The random module

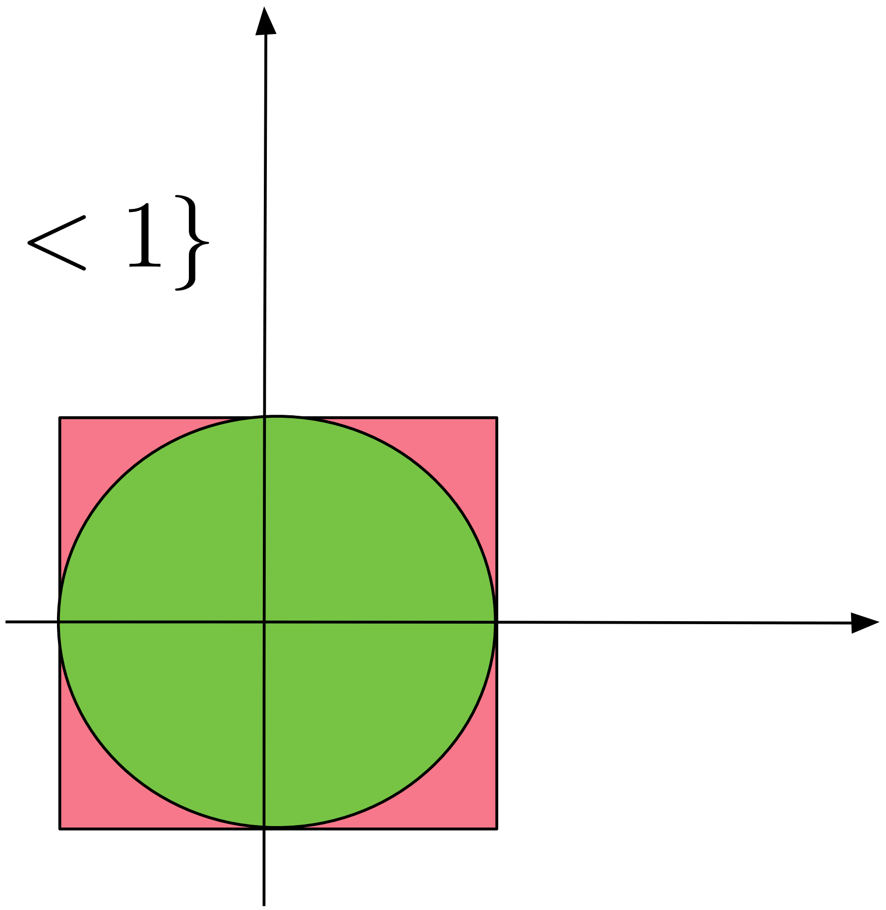
Python
Marquette University

A Monte Carlo Method for Area calculation

- Calculate the area of a circle of radius 1
  - Can be done analytically: $$A = r^2 \cdot \pi$$
  - Can be done with <u>Monte Carlo Method</u>
    - Use pseudo-random numbers in order to determine values probabilistically
    - Named after Stanislav Ulam
      - Used for work on the thermo-nuclear device

A Monte Carlo Method for Area calculation

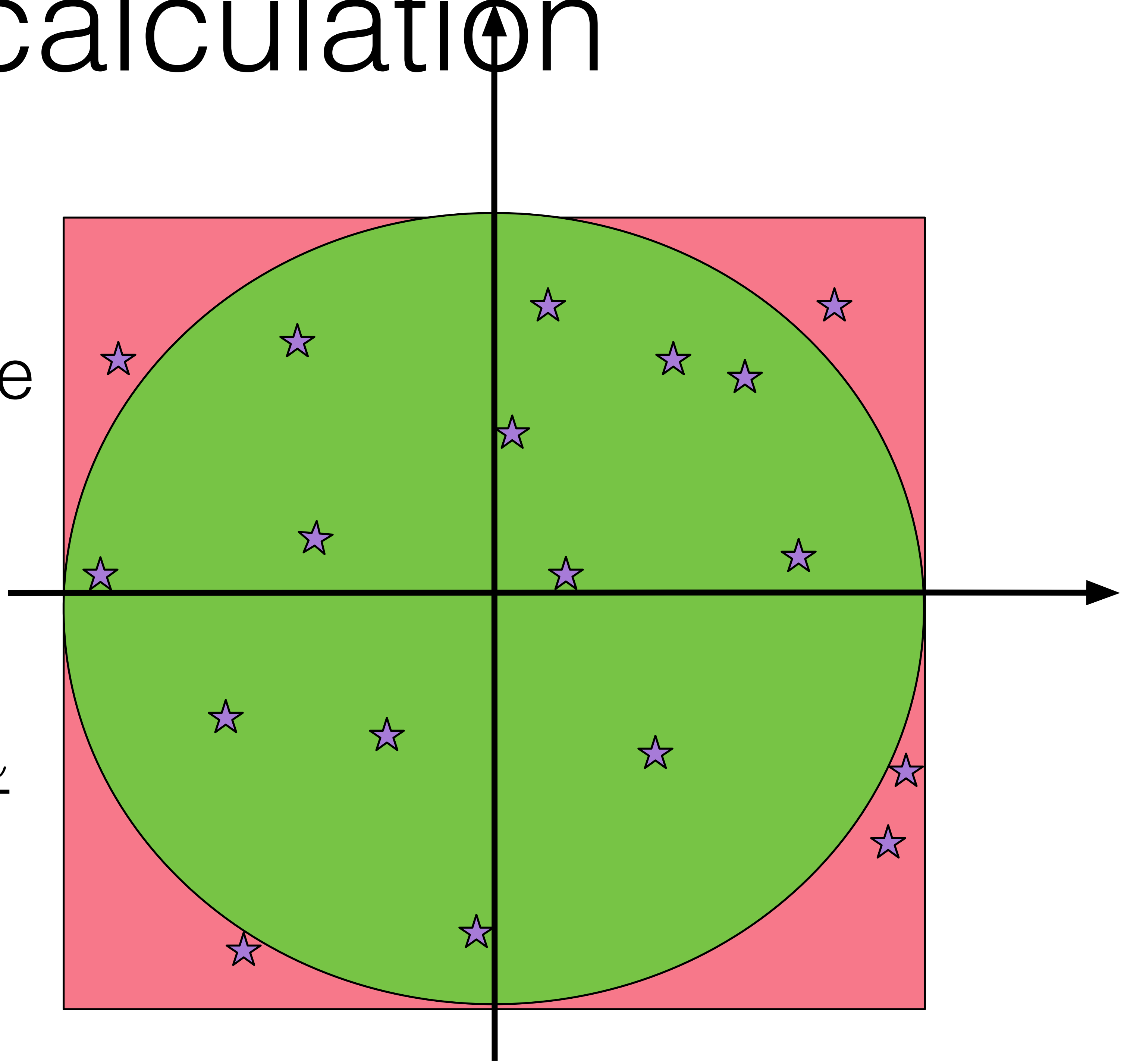
- Inscribe Circle with a square
- Circle: $\{(x,y)|x^2+y^2<1\}$
- Square: $\{(x,y)|-1<x<1,-1<y<1\}$



A Monte Carlo Method for Area calculation

- Method:
  - Choose *n* random points in the square
  - *m* points inside circe

$$\frac{\text{Area of Circle}}{\text{Area of Square}} \approx \frac{m}{n}$$



Random Number Generation

- Computers are deterministic (one hopes) and using a deterministic device to generate randomness is not possible
  - Modern systems can use physical phenomena
    - Geiger counters for radioactive materials
    - Atmospheric radio noise
  - But for large sets of seemingly random numbers, use pseudo-random number generators
    - Create deterministically based on a seemingly random seed output that passes statistical tests for randomness

Random Number Generation in Python

- Sophisticated methods to generate seemingly random sequences of numbers
- Part of a <u>module</u> called random

Interlude: Python Modules

- Anyone can create a python module
  - Just a file with extension .py
  - In a directory in the Python path, which is set for the OS
  - Or just in the same directory as files that use the module
- A module contains definitions of variables and functions
  - Any python script that <u>imports</u> the module can use them

Interlude: Python Modules

- Predefined modules
  - Python defines many modules
    - We already have seen math and os
- To use such a module, say
  - `import random`
    - in order to use the functions within random

Interlude: Python Modules

- If I just import the module random, then I can use its functions by prefixing "random."
    -

imp.py - /Users/thoma

```python
import random

for _ in range(10):
    print(random.random())
```

Using the function random inside the

Interlude: Python Modules

- If I want to avoid writing the module name I can use an “as” clause that redefines the name of the module within the script

imp.py - /Users/tho

```
import random as rd

for _ in range(10):
    print(rd.random())
```

Using the same function in the same m
but now after internally renaming the

Interlude: Python Modules

- By using the “from — import” clause, I can use variables and functions without repeating the module name

imp.py - /Users/thomasschwarz/Docu

```
from random import uniform, randint

for _ in range(10):
    print(uniform(0,2), randint(0,10))
```

```
Importing the two functions uniform
the random module.
```

Interlude: Python Modules

- I could even import everything from a module
  - But this can create havoc if I defined a function with the same name as a function in the module

imp.py - /Users/thomasschwarz/Do

```
from random import *

for _ in range(10):
    print(uniform(0,2), randint(0,10))
```

A dangerous practice: Importing all functions from a module

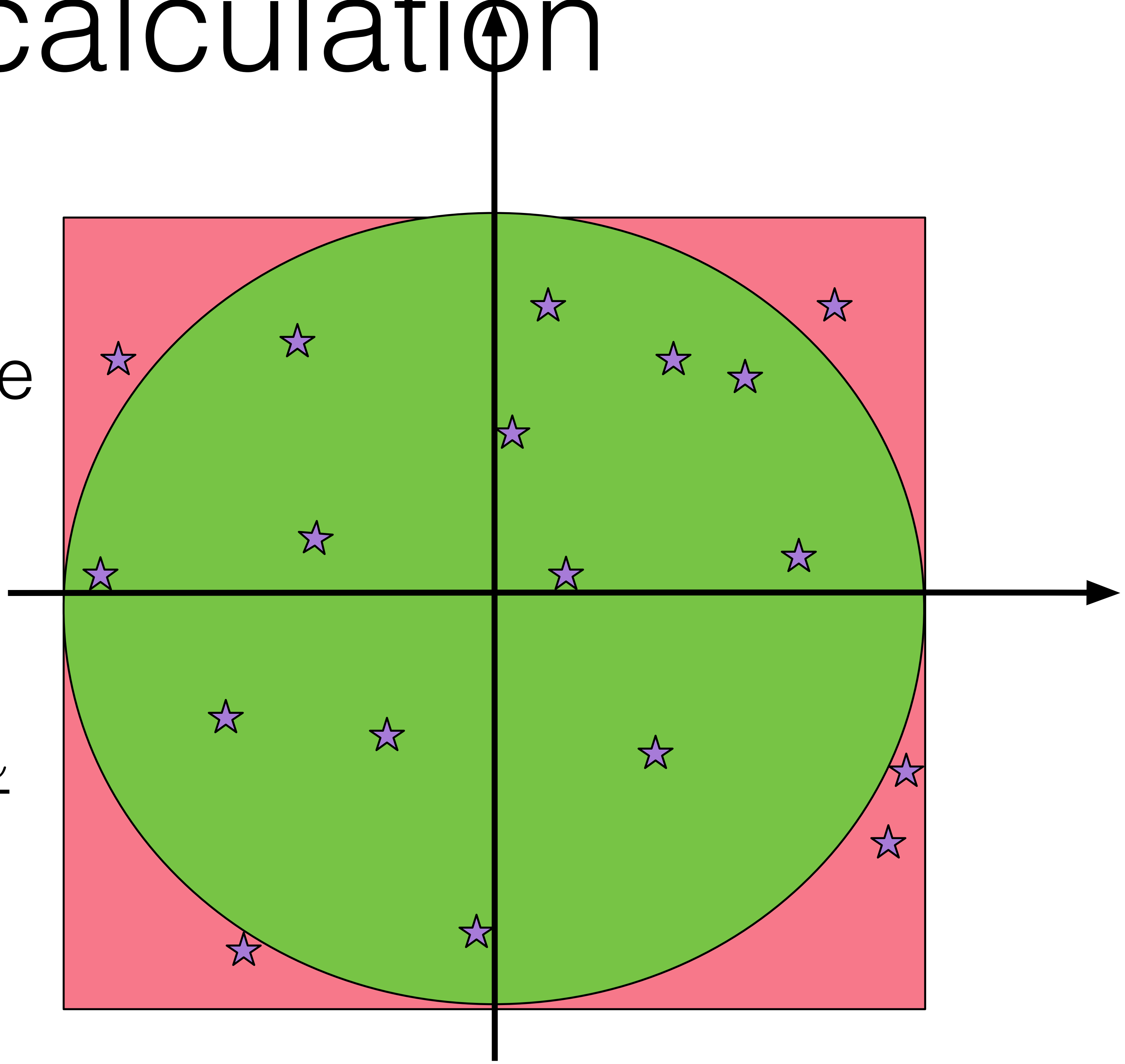
Random Module

- Important functions in the random module
  - `random.randint(a, b)` Selects a random integer between a and b (boundaries included)
  - random.uniform(a, b) Selects a random float between a and b
  - random.random( ) Selects a random number between 0 and 1

A Monte Carlo Method for Area calculation

- Method:
  - Choose *n* random points in the square
  - *m* points inside circe

$$\frac{\text{Area of Circle}}{\text{Area of Square}} \approx \frac{m}{n}$$



A Monte Carlo Method for Area calculation

- Use random module
  - `random.uniform(-1,1)` generates random number between -1 and 1
  - Generating *20* random numbers:

```
import random

for i in range(20):
    x = random.uniform(-1,1)
    y = random.uniform(-1,1)
    print("({:6.3f},{:6.3f})".format(x,y))
```

A Monte Carlo Method for Area calculation

- We then only count those that are inside the circle

```
import random

def approx(N):
    count = 0
    for i in range(N):
        x = random.uniform(-1,1)
        y = random.uniform(-1,1)
        if x*x+y*y<1:
            count += 1
    return (4*count/N)
```

A Monte Carlo Method for Area Calculations

- Since $\frac{\text{count}}{N} \approx \frac{\text{Area Circle}}{\text{Area Box}}$ and the area of the box is 4
- we return $\frac{4\text{count}}{N}$

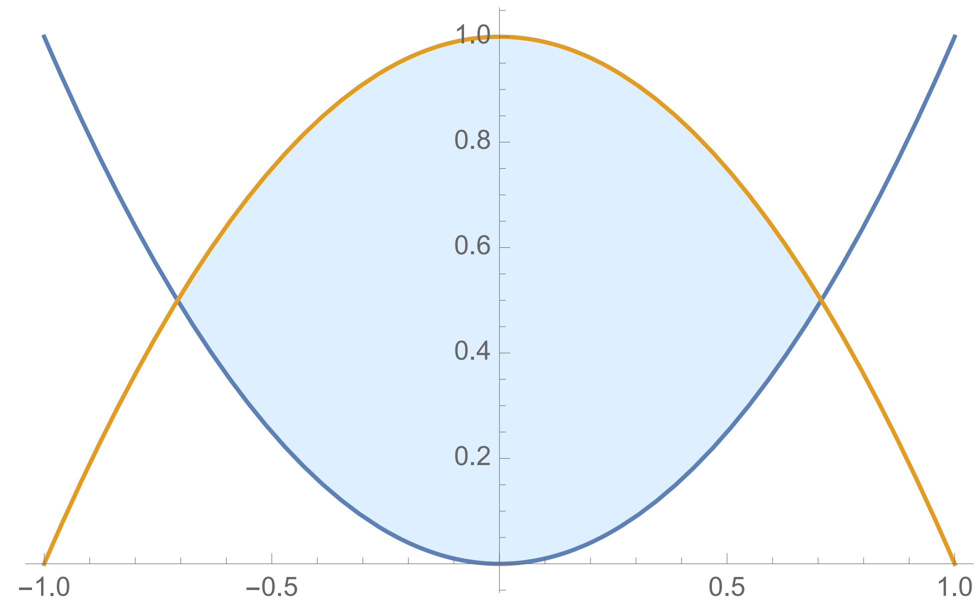
```
import random

def approx(N):
    count = 0
    for i in range(N):
        x = random.uniform(-1,1)
        y = random.uniform(-1,1)
        if x*x+y*y<1:
            count += 1
    return (4*count/N)
```

A Monte Carlo Method for Area calculation

- Need few random point to get a general idea
- Need lots to get any good accuracy
- Method of choice used to determine 6-dimensional integrals for simulation of quantum decay where accuracy is not as important as speed

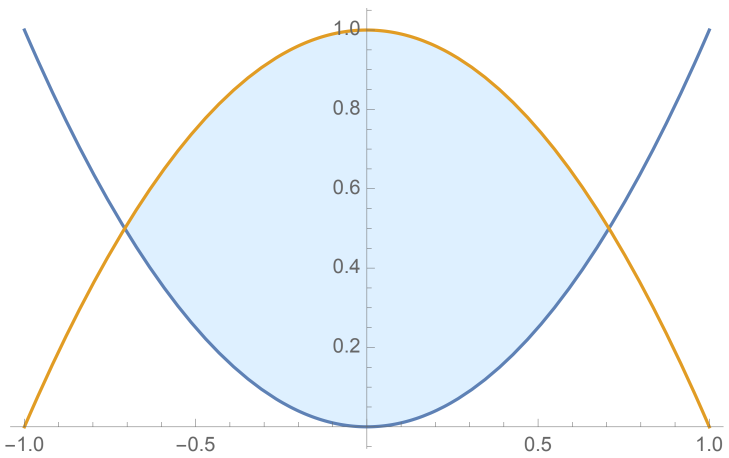
A Monte Carlo Method for Area calculation

- Your task:
  - Determine the area between the curves
    $$y = x^2$$
    $$y = 1 - x^2$$



  - Hint: We draw points in the rectangle [-1,1] x [0,1]
  - (x,y) lies in the area if
    $$x^2 < y < 1 - x^2$$

A Monte Carlo Method for Area calculation



Select random points in the box [-1,1] x [0,1]

Count the number of times that the point falls in the area

Multiply the ratio count / #pts by the area of the box, which is 2

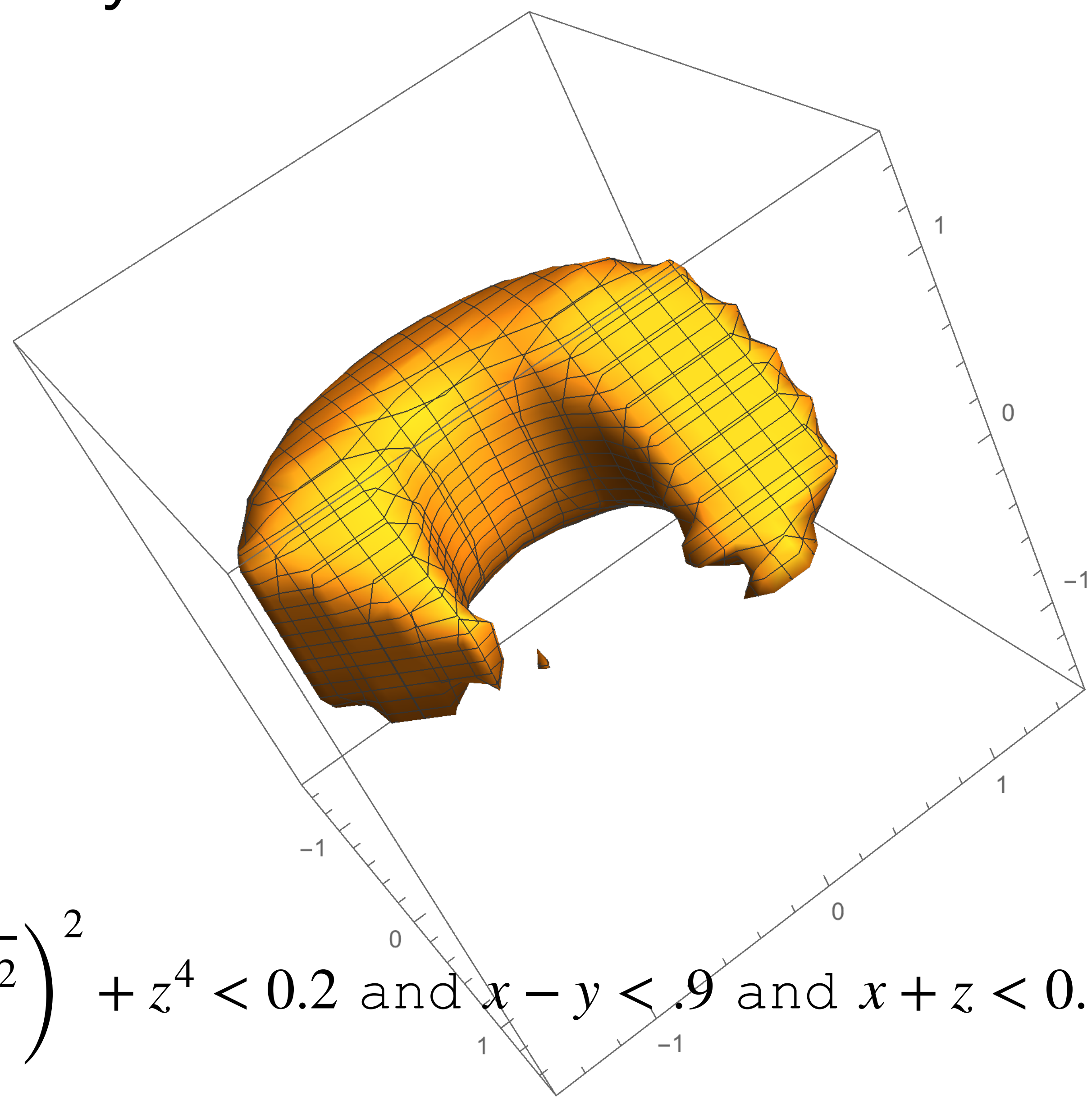
```
import random

N = int(input("Give the number of random points: "))
count = 0
for _ in range(N):
    x = random.uniform(-1,1)
    y = random.uniform(0,1)
    if x*x < y < 1-x*x:
        count += 1
print("The area is approximately", count*2/N)
```

Monte-Carlo Volume Calculation

- Sometimes, Monte-Carlo is the method of choice
  - When there is no need for super-precision
  - When the volume is not easily evaluated using analytic methods.

Volume Calculation

- A partially eaten donut



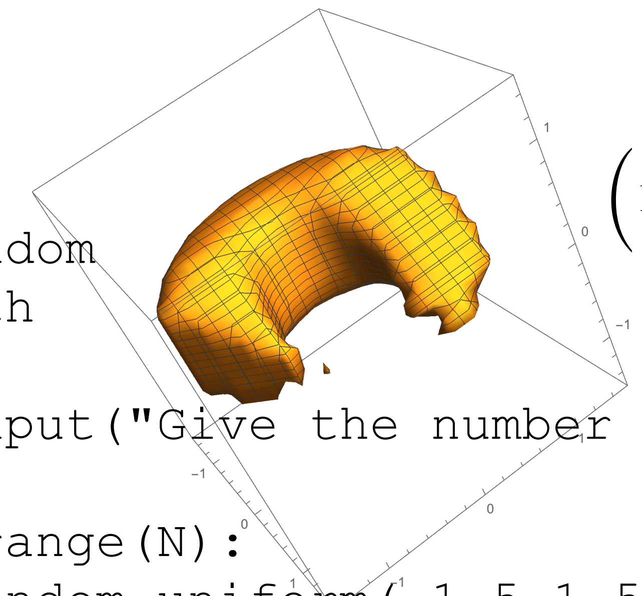
$$\left(1-\sqrt{x^2+y^2}\right)^2+z^4<0.2 \text{ and } x-y<.9 \text{ and } x+z<0.1 \text{ and } x+y<1.8$$

Volume Calculation

- Monte Carlo:
  - Select random points in the box $-1.5<x<1.5$, $-1.5<y<1.5$, $-1.5<z<1.5$.
  - Check whether they are inside the donut
  - Count over total number is approximately area of donut over area of box (which is 9).

Volume Calculation

- A partially eaten donut



$$\left(1-\sqrt{x^2+y^2}\right)^2+z^4<0.2 \text{ and } x-y<.9 \text{ and } x+z<0.1 \text{ and } x+y<1.8$$

```
import random
import math

N = int(input("Give the number of random points: "))
count = 0
for _ in range(N):
    x = random.uniform(-1.5,1.5)
    y = random.uniform(-1.5,1.5)
    z = random.uniform(-1.5,1.5)
    if (1-math.sqrt(x**2+y**2))**2+z**4<0.2 and x-y<0.9 and x+z<0.1 and x+y<1.8:
        count += 1
print("The area is approximately", count*27/N)
```

Additional Exercises

- Find the area of

$$\{(x,y)|(x-2)^2+3*(y-1)^2<1\}$$

  - Hint: First determine maximum and minimum values for x and y