Week 2 - Wednesday

COMP 1800

Last time

- What did we talk about last time?
- π
- **`math`** library
- Archimedes' approximation
- Leibniz's approximation

Questions?

Back to π

Problem solving

- The famous mathematics educator George Pólya outlined a series of steps for solving problems:
  1. Understand the problem
  2. Make a plan
  3. Execute the plan
  4. Look back and reflect

John Wallis

- John Wallis was a 17th century British mathematician who is believed to have come up with the ∞ symbol for infinity
- Of course, he also found an approximation for π
  - $\frac{\pi}{2} = \frac{2}{1} \cdot \frac{2}{3} \cdot \frac{4}{3} \cdot \frac{4}{5} \cdot \frac{6}{5} \cdot \frac{6}{7} \cdot \frac{8}{7} \ldots$

Wallis approximation

- $\frac{\pi}{2} = \frac{2}{1} \cdot \frac{2}{3} \cdot \frac{4}{3} \cdot \frac{4}{5} \cdot \frac{6}{5} \cdot \frac{6}{7} \cdot \frac{8}{7} \ldots$

- As with Leibniz, we will use the Accumulator Pattern
- Although the Accumulator Pattern often adds things up, we'll be multiplying stuff as we go
  - When summing, we start with 0
  - When multiplying, we start with 1
- Note that both the numerator and the denominator are used twice in a row before increasing by two
  - But they change on opposite turns!
- How can we apply Pólya's problem solving approach?

Selection Statements

Behold!

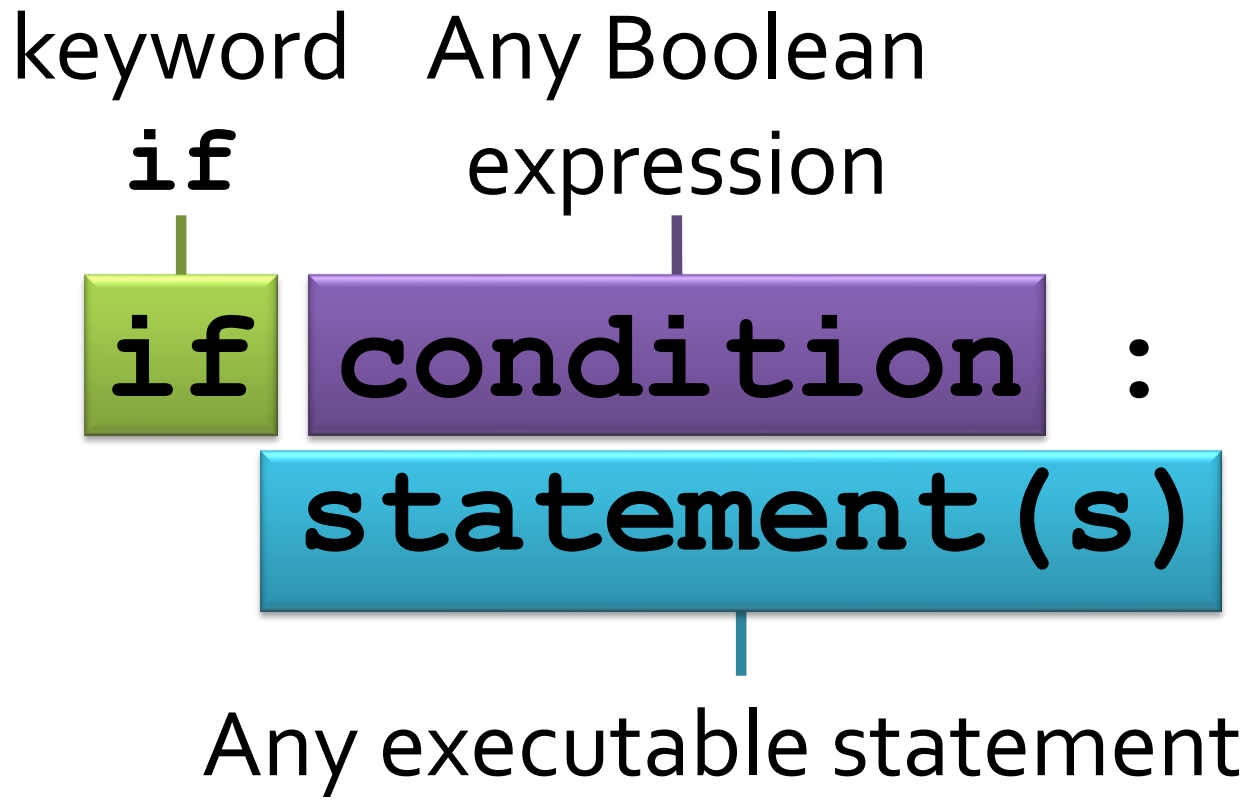
- To make choices in our program, we can use an **`if`**-statement:

```
x = 4

if x < 5:
  print('x is small!')
```

- **`x is small`** will only print out if **`x`** is less than 5
- In this case, we know that it is, but **`x`** could come from user input or a file or elsewhere

Anatomy of an `if`



Note: The colon after the condition and the indentation before the statement are **required**

Conditions in the `if`

- Any statement that evaluates to a Boolean is legal
- Examples:
  - `x <= y`
  - `True`
  - `s == 'Help me!' and z < 4`

- In actual fact, almost anything can be used for the condition
  - Integers and floating-point values are considered `True` as long as they aren't zero
  - Most strings (except the empty string `''`) are considered `True`
  - Avoid using things that aren't Booleans, since it's confusing

Comparison

- The most common condition you will find is a comparison between two things
- In Python, that comparison can be:
  - **==** equals
  - **!=** does not equal
  - **<** less than
  - **<=** less than or equal to
  - **>** greater than
  - **>=** greater than or equal to
- These are called **relational** operators

Equals

- You can use the `==` operator to compare any two things of the same type
- Different numerical types can be compared as well (`3 == 3.0`)
- Be careful with floating-point types, `0.33333333` is not equal to `0.33333332`

```
x = 3
if x == 4:
  print('This doesn't print')
```

Not Equals

- Any place you could have used the **==** operator, you can use the **!=** operator
- If **==** gives **True**, the **!=** operator will always give **False**, and vice versa
- If you want to negate a condition, you can always use the **not** operator

```
if x != 4:
```

is the same as

```
if not(x == 4):
```

= != ==

- Remember, a single equal sign (**=**) is the **assignment** operator (think of a left-pointing arrow)
- A double equals (**==**) is a **comparison** operator
- Assigning variables in **`if`** statements is not allowed

```
y = 10
if y = 6:           # syntax error
  print('Oh, no!')
```

```
b = False
if b = False:       # syntax error
 print('Not this?')
```

Less Than (or Equal To)

- Inequality is very important in programming
- You may want to take an action as long as a value is below a certain threshold
- For example, you might want to keep bidding at an auction until the price is greater than what you can afford

```
if x <= 4:
  print('x is less than 5')
```

- Watch for strict inequality (**<**) vs. non-strict inequality (**<=**)

Greater Than (or Equal To)

- Just like less than or equal to, except the opposite
- Note that (because of the All-Powerful Math Gods) the opposite of **<=** is **>** and the opposite of **>=** is **<**
- Thus,
  - **`not( x <= y )`** is equivalent to **`( x > y )`**
  - **`not( x >= y )`** is equivalent to **`( x < y )`**

and and or

- You can also have multiple Boolean conditions in an **`if`** statement
- You can join them together with:
  - **`and`** (which results in a **`True`** value only if both the conditions it joins are **`True`**)
  - **`or`** (which result in a **`True`** value if either of the conditions it joins are **`True`**)

```
if attempts < 5 and password == 'open sesame':
  print('You know the secret!')
```

Another useful library

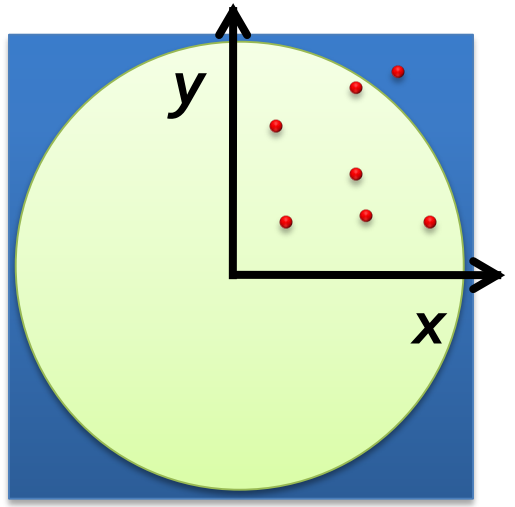
- The **`random`** library lets us produce random numbers
- It has two functions that will be useful to us:
  - **`randint(a, b)`**: Returns a random integer **`n`** where **`a`** ≤ **`n`** ≤ **`b`**
  - **`random()`**: Returns a random floating-point value from [0, 1)
- To use them, import **`random`** and then call the functions qualified by **`random`** followed by a period:

```
import random

dice = random.randint(1, 6)
percentage = random.random()
```

Monte Carlo approximation of π

- We can do something called a Monte Carlo approximation of π
- We "throw" darts at a 1 x 1 square in the upper right corner of a circle with radius 1
- We count the ones that fall inside the circle and divide by the total darts thrown
- That fraction is an estimation of the area of one fourth of the circle
- By multiplying by 4, we approximate π



Monte Carlo approximation function

- Here is a function that performs the Monte Carlo approximation:

```
import random

def monteCarlo(darts):
    hits = 0
    for i in range(darts):
        x = random.random()
        y = random.random()
        if x*x + y*y <= 1.0: # see if dart is in circle
            hits += 1
    return 4.0 * hits / darts
```

Quiz

Upcoming

Next time…

- More on selection statements
- Visualization of Monte Carlo simulation
- We'll mostly have work time for assignments
  - Assignment 1 is due Friday by midnight
  - Assignment 2 will be available to work on as well

- 20 employers in the fields of Engineering and Computer Science
- 20 alumni members attending
- Free professional LinkedIn headshots
- Plenty of food and great conversations
- Build new connections on LinkedIn
- Door prizes
- Network with people in your field
- Learn about possible internships
- Gain new insights about your major
- Required event for all sophomores

ENGINEERING PROFESSIONAL DEVELOPMENT

SEPTEMBER 7, 5PM-7.30PM, @ THE POINT

CAREER JUMPSTART:

ENGINEERING & COMPUTER SCIENCE

Come and network with engineering and computer science alumni and business partners and learn how to be successful in your strategic job and internship search

REGISTER HERE WITH HANDSHAKE

OTTERBEIN UNIVERSITY

Reminders

- Review Chapter 2 of the textbook
- Emergency elections for CS Club
  - Do you want to have a voice in CS Club?
  - Come vote at 4 p.m. this Wednesday, August 30 in Point 113!