Lecture on conditionals in Python:

Conditionals are how the computer checks conditions and changes the behavior accordingly. We do this all the time in life. If it’s raining outside we bring an umbrella….

There are also conditions which must be checked automatically. For example, if you’re at the ATM trying to take money out, the machine has to make sure you have enough money.

Pseudocode on the board:

If withdrawal is more than balance, then stop the transaction.

If withdrawal is less than balance, then give the money and recalculate balance as balance – withdrawal

Note: this second case occurs exactly when the first case fails. Only one of these cases can occur.

if( … some test …):

block of

instructions

all indented

else:

some other

block of instructions

Notice the format. ‘if’ is a keyword. It is followed by an expression, then followed by a colon. The lines of code which will be evaluated if the test is true are all indented. Note: you don’t need the else clause. If you don’t have the else clause then it’s possible the computer will do nothing (if the test fails).

If that test is not true then we SKIP those indented lines of code. They are never evaluated. Conditionals allow the program to **branch** (i.e. to become a **branching program**). Such a program can change the order of instructions based on some test, e.g. the value of some variable.

Next, type the code into Python for the balance and withdrawal code and demonstrate what happens.

This is what makes Python fun: it’s interpreter-based and fun. Origin of the name comes from Monty Python. It was designed in 1989-1991 and was meant to make programming possible for everyone. Designed by Guido van Rossum. Designed to be human readable and many ways to solve any problem.

Next, what if there are more than two possible outcomes? We need what is called a **chained conditional.** For example:

if(x > 0):

print(‘x is positive’)

elif(x < 0):

print(‘x is negative’)

else:

print(‘x is zero’)

Harp on the fact that only one of the if/else-if/else statements is going to fire. They will be tested in order.

What if we wanted to test if x is equal to zero directly (rather than in the else catch-all)? We’d type

if(x == 0):

<block of code>

On == vs. =

The symbol = is an **assignment**. It says take the thing on the left and use it as a name to bind to the thing on the right. Make them the same in value. Note that if x is 5 and y is 3 then

x = y

print(x) -> returns 3

y = 2

print(x) -> returns 3, because x was only set equal to the value of y, not equal forever.

The symbol == is a **comparison**. Is this thing equal to this other thing in value?

It returns True or False. Demonstrate in the Python Shell, by typing 2 == 3 and 2 == 2

Also, bring up != for not equal

Discuss > vs. >=

Now an example for them to code. We want to get from a numerical grade to a letter grade.

If the score is 90 or above then it’s an A

If the score is between 80 and 90 then it’s a B

If 70-80 it’s a C

If 60-70 it’s a D

Otherwise it’s an F … After they code for a few minutes, show them what I wrote for this one.

Next up is **nested conditionals**. Here there is an if-else branch within an if block of code. For example

# Nested conditionals example

balance = 500

withdrawal = 200

if(withdrawal > balance):

print('Sorry, but you do not have that much money.')

else:

if(withdrawal == balance):

balance = 0

print('Okay, here is all your money')

else:

balance = balance - withdrawal

print('Here is your money. New balance is',balance)

These can be confusing to follow, so it’s important to **use comments**. Also, better to avoid if possible. Comments – put them in if they help the reader understand what’s going on. For example, explain the intuition behind a piece of code, or explain what something is there to do. I’m going to over-comment so we can walk through this. Python puts comments in red.

Now we’ll do a more complicated example, but first we need a new operator on integers.

**Mod operator is %, quotient is /**

Consider 7 divided by 3. It’s 2 with a remainder of 1 (because 7=2\*3+1). Python says 7/3 = 2 and 7%3 = 1

If x % y is zero then x is divisible by y. We would test this as if(x % y ==0):

A number z is even exactly when z % 2 == 0

Now let’s talk about leap year. In the Gregorian calendar, for any year after 1582. If the year is divisible by 4 then it's a leap year unless it's divisible by 100. The exception is years divisible by 400, which are leap years. So 2004 is a leap year, 2100 is not, but 2000 and 2400 are.

How would you code this? Get them talking and write what they say on the board. Many diff ways.

Show them my code & diff approaches. Use this as a segue (“seg-way”) to discuss Booleans (True/False) and Boolean operators (and/or/not). They’ll likely discuss this more as the course goes on.

Regarding a bunch of (if, else if, else) statements vs. using Boolean operators:

I like to have multiple ways to think about things. When you’re learning this can help you get a toe-hold on the material. Then, when you’re an expert you use different ways to play off one another.