math module

Constants

```
math.pi
math.tau →     2π
math.e
math.inf x = 10.0 THEN  9 times: x *= x  →    math.inf
math.nan math.inf / math.inf  →  math.nan
```

Trig Functions

- Degrees must be in radians.
  - `Whole circle = 2π radians`
  - `So: 180° = π    90°= π / 2`

- Convert between degrees and radians
  - `math.degrees(3.14159)  close to 180`
  - `math.radians(180)      close to π`

- The Trig functions
  ```
  sin 90°  =  1
  cos 180° =  -1
  tan 45°  =  1
  ```

  - **sin, cos, tan**
    math.sin(math.radians(90)) → 1.0
    math.cos(math.radians(180)) → -1.0
    math.tan(math.radians(45)) → 0.9999999999

  - **asin, acos, atan**
    math.degrees(math.asin(1)) → 90.0
    math.degrees(math.acos(-1)) → 180.0
    math.degrees(math.atan(1)) → 45.0

- **isclose**
  a and b are close if they diff by less that 1e-09 (can change tolerance)
  math.isclose(1.233, 1.4566) → False
  math.isclose(1.233, 1.233) → *True*
  math.isclose(1.233, 1.233000001) → True (5 zeros)
  math.isclose(1.233, 1.23300001) → False (4 zeros)

- **ciel, floor**
  math.ceil(12.0001) → 13
  math.ceil(3.99999) → 4

  math.floor(12.0001) → 12
  math.floor(3.99999) → 3

- **comb**

  math.comb(n, k) returns n choose k, number of ways to select k items from a group of n items, without repetition and without order.

$$\binom{n}{k} = \begin{cases} \dfrac{n!}{k!\,(n-k)!} & k \le n \\ 0 & k > n \end{cases}$$

math.comb(4,2) → 6

- **sqrt**

  math.sqrt(25) → 5.0

- **hypot**

  math.hypot(x, y) returns $\sqrt{x^2 + y^2}$

- **dist** – Distance between two points

  math.dist((1, 2), (1, 3)) → 1.0

Log Functions

- **log(*x* [, *base*])** default base is e

  math.log(125,5) → 3.000004
  math.log(32,2) → 5.0

- **log10(x)** Log base 10

  math.log10(1000) → 3.0

- **log2(x)** Log base 2

  math.log2(32) → 5.0

1. Archimedes' Approximation of π

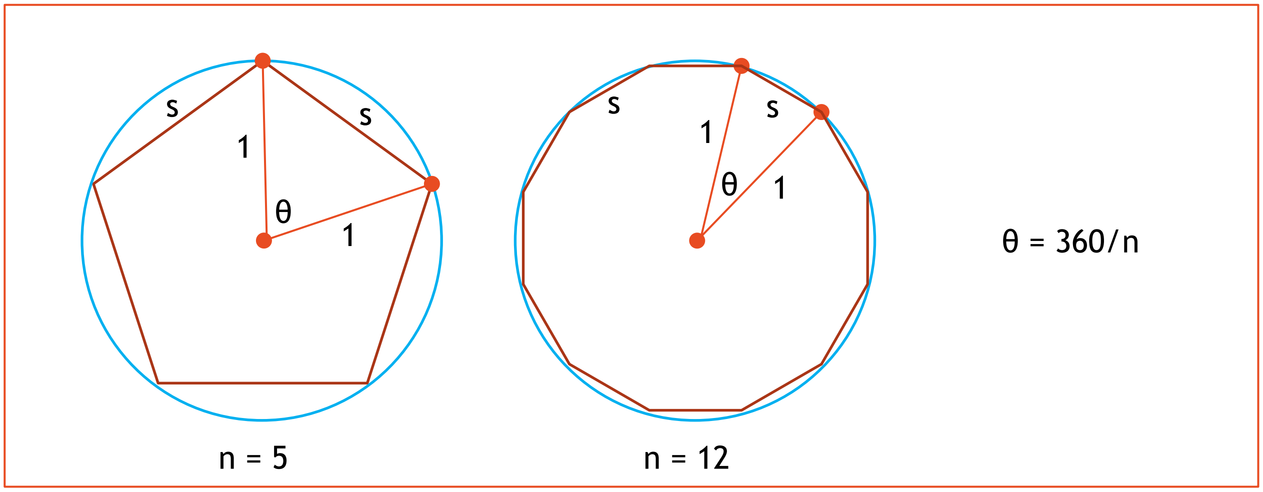
Good Example of 80% thinking, 20% coding.
In this case 95% thinking, 5% coding.

To approximate π - Approximate a Circle

- We know c = 2πr
- When r = 1, c = 2π, which means $\pi = \frac{c}{2}$

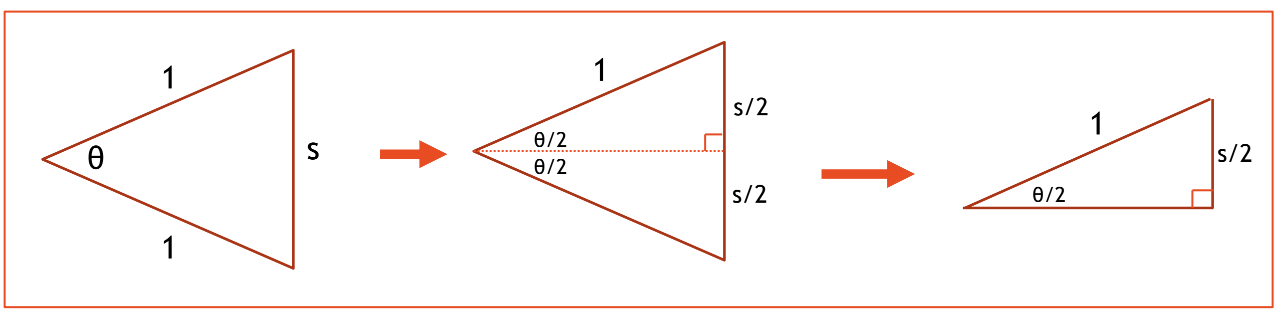
Approximate **circumference c** of a circle with **radius 1**

Using an **n-sided** regular polygon (input: n, larger n → better π approximation)
Each side has **length s**, as n↑ s↓



c ≈ polyCirc = s × n

So, **given n, find s**



Find s

$$
\begin{aligned}
\sin\left(\frac{\theta}{2}\right) &= \frac{s}{2} \\
\sin\left(\frac{1}{2}\theta\right) &= \frac{s}{2} \\
\sin\left(\frac{1}{2} \times \frac{360}{n}\right) &= \frac{s}{2} \\
\sin\left(\frac{180}{n}\right) &= \frac{s}{2} \\
s &= 2\sin\left(\frac{180}{n}\right)
\end{aligned}
$$

Now plug formula for s into our π approximation equation

$$
\begin{aligned}
\pi &= \frac{c}{2} \\
&\approx \frac{n \times s}{2} \\
&= \frac{n\left(\cancel{2}\sin\left(\frac{180}{n}\right)\right)}{\cancel{2}} \\
&= n\sin\left(\frac{180}{n}\right)
\end{aligned}
$$

Finally, write a python function

```
def archimedes(numSides):
    return numSides * math.sin(math.radians(180/numSides))
```

02-01-Archimedes Pi Approximation.py

Section 2.5 Accumulator Approximations (Miller 3rd ed)

Accumulator Pattern

1. Initialize accumulator
2. Add to the accumulator, typically with a for loop
3. At the end, accumulator holds the result

Sum the numbers from 1 to n

Start by *printing* the numbers from 1 to n.
Use a range

n = 10
list(range(1, n + 1)) → [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

Add them:

```
acc = 0
for num in range(1, n+1)
    acc += num
```

Look at the answer

acc → 55

Python has a built-in function for this

sum(range(1, n+1)) → 55

Product of the numbers from 1 to n

```
n = 10
acc = 1
for num in range(1, n+1)
    acc *= num
```

acc → 3628800

Can use math module

math.factorial(10) → 3628800