Leibniz Approximation of π

$$\pi = \frac{4}{1} - \frac{4}{3} + \frac{4}{5} - \frac{4}{7} + \frac{4}{9} - \frac{4}{11} \cdots$$

Value of i	0	1	2	3	4	5
term	$+\frac{4}{1}$	$-\frac{4}{3}$	$+\frac{4}{5}$	$-\frac{4}{7}$	$+\frac{4}{9}$	$-\frac{4}{11}$

Observations:

1. Numerator is always 4
2. Denominator starts at 1 and goes up by 2
3. Sign starts + and alternates

First Leibniz Function

02-02-Leibniz Pi Approximation.py

```
def leibniz1(numTerms):
    acc = 0

    # Values for the first term
    numerator = 4
    denominator = 1
    sign = 1

    for i in range(numTerms):
        term = sign * (numerator / denominator)
        acc += term

        # Get ready for next term
        denominator = denominator + 2
        sign = -sign

    return acc
```

A Leibniz function based on the **value** of i

Recall

$$\pi = \frac{4}{1} - \frac{4}{3} + \frac{4}{5} - \frac{4}{7} + \frac{4}{9} - \frac{4}{11} \cdots$$

Then

$$term\ i = \frac{4 \cdot (-1)^i}{2i + 1}$$

The first 4 Leibniz terms:

i	$\mathbf{\frac{4 \cdot (-1)^i}{2i + 1}}$
0	$\frac{4 \cdot (-1)^0}{2 \cdot 0 + 1} = \frac{4}{1}$
1	$\frac{4 \cdot (-1)^1}{2 \cdot 1 + 1} = \frac{-4}{3}$
2	$\frac{4 \cdot (-1)^2}{2 \cdot 2 + 1} = \frac{-4}{5}$
3	$\frac{4 \cdot (-1)^3}{2 \cdot 3 + 1} = \frac{-4}{7}$

Second Leibniz Function

```
def leibniz2(numTerms):
    acc = 0

    for i in range(numTerms):
        term = (4 * (-1) ** i) / (2 * i + 1)
        acc += term

    return acc
```

Formatting Strings with f-strings

- Example 1 – Field widths
- Example 2 – Justification
    - < Left
    - ^ Center
    - > Right
- Example 3 – Add commas
- Example 4 – Floating point
    - Width not specified
    - 2 decimal places
- Example 5
    - Field width of 7
    - 2 decimal places

02-03-String Formatting – f-strings.py

Wallis Approximation of π

$$\frac{\pi}{2} = \frac{2}{1} \times \frac{2}{3} \times \frac{4}{3} \times \frac{4}{5} \times \frac{6}{5} \times \frac{6}{7} \times \frac{8}{7} \times \frac{8}{9} \cdots$$

This is a product, not a sum → Accumulator will start at 1

Look at terms in pairs

$$\frac{\pi}{2} = \left[\frac{2}{1} \times \frac{2}{3}\right] \times \left[\frac{4}{3} \times \frac{4}{5}\right] \times \left[\frac{6}{5} \times \frac{6}{7}\right] \times \left[\frac{8}{7} \times \frac{8}{9}\right] \cdots$$

For **each pair**

1. Numerator starts at 2 and goes up by 2
2. Left-Denominator = numerator - 1
3. Right-Denominator = numerator + 1

First Wallis Function

02-04-Wallis Pi Approximation.py

```
def wallis1(numPairs):
    acc = 1
    numerator = 2

    for pair in range(numPairs):
        leftTerm = numerator / (numerator − 1)
        rightTerm = numerator / (numerator + 1)

        acc = acc ∗ leftTerm ∗ rightTerm

        numerator += 2

    pi = 2 ∗ acc
    return pi
```

A Wallis function based on the **value** of a loop index i

Recall

$$\frac{\pi}{2} = \left[\frac{2}{1} \times \frac{2}{3}\right] \times \left[\frac{4}{3} \times \frac{4}{5}\right] \times \left[\frac{6}{5} \times \frac{6}{7}\right] \times \left[\frac{8}{7} \times \frac{8}{9}\right] \cdots$$

Value of i	0	1	2	3
Pairs	$\frac{2}{1} \times \frac{2}{3}$	$\frac{4}{3} \times \frac{4}{5}$	$\frac{6}{5} \times \frac{6}{7}$	$\frac{8}{7} \times \frac{8}{9}$

For pair i

Numerator	=	2i + 2
Left denominator	=	2i + 1
Right denominator	=	2i + 3

The first Wallis 4 pairs:

i	$\frac{\mathbf{2}\boldsymbol{i}+\mathbf{2}}{\mathbf{2}\boldsymbol{i}+\mathbf{1}} \times \frac{\mathbf{2}\boldsymbol{i}+\mathbf{2}}{\mathbf{2}\boldsymbol{i}+\mathbf{3}}$	
0	$\frac{2\cdot\mathbf{0}+2}{2\cdot\mathbf{0}+1} \times \frac{2\cdot\mathbf{0}+2}{2\cdot\mathbf{0}+3}$	$\frac{2}{1} \times \frac{2}{3}$
1	$\frac{2\cdot\mathbf{1}+2}{2\cdot\mathbf{1}+1} \times \frac{2\cdot\mathbf{1}+2}{2\cdot\mathbf{1}+3}$	$\frac{4}{3} \times \frac{4}{5}$
2	$\frac{2\cdot\mathbf{2}+2}{2\cdot\mathbf{2}+1} \times \frac{2\cdot\mathbf{2}+2}{2\cdot\mathbf{2}+3}$	$\frac{6}{5} \times \frac{6}{7}$
3	$\frac{2\cdot\mathbf{3}+2}{2\cdot\mathbf{3}+1} \times \frac{2\cdot\mathbf{3}+2}{2\cdot\mathbf{3}+3}$	$\frac{8}{7} \times \frac{8}{9}$

Second Wallis Function

02-04 - Wallis Pi Approximation.py - again

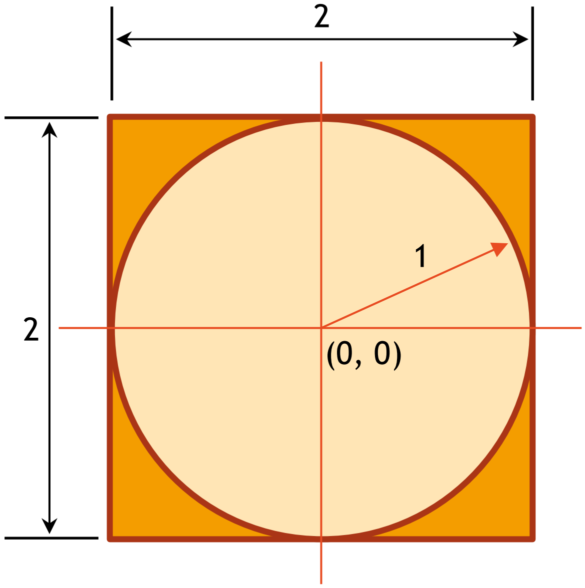
```
def wallis2(numPairs):
    acc = 1
    for i in range(numPairs + 1):
        leftTerm = (2 * i + 2)/(2 * i + 1)
        rightTerm = (2 * i + 2)/(2 * i + 3)
        acc = acc * leftTerm * rightTerm

    pi = 2 * acc
    return pi
```

Section 2.6 Monte Carlo Simulation (Miller 3rd ed)

Simulation Overview

1. Uses randomness to come up with π approximation
2. Idea: Simulate throwing darts at a dartboard
    a. Count the number thrown vs. number that land in the circle



$$Square\ Area = 2 \times 2 = 4$$

$$\text{Circle Area} = \pi r^2 = \pi \times 1^2 = \pi$$

$$\frac{Circle\ Area}{Square\ Area} = \frac{\pi}{4}$$

$$\pi = 4 \cdot \frac{Circle\ Area}{Square\ Area}$$

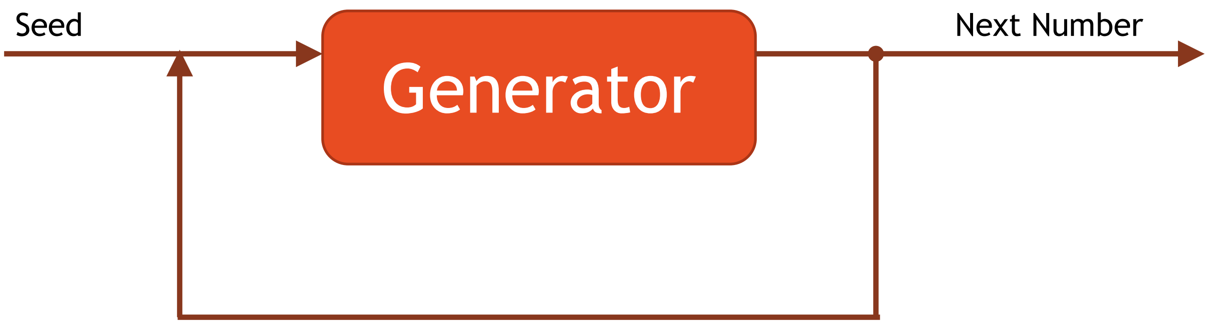
Our Function will:

1. Throw darts that will randomly land on the 2 × 2 board
    a. Some will land **inside** the circle → numInCircle
    b. Some will land **outside** the circle → numThrown

$$\pi \approx 4 \cdot \frac{numInCircle}{numThrown}$$

A lot to do first…

Pseudo Random Number Generators



In Python Shell

1. **import random**

2. **random.random()**
   a. Returns a float from half open interval [0, 1)

   random.random()
   Do a few times
   Put in loop to print 10 times

3. **random.uniform(a, b)**
   a. Returns a float from closed interval [a, b]

   random.uniform(5, 10)
   random.uniform(1.25, 1.29)

4. **random.randrange(start, stop, interval)**
   a. Returns number from the range

   list(range(2,9,3)) → [2, 5, 8]
   random.randrange(2,9,3)

   Do several times

5. **random.randint(a, b)**
   a. Returns integer from closed interval [a, b]

   random.randint(1, 1) → Always returns 1
   random.randint(0, 1) → Flip a coin
   random.randint(1, 6) → Roll a die

6. **random.seed()**
   a. No argument – Uses current time
   b. Can provide initial seed

```
random.seed(1210)

for i in range(5):
   print(random.randint(1, 100))
```