Starting chapter 2

An ancient problem: finding π

- Ratio of a circle’s circumference to its diameter

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

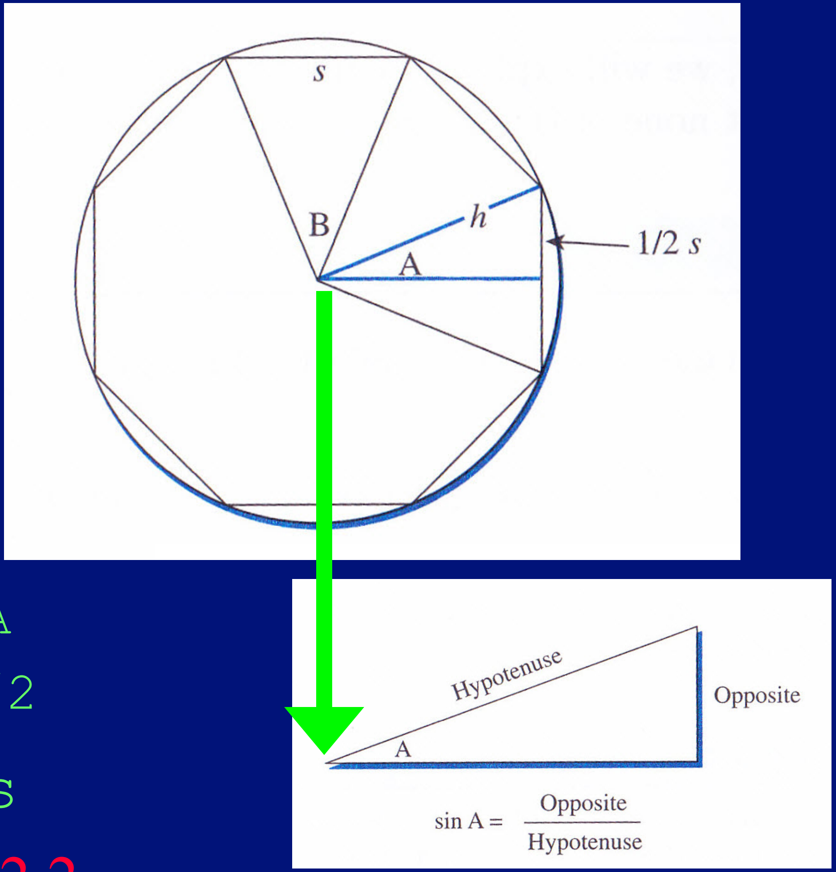
- Irrational number: an infinite series of non-repeating digits
  - So it can never be represented exactly, only *approximated*
- Chapter 2 explores various ways to approximate pi
  - But just to teach problem-solving. For calculating, use `math.pi`:

```
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



Accumulator Pattern

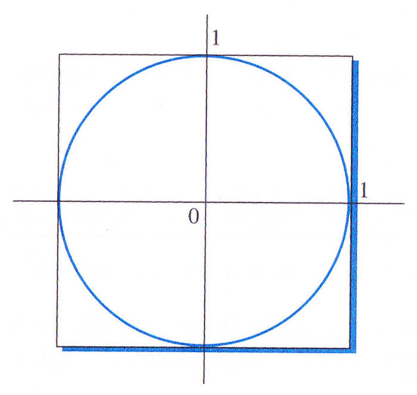
- Introduced by other ways to find pi – infinite series and infinite product expansions
  - General idea applies to counting, summing, …
- Idea: set initial value, then loop to update
  - e.g., add numbers 1 through 5:

```
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)

“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: $k = n * \pi/4$. So $\pi = 4 * k/n$



- See Listing 2.5 – but first random, Boolean, and `if`

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              ← True
4 != 4             ← False
8.5 <= 7 + 3.2     ← True
```

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

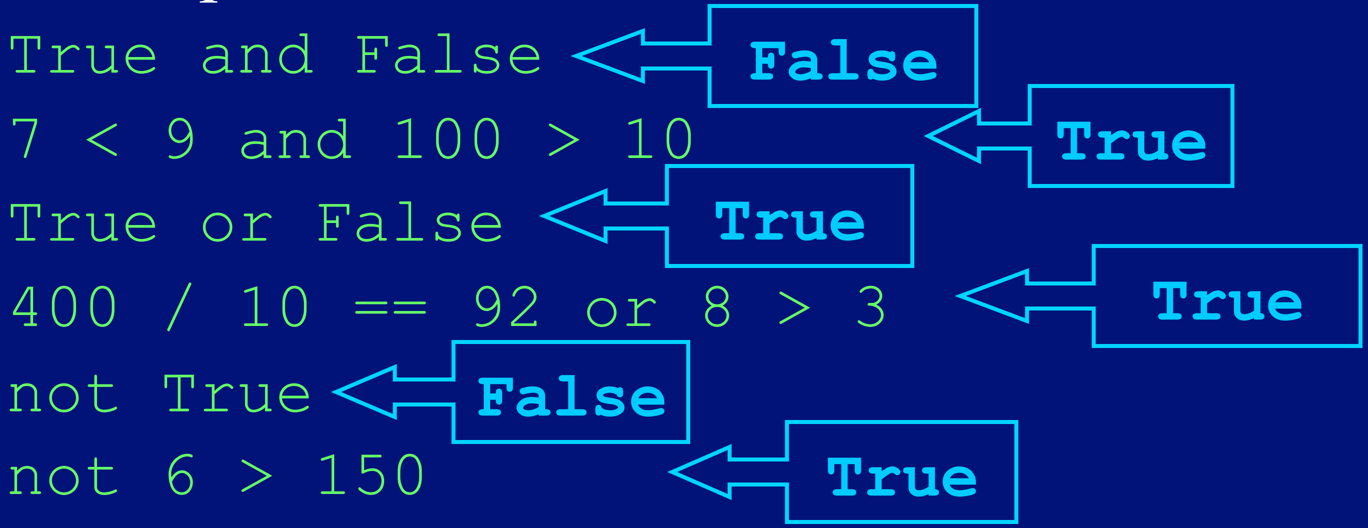
```
100/3 == 33.3333   ← False
```

  – Instead compare absolute difference to a small value

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

Compound Boolean Expressions

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



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

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

Selection statements

- `if` *Boolean expression is True*`:`

  `#` block executes if expression true

  `#` block is skipped otherwise

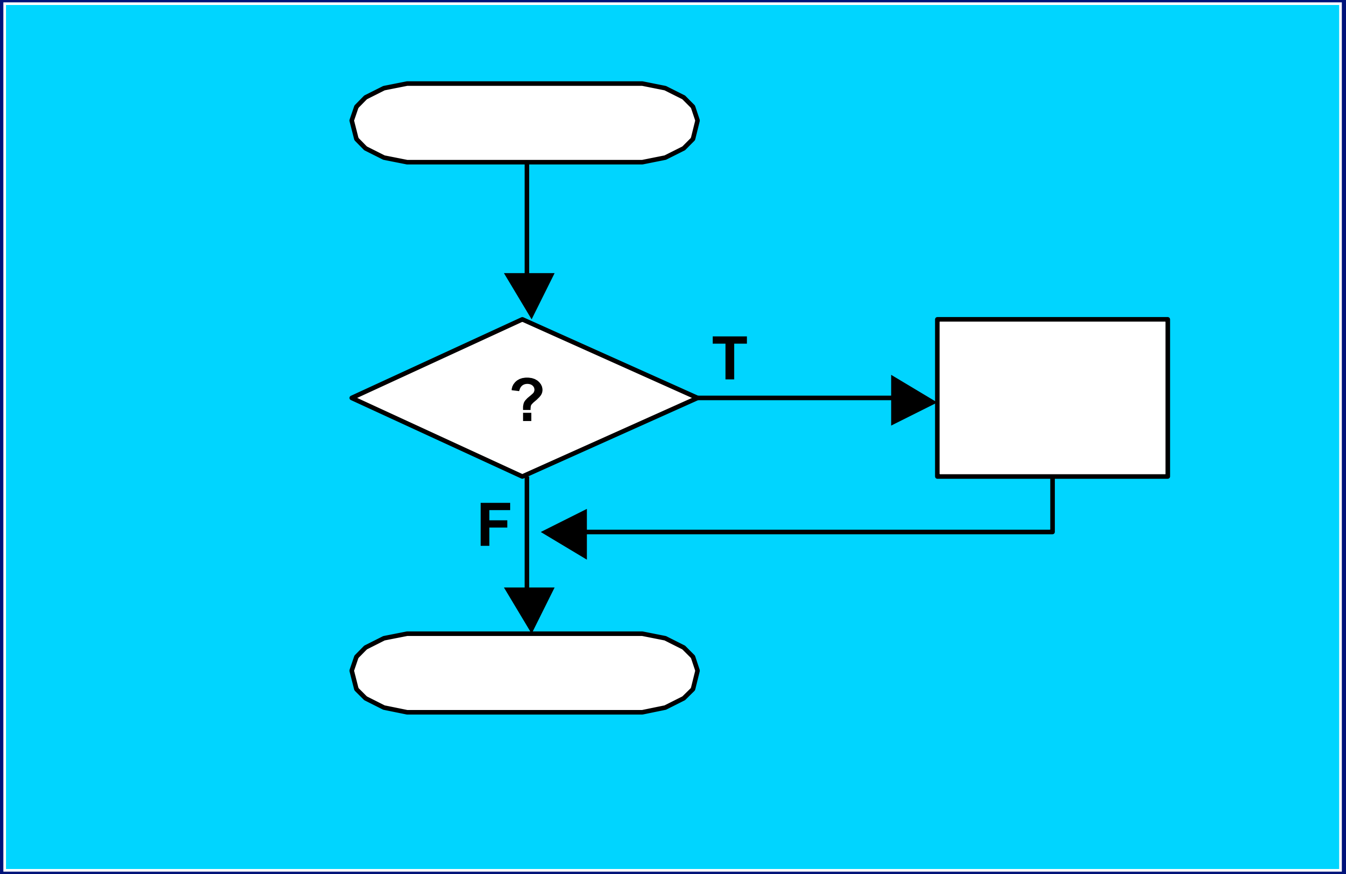
  ```
  if dice == 7 or dice == 11:
      print(“You win!”)
  ```

- See/run demo montePi.py (combined listings 2.5 and 2.6)
- Also can use an optional else clause

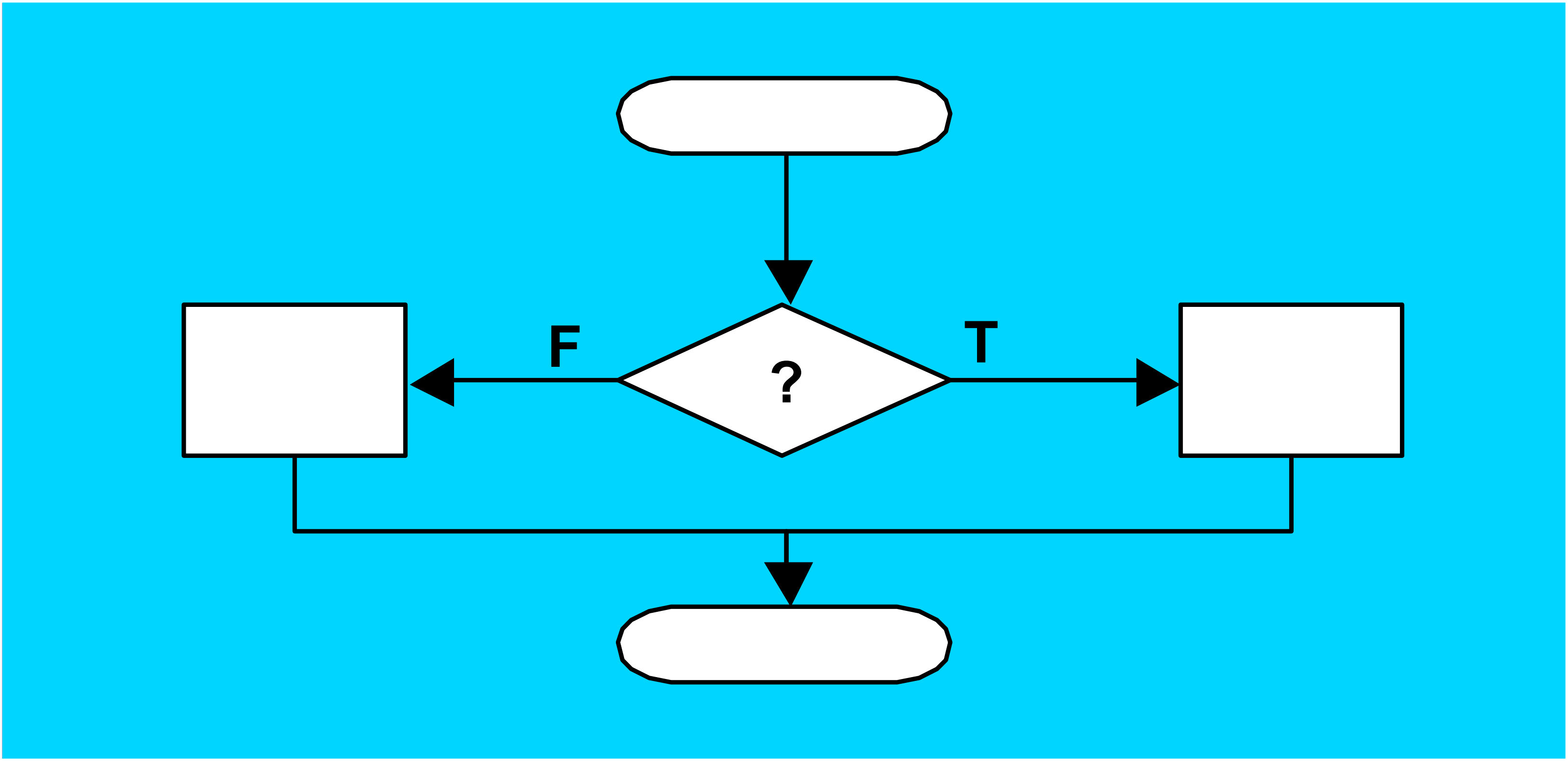
  ```
  else:
      print(“You don’t win yet.”)
  ```

  - Says what to do if expression evaluates to false
- Can summarize how it works by “flow charts”

if Selection Structure



if/else Selection Structure



Nested selection

```
else:
    if dice < 4 or dice == 12:
        print(“You lose.”)
```

  - Only evaluated when first expression is false

- So common, there is a shortcut notation:

```
elif dice < 4 or dice == 12:
```

- Any `else` that follows matches up with `if` at same level of indentation
  - Note – this rule avoids “dangling else” problem encountered frequently in other languages

Next

Character data and strings