Conditional Execution

- We've already seen how programs execute
  - Sequentially: one line at a time, top to bottom
  - Repeatedly (iteratively): using a *for* statement
- Now we add *selective* or *conditional* execution
  - Do this or not, depending on the situation
  - Choices come up all the time in real-world processes (algorithms)
- Conditional execution uses the *if* statement, which works just the way it sounds:

```
x = input("How old are you?")
if x > 39:
  print("That's really old")
```

  - "x > 39" is the *condition* (the situation we're checking)
  - The print statement is executed or not depending on the value in x

Boolean Data Type and Logical Expressions

- *Boolean* is a data type, just like integer
  - Values are *True* and *False*, instead of 1, 2, 567, etc.
  - Named for George Boole
- Arithmetic expressions evaluate to numbers
  - Using arithmetic operators (+, -, %, etc)
  - E.g. 1 + 5 evaluates to 6
- Logical expressions evaluate to Boolean (True or False)
  - Using *relational* (comparison) operators
  - 3 < 7 evaluates to False (3 is not less than 7)
  - 17.4 >= 15 evaluates to True (17.4 is greater than or equal to 15)
  - 4 == 4 evaluates to True (4 is equal to to 4)
  - Note that = is assignment, == is comparison

Relational Operators

- Equal: ==
  - Remember, = is already taken for assignment
- Not equal: !=
- Greater than: >
- Less than: <
- Greater than or equal to: >=
- Less than or equal to: <=

The Truth about Booleans

- Just like characters are actually numbers (ASCII codes)…
- …Booleans are just 0 (False) or 1 (True) to the computer

- For historical reasons, any number that is not 0 is considered a True value

```
if 7:
   print(“Yes, this will print, because 7 is not 0”)

if 17*4:
   print(“This too”)

if math.cos(2.3):
   print(“Even this”)
```

Back to Conditions

- An *if* condition can be anything that evaluates to True or False
  - A literal value

```
if True:
   print(“This is silly, it always prints”)
```

  - A variable with a Boolean value

```
raining_today = False
if raining_today:
   print(“Only prints if the raining_today variable is set to True”)
```

  - A logical expression

```
number = input(“What is your favorite number?”)
if number == 17:
   print(“Only people who like 17 are worthy to see this”)
```

  - Also, a function that returns a Boolean value

Comparing Numbers

- Integer and floating-point types can be compared
  - `8 < 15 evaluates to True`
  - `6 != 6 evaluates to False`
  - `2.5 > 5.8 evaluates to False`
  - `5.9 <= 7 evaluates to True`

Comparing Other Data Types

- Characters are compared by their ASCII value
  - Alphabetical order
  - Except all the uppercase letters are before all the lowercase letters
- Strings are compared character-by-character
  - Again, basically the same way you would alphabetize
  - Because that's both straightforward and useful
- Lists are compared item-by-item
  - Lists and strings are both sequences
  - Makes sense to be consistent

Two-Way Conditional Execution

- What about choosing between multiple options?
  - Also comes up all the time
  - Can use an *else* statement together with an *if* statement

- Again, works in a pretty intuitive way:

```
x = input(“How old are you?”)
if x > 39:
   print(“That’s really old”)
else:
   print(“How you doin’?”)
```

Blocks

- An *if* statement controls whether to execute a *block* of code
  - Can be a single statement, or multiple statements
  - Just like with a *for* loop
- All statements in the block are indented:

```
x = input(“How old are you?”)
if x > 39:
   print(“That’s really old”)
   x = x - 5
   print(“there, isn’t that better?”)
else:
   print(“How you doin’?”)
```