ITEC 1150 Week 4 Chapter 2 Part1

CONDITIONAL STATEMENTS, MORE FORMATTING & START ON VALIDATION

Check in

- Remember, turning in partial work on the due date and finishing later is better than turning in everything late!
- Email me with simple questions…
- For complicated questions,
  - Email me to set up time to meet
  - Come to office hours
  - Meet with a tutor
- I know the videos can be long, but I explain the concepts more fully that the slides alone do. At the least, if you are confused by something in the slides, find the place in the video where I'm talking about that slide and watch that part.
- Long story short, I'm here to help!

The Boolean Datatype

ALWAYS EITHER TRUE OR FALSE

The Boolean Datatype

- The `Boolean` data type has two values: `True` and `False`
- Comparison operators are used to compare values:
  - `a == b` Is `a` equal to `b`?
  - `a != b` Is `a` *not* equal to `b`?
  - `a < b` Is `a` less than `b`?
  - `a > b` Is `a` greater than `b`?
  - `a <= b` Is `a` less than or equal to `b`?
  - `a >= b` Is `a` greater than or equal to `b`?
- The result of any comparison will be `True` or `False`

```
'Hello' == 'Hello'     # True
'Dog' != 'Cat'         # True
42 == 24               # False
'Dog' == 'Cat'         # False
10 <= 10               # True
10 < 10                # False
```

Boolean Operators

- We can evaluate multiple comparisons with the Boolean operators **and, or,** and **not**
- The result will be a single **True** or **False** answer

AND statements:

```
'Dog' == 'Dog' and 'Cat' == 'Cat'      # True if all the comparisons are True
'Dog' == 'Dog' and 'Cat' == 'Tiger'    # False if any comparisons are False
```

OR statements:

```
'Dog' == 'Dog' or 'Cat' == 'Tiger'          # True if at least 1 of the comparisons are True
'Dog' == 'Cat' or 'Cat' == 'Tiger'          # False if none of the comparisons are True
```

Boolean Operators (cont.)

- We can combine these into some very complicated expressions:

```
1. temp = 45 # in Celsius
2. pressure = 80 # in kPa
3. # Sound alarm if:
4. # - Temp over 40 C or under 0 C
5. # - Pressure over 100 kPa or under 50 kPa
6. if (temp > 40 or temp < 0) or (pressure > 100 or pressure < 50):
7.     sound_alarm()
```

You can group `Boolean` logic with parenthesis `()`.
The pressure and temp checks are evaluated to `True`/`False` and then the `or` in the middle is evaluated.

- We use these types of comparisons to control the flow of a program by using **`if`** statements…

Code Blocks

BEFORE WE CAN LOOK AT IF STATEMENTS, WE NEED TO KNOW HOW PYTHON CODE IS STRUCTURED INTO DIFFERENT EXECUTION PATHS

Block Basics

```
1. score = int(input('Enter quiz score: '))
2. if score >= 90:
3.     print('Grade: A')
4. elif score >= 80: #elif means 'else if'
5.     print('Grade: B')
6. elif score >= 70:
7.     print('Grade: C')
```

- Each of the indented sections is a **block**.
- A block is related to the line above it, which ends with a colon **:**
- Code inside of a block only runs if the statement above it is **True**
- The convention for indentation is 4 spaces.
- Python is very strict about indentation; accidentally indenting a block one space too many or too few will cause an error.

Block Basics (cont.)

```
1.  score = int(input('Enter quiz score: '))
2.  if score >= 90:
3.      name = input('Enter student name: ')
4.      print(f'Great job, {name}!')
5.      print('Grade: A')
6.  elif score >= 80:
7.      print('Grade: B')
8.  elif score >= 70:
9.      print('Grade: C')
10. # name no longer exists down here!
```

- A block can have many lines of code inside of it, and you can even declare variables in a block
- Variables declared in a block only exist until the block is done!

Because `name` is first declared inside this block, it only exists until line 5 and then it's gone!

Controlling Program Decisions

IF, ELIF (ELSE IF), ELSE

Making Decisions In Code

- It's useful for programs to make decisions on what code to run.
- Decisions are made based on whether something is True or False.
- In your exercise file for lesson2-1, try this code:

```
temp = 67
if temp > 0:
    print(f'The temperature {temp}f is greater than 0.')
```

Note the syntax. "`if`" statements end with a colon ":"

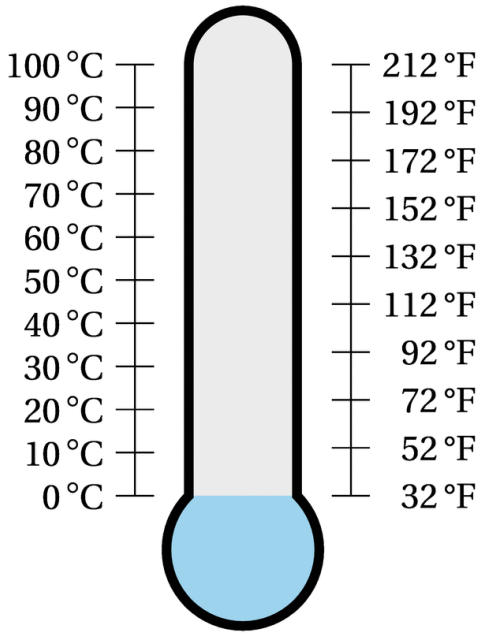
Lines that are 'controlled' by the **`if`** statement are indented under the **`if`**

The IF Statement

- Adapt this program to collect the temp from the user :

```
1. temp = float(input('Enter the temperature: '))
2. if temp > 0:
3.     print(f'The temperature {temp}f is greater than 0.')
4. print('That\'s all folks!')
```

- Run and test your program with positive and negative temps.



The ELSE Statement

- Make further modifications to your program.
- Note the colons and indentations! It creates a "shape" that helps us troubleshoot.

```
1. temp = float(input('Enter the temperature: '))
2. if temp > 0:
3.     print(f'The temperature {temp}f is greater than 0 – thankfully!')
4. else:
5.     print(f'The temperature {temp}f is less than 0 - brrr!')
6. print('That\'s all folks!')
```

- Run and test your program with different temperatures

The ELIF (else if) Statement

- Make another modification. You use the double equal sign "==" in the **elif** line

```
1. temp = float(input('Enter the temperature: '))
2. if temp > 0:
3.     print(f'The temperature {temp}f is greater than 0 – thankfully!')
4. elif temp == 0:
5.     print('The temperature is exactly 0!')
6. else:
7.     print(f'The temperature {temp}f is less than 0 - brrr!')
8. print('That\'s all folks!')
```

Combining Conditional Statements

- Think about why you might want to use `elif` and `else` instead of three `if` statements. Sometimes you only want a condition to be checked if a previous condition was `False`:

```
1. temp = float(input('Enter the temperature: '))
2. sky_condition = input('Enter the sky condition: ')
3. if temp > 0:
4.     print(f'The temperature {temp}f is greater than 0 – thankfully!')
5. elif sky_condition == 'Sunny':
6.     print('It might be cold, but at least it\'s sunny!')
7. else:
8.     print(f'The temperature {temp}f is less than 0, and it\'s dark or cloudy.')
9. print('That\'s all folks!')
```

Equivalency vs Assignment Operators

- Note that this code uses both the assignment operator = and the equivalency operator ==.
- Add this code to your exercise file:

```
1. college = input('Please enter your college: ')
2. if college == 'Minneapolis College’:
3.     print('Good choice!')
```

It's easy to make a mistake, so pay careful attention to whether you're assigning a value or checking equality.

Using Not Equals

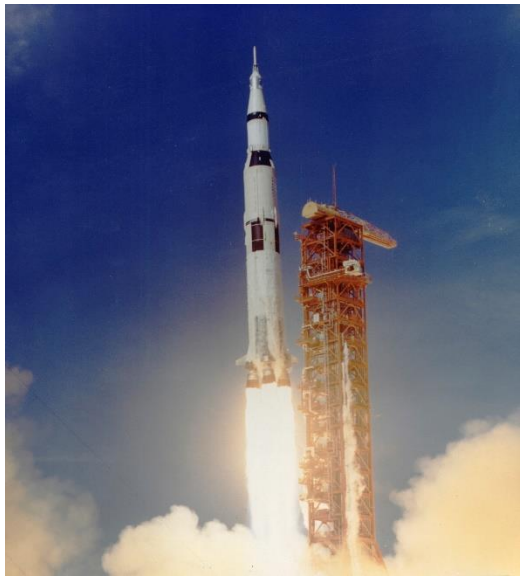
- This code uses != (not equal) to test if something is equal to something else
- It is the logical equivalent to the slide before
- Try this code:

```
1. college = input('Please enter your college: ')
2. if college != 'Minneapolis College' :
3.     print('You should go to Minneapolis College!')
4. # any difference will return True
```

Making Decisions In Code (cont.)

- Try the following code:

```
year = int(input('What year did Apollo 11 land on the moon? '))
if year != 1969:
    print(f'Sorry, {year} is the wrong answer.')
else:
    print(f'Correct! {year} is right!')
```



Logical Equivalence

- You will notice, there are a lot of correct ways to write a decision statement
- The code below is logically equivalent to the code on the previous slide
- Put this version into a new apollo.py file, since it will work better for our lab.

```
1. year = int(input('What year did Apollo 11 land on the moon? '))
2. if year == 1969:
3.     print(f'Correct! {year} is right!')
4. else:
5.     print(f'Sorry, {year} is the wrong answer.')
```

Covering All Conditions

- Think over the code below. What happens if you enter a score of 69?

```
1. score = int(input('Enter quiz score: '))
2. if score >= 90:
3.     print('Grade: A')
4. elif score >= 80:
5.     print('Grade: B')
6. elif score >= 70:
7.     print('Grade: C')
```

- Sometimes, doing nothing for some conditions is fine:

```
1. if user_birthday == datetime.today():
2.     print('Happy Birthday!')
3. # no else needed here!
```

- But make sure to think about all possibilities and what you need to do for each one.

Nested Conditionals

- Blocks can be inside blocks. Note the code on the right.
- On line 3, `isnumeric()` returns `True` if all the characters are digits 0-9. (This can't be used to check for floats, which contain a decimal point.)
  - If `True`, we know we can convert it to an `int` and continue our grading
- Can you see how our grading code is contained in the `if` block starting on line 3?

```
1.  score = input('Enter quiz score – whole number 0-100: ')
2.  # validation block
3.  if score.isnumeric() is True:
4.      score = int(score)
5.      if score >= 90:
6.          print('Grade: A')
7.      elif score >= 80:
8.          print('Grade: B')
9.      elif score >= 70:
10.         print('Grade: C')
11.     else:
12.         print('Your grade is currently undefined.')
```

Nested Conditionals and Validation

- Now we can add simple input validation!
- We already know we can do our grading if `isnumeric()` is `True`
- If it's `False`, we can print an error message.

```
1.  score = input('Enter quiz score – whole number 0-100: ')
2.  # validation block
3.  if score.isnumeric() is True:
4.      score = int(score)
5.      if score >= 90:
6.          print('Grade: A')
7.      elif score >= 80:
8.          print('Grade: B')
9.      elif score >= 70:
10.         print('Grade: C')
11.     else:
12.         print('Your grade is currently undefined.')
13. else:
14.     print('Try the program again; it only takes whole numbers.')
```