Finding Pi

CS 8: Introduction to Computer Science
Lecture #5

Ziad Matni
Dept. of Computer Science, UCSB

Administrative

- This class is currently **FULL**
  - Sorry, no more adds ☹
- Project #1
  - The syllabus shows this due today (it's not)
  - I'll assign the 1st project soon (prob. Thursday)
- Midterm #1 is NEXT WEEK!
  - omgomgomgomgomgomgomg

MIDTERM IS COMING!

- Material: ***Everything*** we've done**, incl. up to Th. 4/20**
  - Homework, Labs, Lectures, Textbook
- **Tuesday, 4/25** in this classroom
- **Starts at 3:30pm **SHARP****
- **Pre-assigned seating**
- Duration: **1 hour long**

- Closed book: no calculators, no phones, no computers
- Only 1 sheet (single-sided) of written notes
  - Must be no bigger than 8.5" x 11"
  - **You have to turn it in with the exam**
- **You will write your answers on the exam sheet itself.**

A Function To Draw A Square

- Part of listing 1.2 from the text (p. 30)

```
def drawSquare(myTurtle, sideLength):
    myTurtle.forward(sideLength)
    myTurtle.right(90)   # side 1
    …
```

- Then to invoke it for drawing a square that has 20 pixels on each side using a turtle named `t`:

```
>>> drawSquare(t, 20)
```

- What might happen if we invoked `drawSquare(20, t)`?

Let's try it out!

More drawing abstraction

- Contrast – a triangle vs. a square (Listing 1.5)

```
def drawTriangle(myTurtle, sideLength):
    for i in range(3): # draw 3 sides, not 4
        myTurtle.forward(sideLength)
        myTurtle.right(120) # 120°× 3
```

- Hmm…any regular polygon? (Listing 1.6, p. 38)

```
def drawPolygon(myTurtle,sideLength,numSides):
    turnAngle = 360 / numSides
    for i in range(numSides):
        myTurtle.forward(sideLength)
        myTurtle.right(turnAngle)
```

Let's try these out!

An Ancient Problem: Finding 

- Ratio of a circle's circumference to its diameter

```
π = circumference / diameter    # for any circle
```

- Irrational number: an infinite series of non-repeating digits
  - So it can never be represented exactly, only *approximated*

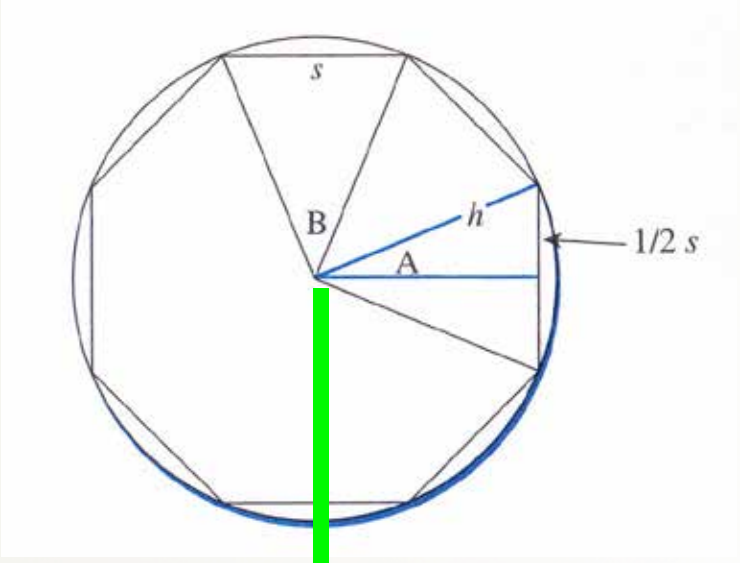
- Chapter 2 explores various ways to approximate pi
  - But just to teach problem-solving. For calculating, use **`math.pi`** module

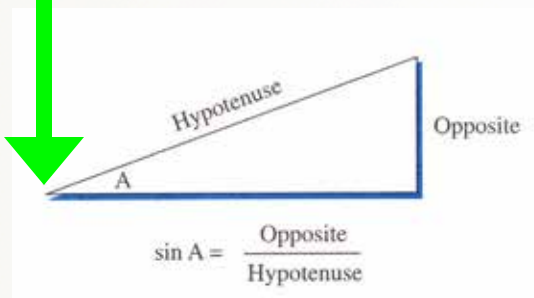
```
import math # necessary to use math module
area = math.pi * radius * radius
```

- By the way, the math module has lots of other cool stuff
  - Square root, trig functions, e, … try `>>> help(math)`

Archimedes Approach

- Recall: `π = C / d` and
  `d = 2 * r`
- Simplify: set `r = 1`,
  then `π = C / 2`
- Solve for `C` to find `π`
  - Need trig: `½ s = sin A`
    where `A = 360/sides/2`
- Finally `C = sides * s`
  - See Session 2.3, Listing 2.2 (page 52)





Accumulator Pattern

- Introduced by other ways to find pi
  - Uses infinite series and infinite product expansions
  - General idea applies to counting, summing, …

- Idea: set initial value, then loop to update
  - e.g., add numbers 1 through 5:

```
sum = 0 # initialize sum (accumulator variable)
for number in range(1, 6):
    sum = sum + number # update sum
```

- Applied in text to find pi two different ways:
  - Leibniz Formula – summation of terms (p.58)
  - Wallis Formula – product of terms (p. 60)

Liebniz Formula

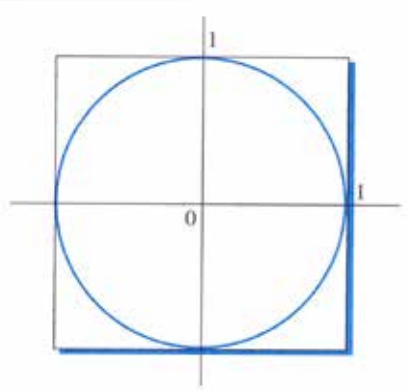
$$1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \cdots = \frac{\pi}{4}.$$

- So, the formula suggests:

$\pi = 4 \,.\, \Sigma\, (-1)^n \,.\, [1 \,/\, (2n + 1)]$ *as n goes from 0* $\rightarrow \infty$

"Monte Carlo Simulation"

- Name refers to use of randomness to see effects
  - Used in many situations – traffic flows, bank queues, …



- In the case of finding pi –
  imagine throwing darts at a unit circle (`r=1`) inscribed in a square
  - Circle area is $\pi r^2 = \pi$
  - Square area is `2 * 2 = 4`
  - So if **`n`** darts hit the square, how many darts (`k`) should land inside the circle by chance alone?
  - Answer: `k = n * π/4`. So π `= 4 * k/n`

- See Listing 2.5

Random Values

- "Pseudorandom" values available by special functions in most programming languages
  - Based on very large numbers and memory overflow
- In Python use functions of the `random` module
  - Simplest is `random.random()` – returns a floating point value between 0.0 and 1.0
  - Also `randrange(n)`, `randint(low, high)`, `shuffle(list)` and many others
  - Try `help(random)` to learn more … and *play* with it
- Listing 2.5 uses `random()` for x, y dart locations

Boolean Expressions

- Expressions that evaluate to `True` or `False`
- Relational operators: `<  <=  >  >=  ==  !=`

```
9 > 7
```



```
4 != 4
```



```
8.5 <= 7 + 3.2
```


- Beware `==` or `!=` with floating point numbers

```
100/3 == 33.3333
```


  - Instead compare absolute difference to a small value

```
abs(100/3 - 33.3333) < 0.0001
```


Compound Boolean Expressions

- Logical operators: `and`, `or`, `not`
- Their operands are Boolean values:

```
True and False
7 < 9 and 100 > 10
True or False
400 / 10 == 92 or 8 > 3
not True
not 6 > 150
```

False
True
True
True
False
True

- Special Python feature: `low <= value <= high`
  - See other behavior notes in Table 2.2 (p. 66)

Next

Character data and strings

YOUR TO-DOs

- Read **Chapter 3**
- Finish **Homework2** (due **Thursday 4/20**)
- Prepare for **Lab2**

- Study for Midterm #1!!!!

- Be cool