalid: guess should only use digits")

elif guess == "1":

print("Guess is too low")

elif guess == secret\_num:

print("Guess is right")

elif guess == "3":

print("Guess is too high")

else:

print(guess, "is not a valid guess (1-3)")

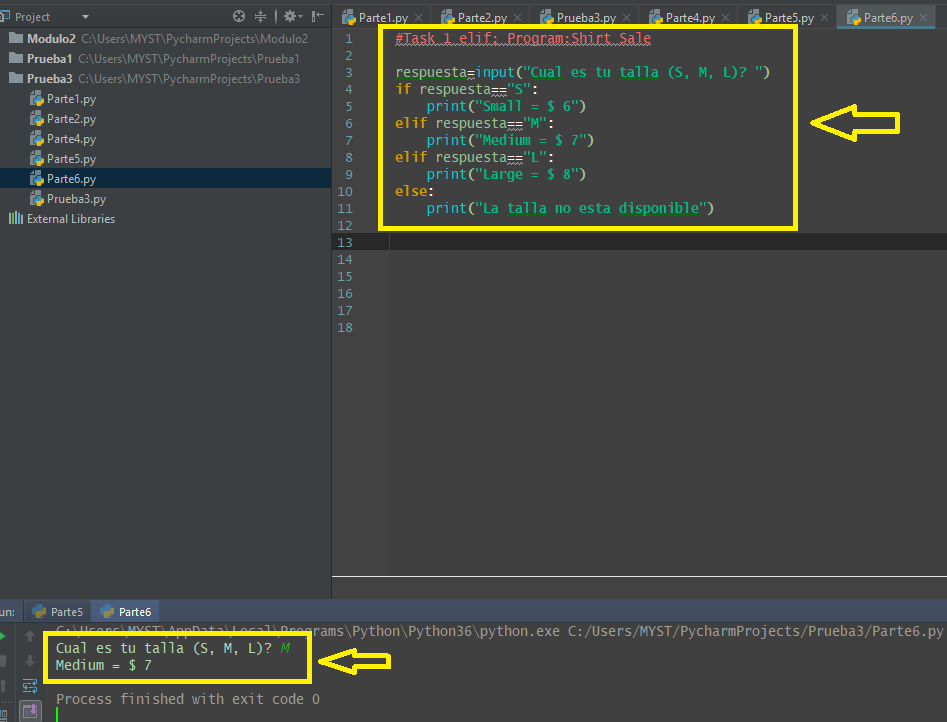
# Task 1

## Program: Shirt Sale

### Complete program using   if, elif, else

* Get user input for variable size (S, M, L)
* reply with each shirt size and price (Small = $ 6, Medium = $ 7, Large = $ 8)
* if the reply is other than S, M, L, give a message for not available
* optional: add additional sizes

# [ ] code and test SHIRT SALE



**4.3 Casting**

**Video:** **CastingV8.mp4**

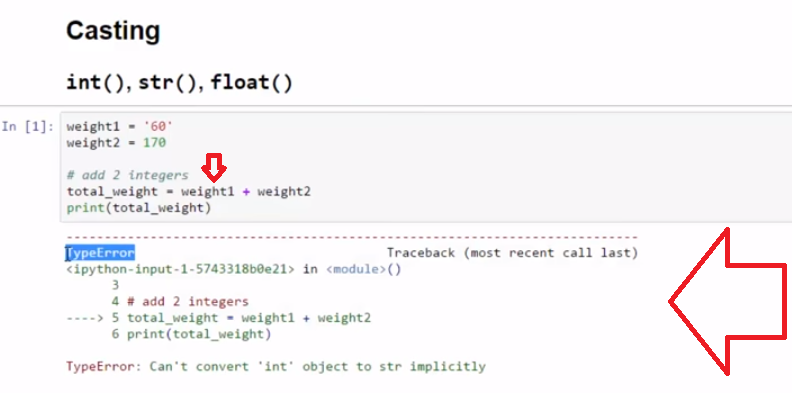
# Concept: Casting

## Using casting to change data type

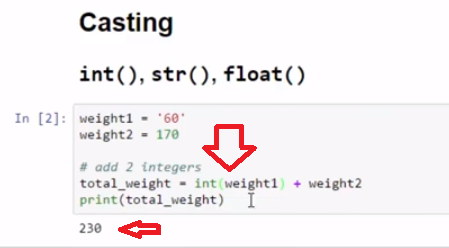
Casting is the conversion from one data type to another Such as converting from **str** to **int**

**Casting input is a common task.** **Input always is a string** **and so it's important sometimes that we are able to change it into a float, or an integer, or back to a string.** >> **Casting is when we force a type upon an object.** **We're gonna look at three different types of casts. Int, forces something to become an integer, str is a string cast, and then float.**

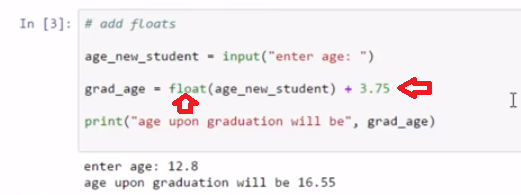
Let's look at this example where we have two weights. One is a string, and the second one is an integer. When we add numbers, they can be floats and integers but not strings. In this case, if we run the code, **we get the familiar TypeError that we saw earlier when we did string concatenation and try to add an integer to a string. It's the same error, here,**



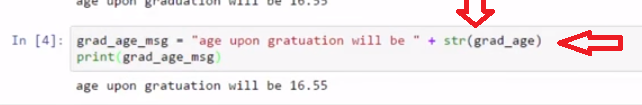
**we'll need weight1, the string, to become an integer. Cast it by typing the keyword int in front of the object and place it in parenthesis.** **Now, 60 changes from digits that are string to numbers. And we add and get our result 230.**



In this example, we're going to calculate the graduation age of a student. First, let's get some input. Age of the new student. **Input is always a string.** **So I'm going to take that string input and add it to 3.75 to calculate the graduation age.** **But I'll need the age of the new student to be a float value, so in order to cast that to a float in parenthesis, I'm putting the string that I want to cast and preceeding it by the keyword float.** This allows me to add the input together. Now the numbers will add up. So we see this grad\_age is the 16.55, it's a float.



For my graduation message, I want to string concatenation and add this age upon graduation will be plus the 16.55. **In order to avoid the type error, I'm going to change grad\_age into a string.** In order to print it, I'll just create a print statement.



We can avoid the type error by casting to int, string, and float.

### int()

the **int()** function can convert stings that represent whole counting numbers into integers and strip decimals to convert float numbers to integers

* int("1") = 1 the string representing the integer character "1", cast to a number
* int(5.1) = 5 the decimal (float), 5.1, truncated into a non-decimal (integer)
* int("5.1") = ValueError "5.1" isn't a string representation of integer, int() can cast only strings representing integer values

# Example

weight1 = '60' # a string

weight2 = 170 # an integer

# add 2 integers

total\_weight = int(weight1) + weight2

print(total\_weight)

# Task 2

## casting with int() & str()

str\_num\_1 = "11"

str\_num\_2 = "15"

int\_num\_3 = 10

# [ ] Add the 3 numbers as integers and print the result

str\_num\_1 = "11"

str\_num\_2 = "15"

int\_num\_3 = 10

# [ ] Add the 3 numbers as test strings and print the result

# Task 2 cont...

### Program: adding using int casting

* **[ ]** initialize **str\_integer** variable to a **string containing characters of an integer** (quotes)
* **[ ]** initialize **int\_number** variable with an **integer value** (no quotes)
* **[ ]** initialize **number\_total** variable and **add int\_number + str\_integer** using **int** casting
* **[ ]** print the sum (**number\_total**)

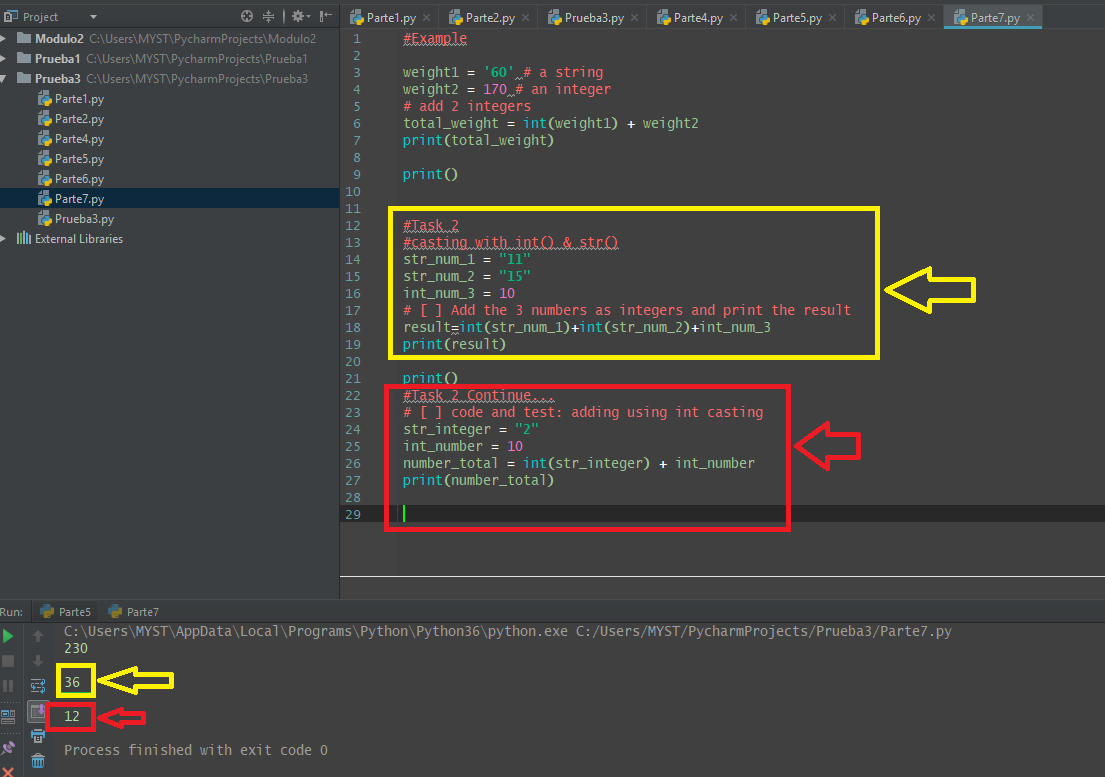
# [ ] code and test: adding using int casting

str\_integer = "2"

int\_number = 10

number\_total = int(str\_integer) + int\_number

print(number\_total)



**4.4 Casting input**

**Concept: Casting Numeric Input**

Casting input() strings that represent numbers to integer values

**Example**

# [ ] review and run code

student\_age = input('enter student age (integer): ')

age\_next\_year = int(student\_age) + 1

print('Next year student will be',age\_next\_year)

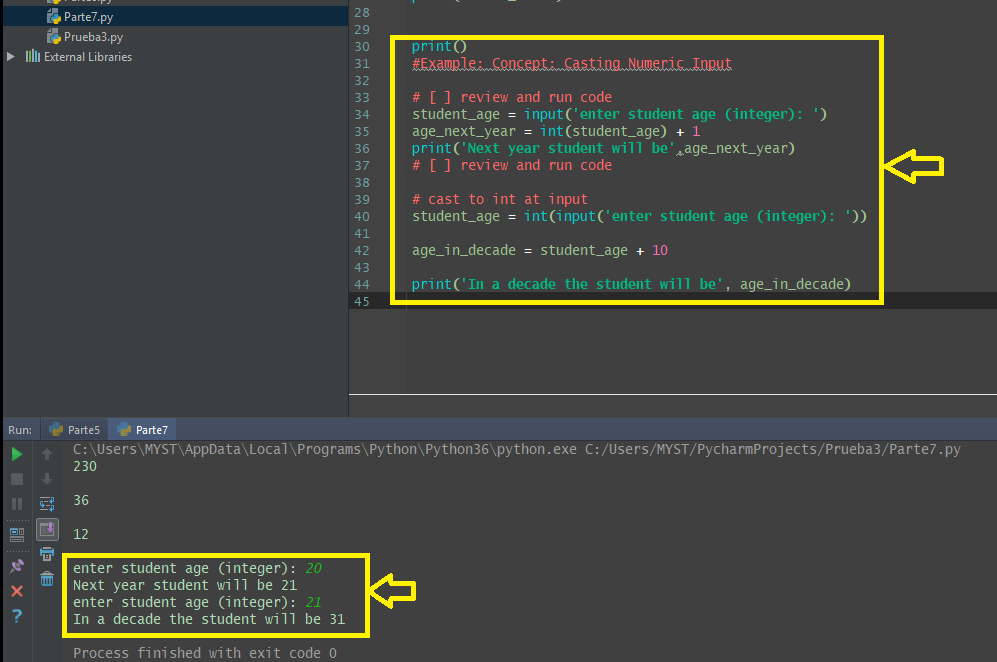
# [ ] review and run code

# cast to int at input

student\_age = int(input('enter student age (integer): '))

age\_in\_decade = student\_age + 10

print('In a decade the student will be', age\_in\_decade)



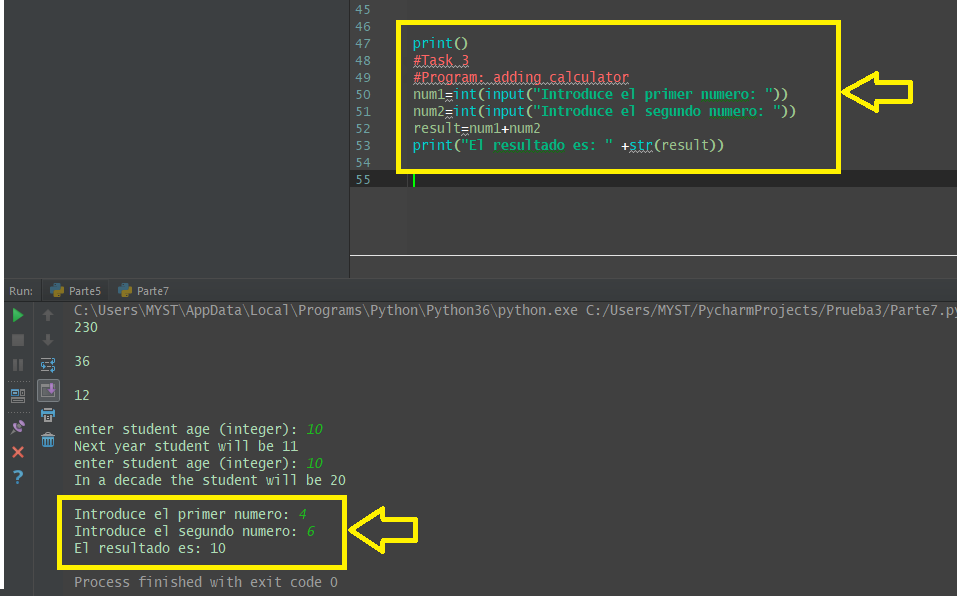
**Task 3**

Program: adding calculator

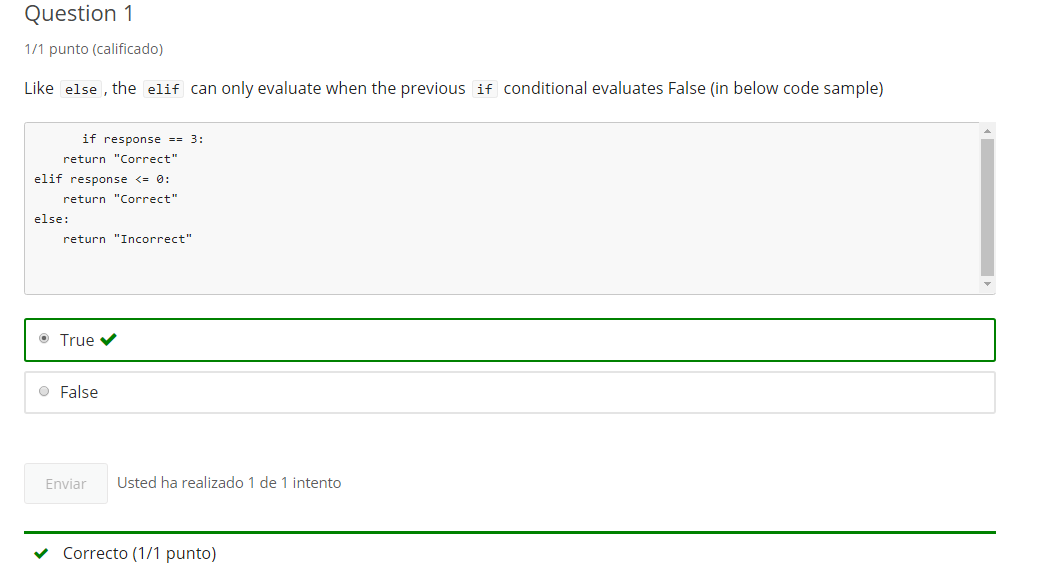
* get input of 2 **integer** numbers
* cast the input and print the input followed by the result
  + Output Example: **9 + 13 = 22**

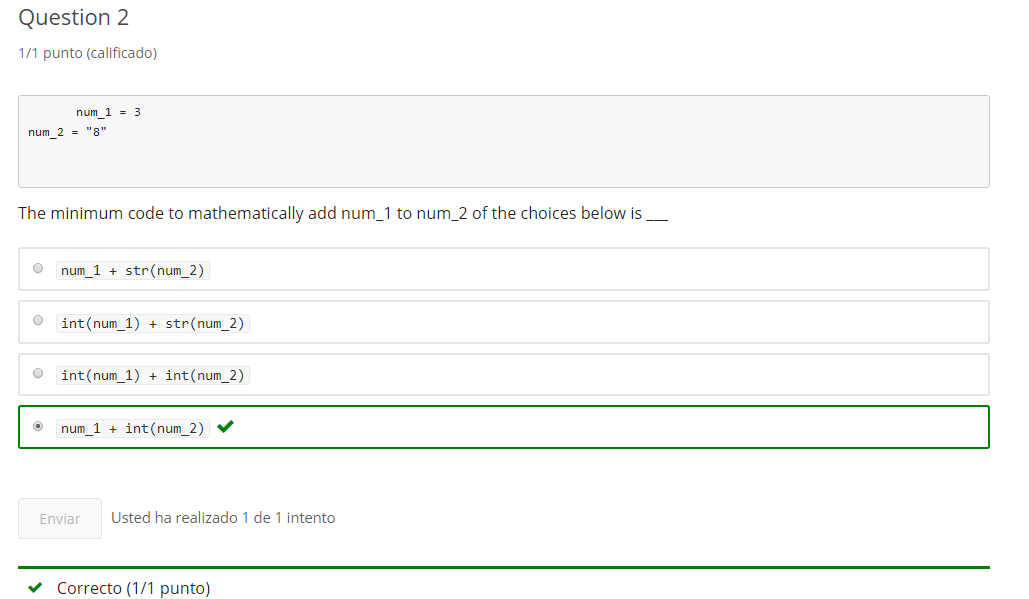
Optional: check if input .isdigit() before trying integer addition to avoid errors in casting invalid inputs

# [ ] code and test the adding calculator

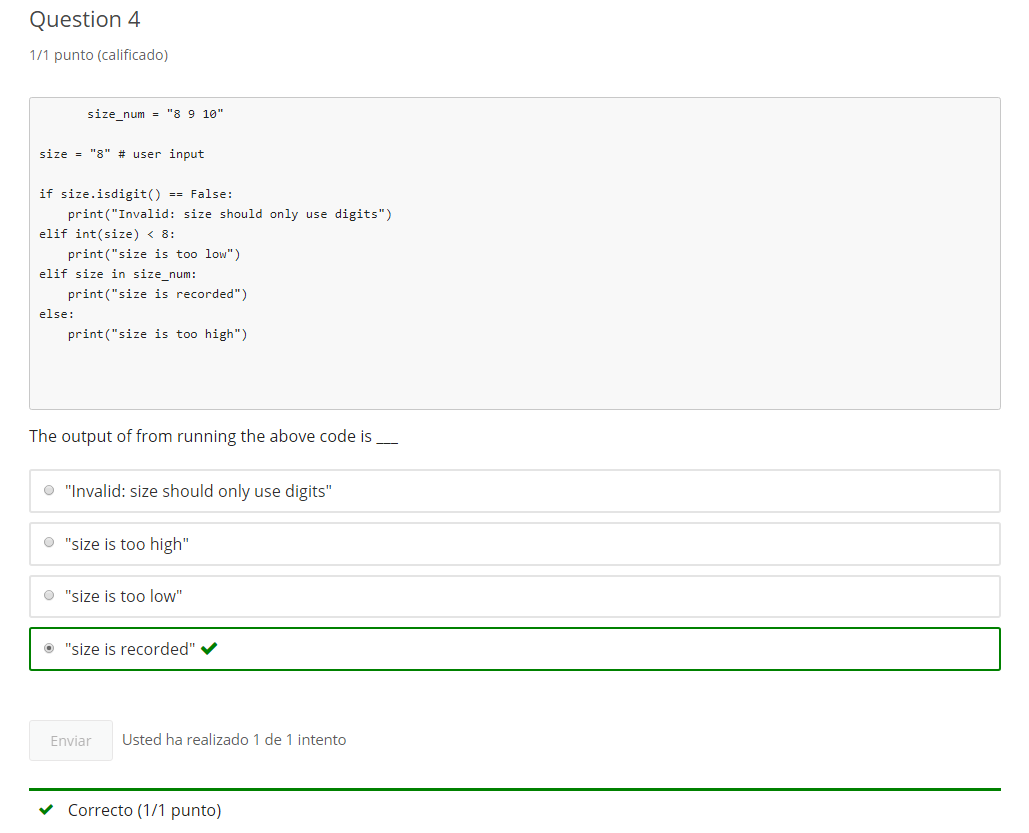


**4.5 Self-Check: Module 3 - Section 4**









**5. Math Operators**

**5.1 Intro Python**

5.2 Python Intro

Jupyter Notebook: MOD03\_1-5.2\_Intro\_Python.ipynb

conditionals, type, and mathematics extended

* conditionals: elif
* casting
* **basic math operators**

**Student will be able to**

* **perform subtraction, multiplication and division operations in code**

**5.2 Basic Math Operators**

**Video:** **BasicMathOperatorsV9.mp4**

# Concept: Math Operators

## Math basic operators

### + addition

### - subtraction

### \* multiplication

### / division

Python contains all of the basic math operators. Not just addition, which we've seen, but also subtraction, multiplication, and division. Let's review the basic math operators built into the Python language. The + sign for addition, the dash or- sign for subtraction, the \* for multiplication and the / also known as forward slash for division. We do the math in a cell or in a console environment. We can just echo that right to the screen. So let's look at 3 + 5. And we can run that and it will give us our answer right there. We also can use order of operations. Things inside of parentheses are evaluated before things that are outside of parentheses. So 3 + 7 times the parentheses. First thing to be evaluated will be the parentheses. So that was 8 we saw. And then the multiplication \* 7. And then + 3. So we see that 7 \* 8 is 56. And plus 3 is 59. We can also evaluate math inside of a print statement. So we see here an example of using multiplication. Division, which will always result in a float. Here we have order of operations with parentheses first and then multiplication. Here we have three statements of addition and subtraction. Again, order of operations, multiplication will happen first. In the other order, multiplication will happen first, 5 \* 10. And then, we have a case where we will do two sets of parenthesis evaluated first. And then they will be multiplied. So let's check that one. 3- 1 should be 2. 22 \* 3 should be 66, so let's look over here. It's 2 \* 66 and we get 132. We can use all of the basic math operators to create equations inside of our programs.

# Example

# [ ] review and run example

print("3 + 5 =",3 + 5)

print("3 + 5 - 9 =", 3 + 5 - 9)

print("48/9 =", 48/9)

print("5\*5 =", 5\*5)

print("(14 - 8)\*(19/4) =", (14 - 8)\*(19/4))

# [ ] review and run example - 'million\_maker'

def million\_maker():

make\_big = input("enter a non-decimal number you wish were bigger: ")

return int(make\_big)\*1000000

print("Now you have", million\_maker())

# Task 1

## use math operators to solve the set of tasks below

# [ ] print the result of subtracting 15 from 43

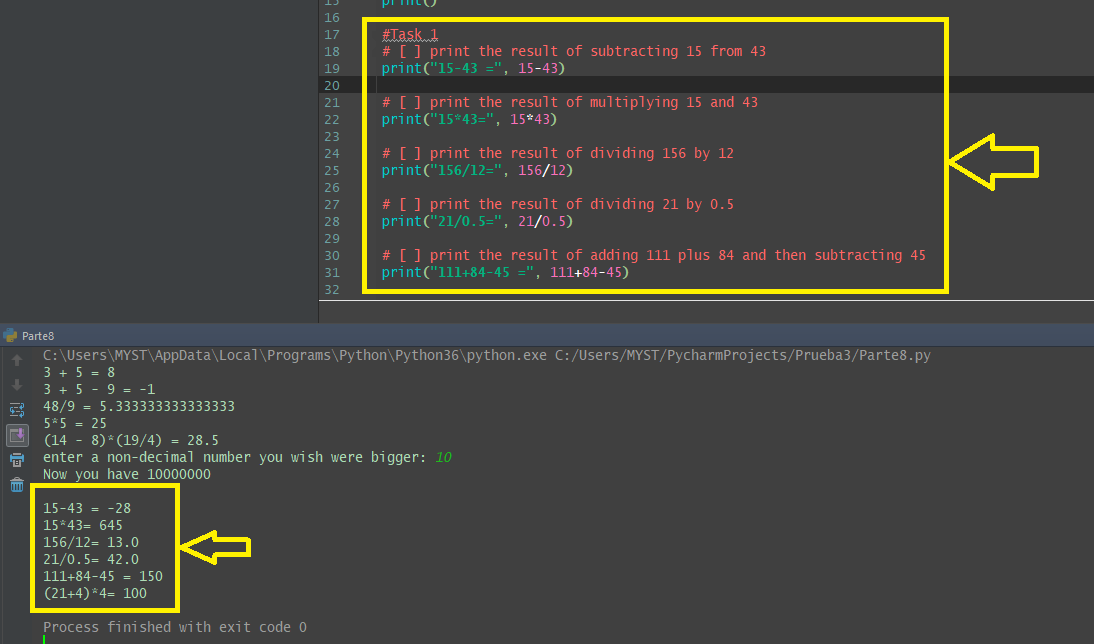
# [ ] print the result of multiplying 15 and 43

# [ ] print the result of dividing 156 by 12

# [ ] print the result of dividing 21 by 0.5

# [ ] print the result of adding 111 plus 84 and then subtracting 45

# [ ] print the result of adding 21 and 4 and then multiplying that sum by 4

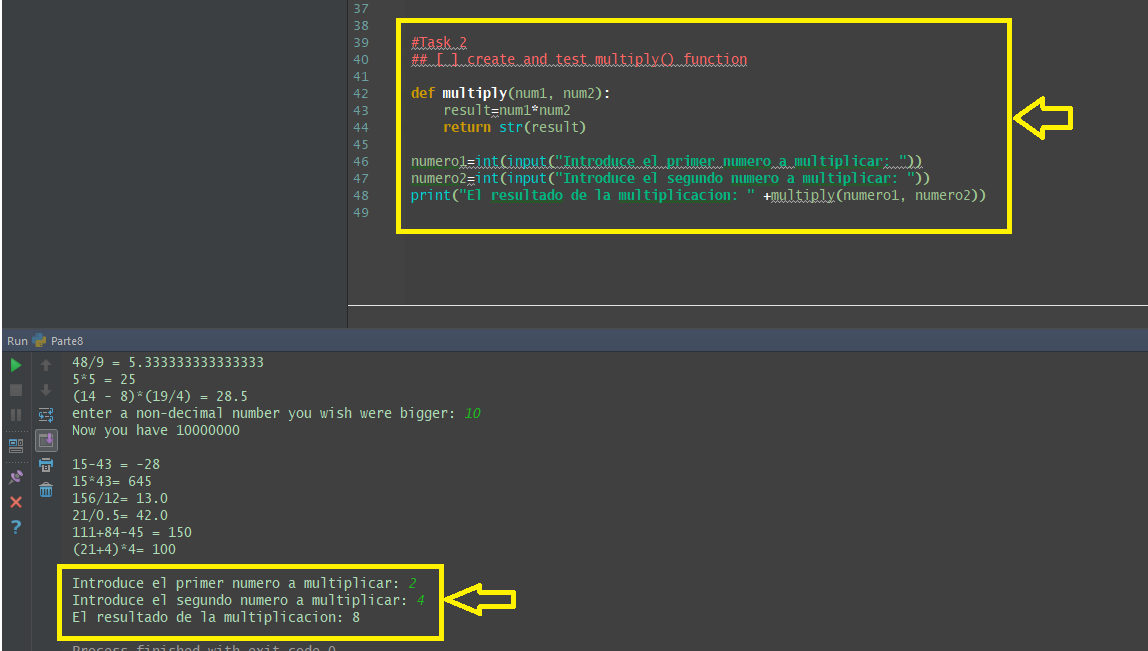


# Task 2

## Program: Multiplying Calculator Function

* define function **multiply()**, and within the function:
  + gets user input() of 2 strings made of whole numbers
  + cast the input to **int()**
  + multiply the integers and **return** the equation with result as a **str()**
    - **return** example
  + 9 \* 13 = 117

# [ ] create and test multiply() function



# Task 3

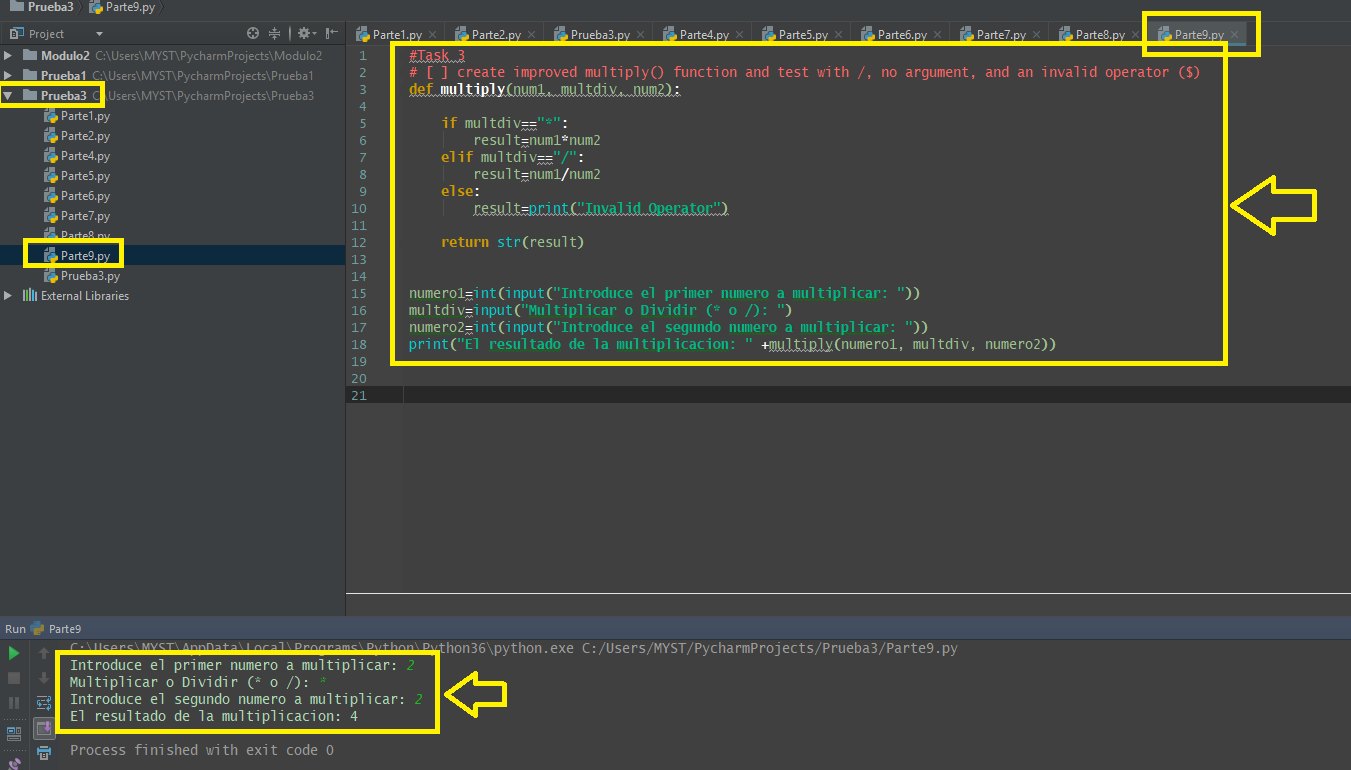
## Project: Improved Multiplying Calculator Function

### putting together conditionals, input casting and math

#### update the multiply() function to multiply or divide

* + single parameter is **operator** with arguments of **\*** or **/** operator
  + default operator is "\*" (multiply)
  + **return** the result of multiplication or division
  + if operator other than **"\*"** or **"/"** then **return "Invalid Operator"**

# [ ] create improved multiply() function and test with /, no argument, and an invalid operator ($)



# Task 4

## Fix the Errors

# Review, run, fix

student\_name = input("enter name: ").capitalize()

if student\_name.startswith("F"):

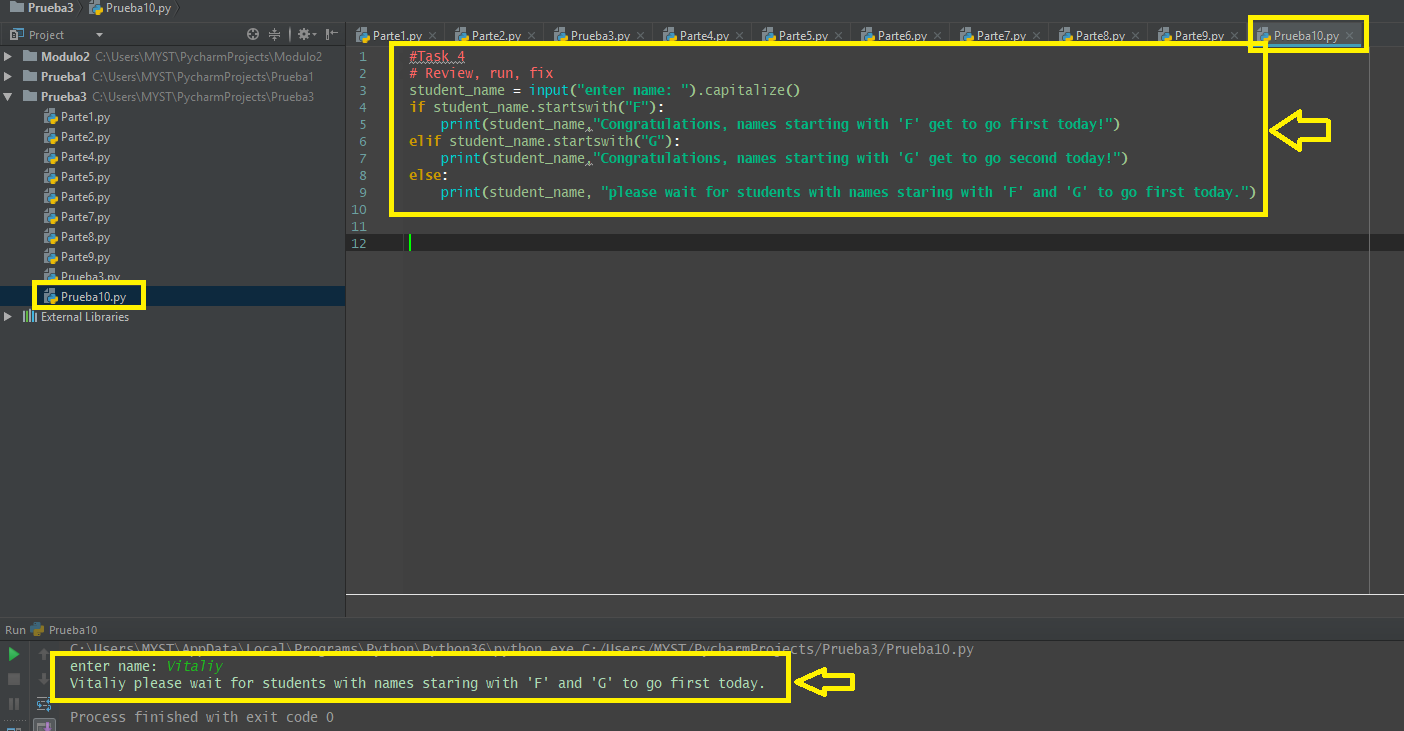
print(student\_name,"Congratulations, names starting with 'F' get to go first today!")

elif student\_name.startswith("G")

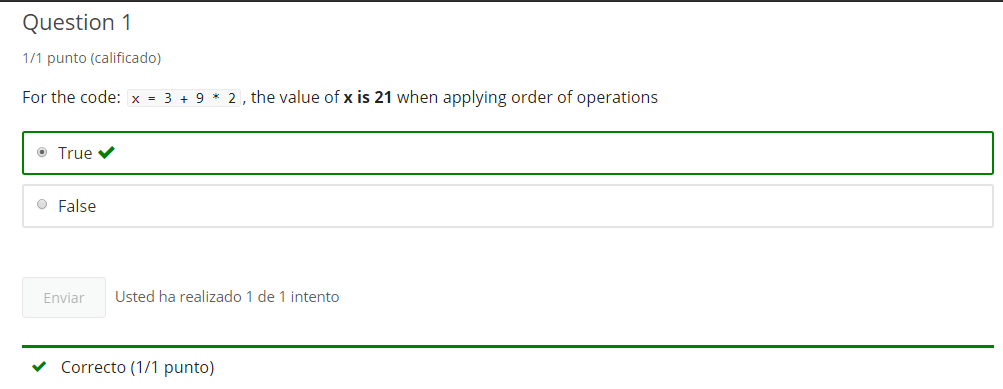
print(student\_name,"Congratulations, names starting with 'G' get to go second today!")

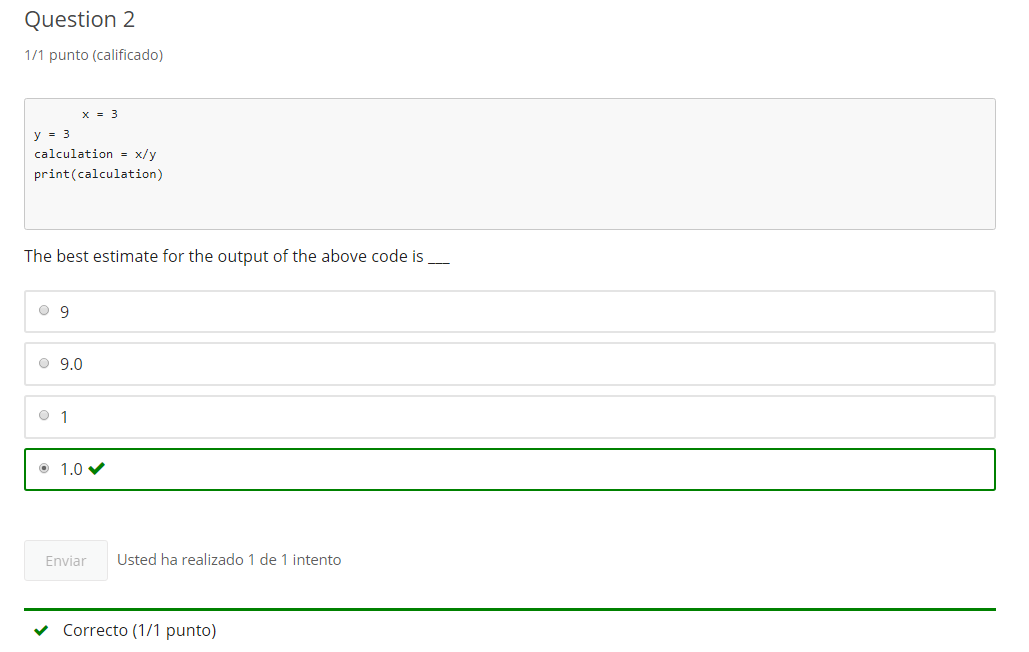
else:

print(student\_name, "please wait for students with names staring with 'F' and 'G' to go first today.")



**5.3 Self-Check: Module 3 - 5**









**6. Conditionals Practice**

**6.1 Intro Python Practice - notebook 4**

Optional practice

Jupyter Notebook: MOD03Practice\_1-4\_IntroPy.ipynb

## Conditionals

# Student will be able to

* **control code flow with if... else conditional logic**
  + using Boolean string methods (.isupper(), .isalpha(), .startswith()...)
  + using comparison (>, <, >=, <=, ==, !=)
  + using Strings in comparisons

## if else

# [ ] input a variable: age as digit and cast to int

# if age greater than or equal to 12 then print message on age in 10 years

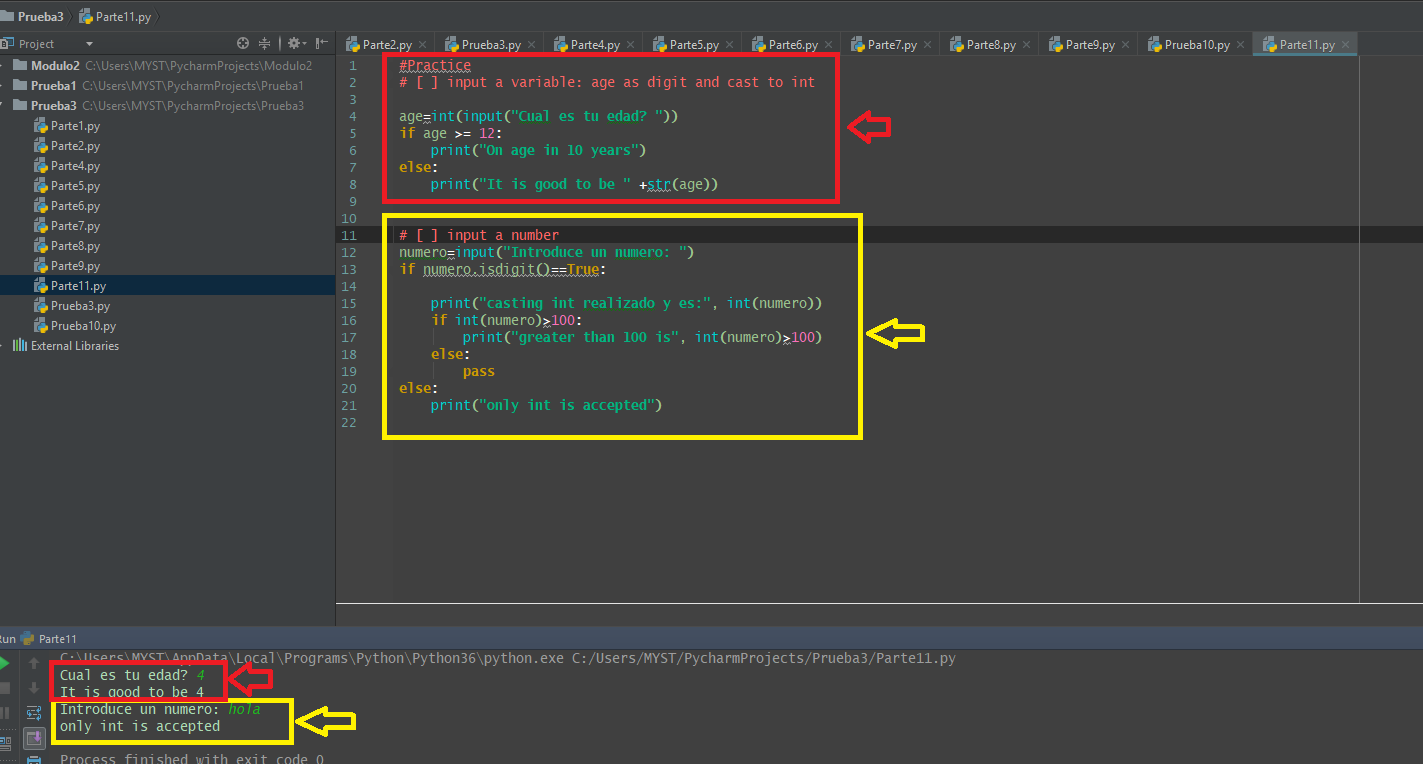
# or else print message "It is good to be" age

# [ ] input a number

# - if number IS a digit string then cast to int

# - print number "greater than 100 is" True/False

# - if number is NOT a digit string then message the user that "only int is accepted"



### Guessing a letter A-Z

**check\_guess()** takes 2 string arguments: **letter and guess** (both expect single alphabetical character)  
- if guess is not an alpha character print invalid and return False - test and print if guess is "high" or "low" and return False - test and print if guess is "correct" and return True

# [ ] create check\_guess()

# call with test

# [ ] call check\_guess with user input

### Letter Guess

**create letter\_guess() function that gives user 3 guesses**

* takes a letter character argument for the answer letter
* gets user input for letter guess
* calls check\_guess() with answer and guess
* End letter\_guess if
  + check\_guess() equals True, return True
  + or after 3 failed attempts, return False

# [ ] create letter\_guess() function, call the function to test

### Pet Conversation

**ask the user for a sentence about a pet and then reply**

* get user input in variable: about\_pet
* using a series of **if** statements respond with appropriate conversation
  + check if "dog" is in the string about\_pet (sample reply "Ah, a dog")
  + check if "cat" is in the string about\_pet
  + check if 1 or more animal is in string about\_pet
* no need for **else**'s
* finish with thanking for the story

# [ ] complete pet conversation

**6.2 Module 3 Practice - notebook 5**

optional practice

Jupyter Notebook: MOD03Practice\_1-5\_IntroPy.ipynb

## conditionals, type, and mathematics extended

# Student will be able to

* code more than two choices using **elif**
* gather numeric input using type casting
* perform subtraction, multiplication and division operations in code

# Tasks

### Rainbow colors

ask for input of a favorite rainbow color first letter: ROYGBIV

Using if, elif, and else:

* print the color matching the letter
  + R = Red
  + O = Orange
  + Y = Yellow
  + G = Green
  + B = Blue
  + I = Indigo
  + V = Violet
  + else print "no match"

# [ ] complete rainbow colors

# [ ] make the code above into a function rainbow\_color() that has a string parameter,

# get input and call the function and return the matching color as a string or "no match" message.

# Call the function and print the return string.

**Create function age\_20() that adds or subtracts 20 from your age for a return value based on current age** (use if)

* call the funtion with user input and then use the return value in a sentence  
  example age\_20(25) returns **5**:

"5 years old, 20 years difference from now"

# [ ] complete age\_20()

**create a function rainbow\_or\_age that takes a string argument**

* if argument is a digit return the value of calling age\_20() with the str value cast as **int**
* if argument is an alphabetical character return the value of calling rainbow\_color() with the str
* if neither return FALSE

# [ ] create rainbow\_or\_age

**Additional Practice**

# [ ] add 2 numbers from input using a cast to integer and display the answer

# [ ] Multiply 2 numbers from input using cast and save the answer as part of a string "the answer is..."

# display the string using print

# [ ] get input of 2 numbers and display the average: (num1 + num2) divided by 2

# [ ] get input of 2 numbers and subtract the largest from the smallest (use an if statement to see which is larger)

# show the answer

# [ ] Divide a larger number by a smaller number and print the integer part of the result

# don't divide by zero! if a zero is input make the result zero

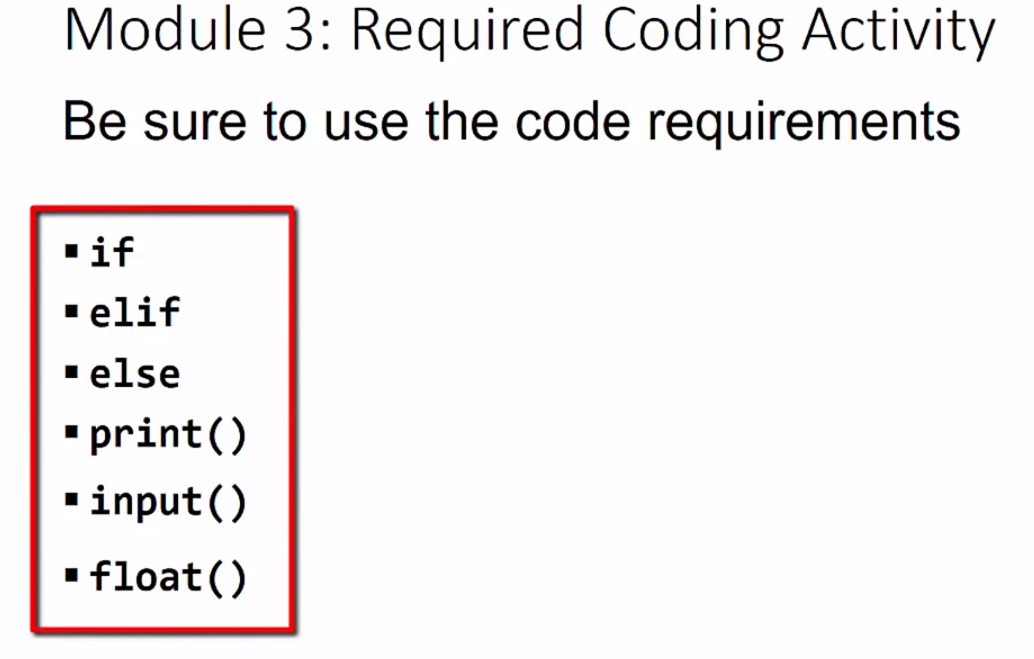
# [ ] cast the answer to an integer to cut off the decimals and print the result

**7. end of Mod coding assignment**

**7.1 Module 3 Required Code Description**

**Video:** **Module3RequiredCodeDescriptionV10.mp4**

We're going to walk through the requirements for the Required Coding Activity for Module 3 and we're going to do a cheese ordering program. And so there are the required keywords if, elif, else, print, input, and float. **If any of these are excluded from your program solution, you will not get credit. So make sure that we are covering each of these items.**



Let's look at edX page. And we see there are requirements, right here, that we just talked about and we also show some of the requirements and code running. We also have that on our Jupiter notebook and in this case I'm going to run some sample code that I've pre-loaded and just show you how the program should work. So, we have the requirements to enter in some values that are above 100, below .25, and somewhere in between and so that we can give different types of messages. So you put in your order value. So here I'll put in something bigger than 100. So we'll just go a little bit bigger 101 and you can also do that with decimal places, because we are going to convert this string input into a float and then it gives us a message that "101.9 is more than is currently available in stock" so they can fulfill that order, we'll run it again and then we'll put in something below .25 so I go .01 and we see that is .01 is below the minimum order amount. So let's order something in between .25 and 100. So I'll just say, 55.3 and then it gives us a cost and so that cost has to convert that amount into a float value and multiply it by a fixed count per unit. So we're going to have to by 55.5 times our requirement. So in our module let's make sure that we're testing our input within if, and elif, and then else statement that we are printing some results, that we're gathering the input that we used, and we're converting that input into float values.

**Module 3 Required Coding Activity**

Introduction to Python Unit 1

**This Activity is intended to be completed in the jupyter notebook, Required\_Code\_MOD3\_IntroPy.ipynb and then pasted into the assessment page that follows.**

All course .ipynb Jupyter Notebooks are available from the project files download topic in Module 1, Section 1.

This is an activity based on code similar to the Jupyter Notebook **Practice\_MOD03\_1-4\_IntroPy.ipynb** and **Practice\_MOD03\_1-5\_IntroPy.ipynb** which you may have completed as practice.

| **Assignment Requirements** |
| --- |
| **NOTE:** This program requires the use of **if, elif, else**, and casting between strings and numbers. The program should use the various code syntax covered in module 3. The program must result in print output using numeric input similar to that shown in the sample output below. |

Program: Cheese Order

* set values for maximum and minimum order variables
* set value for price variable
* get order\_amount input and cast to a number
* check order\_amount and give message checking against
  + over maximum
  + under minimum
* else within maximum and minimum give message with calculated price

Sample input and output:

Enter cheese order weight (numeric value): 113

113.0 is more than currently available stock

Enter cheese order weight (numeric value): .15

0.15 is below minimum order amount

Enter cheese order weight (numeric value): 2

2.0 costs $15.98

# [ ] create, call and test

# then PASTE THIS CODE into edX

Important: [How to submit code by pasting](https://courses.edx.org/courses/course-v1:Microsoft+DEV236x+1T2017/wiki/Microsoft.DEV236x.1T2017/paste-code-end-module-coding-assignments/)

**7.2 Module 3 Required Code Submission**

