02-math

January 27, 2017

1 Math in Python

2 math module

- standard math functions like sin, exp, etc.
- constants
  - math.e
  - math.pi

```
In [1]: import math
        dir(math)
```

```
Out[1]: ['__doc__',
         '__file__',
         '__loader__',
         '__name__',
         '__package__',
         '__spec__',
         'acos',
         'acosh',
         'asin',
         'asinh',
         'atan',
         'atan2',
         'atanh',
         'ceil',
         'copysign',
         'cos',
         'cosh',
         'degrees',
         'e',
         'erf',
         'erfc',
         'exp',
         'expm1',
         'fabs',
         'factorial',
         'floor',
         'fmod',
         'frexp',
         'fsum',
         'gamma',
         'gcd',
         'hypot',
         'inf',
         'isclose',
         'isfinite',
         'isinf',
         'isnan',
         'ldexp',
         'lgamma',
         'log',
         'log10',
         'log1p',
         'log2',
         'modf',
         'nan',
         'pi',
         'pow',
         'radians',
         'sin',
         'sinh',
         'sqrt',
         'tan',
         'tanh',
         'trunc']
```

```
In [2]: # sin, cos, etc take radians

        [math.radians(90), math.sin(math.radians(90)), math.sin(math.pi), math.exp
```

```
Out[2]: [1.5707963267948966, 1.0, 1.2246467991473532e-16, 2.718281828459045]
```

```
In [3]: [math.sqrt(2), math.pow(2,3), math.modf(3.2), math.e, math.pi]
```

```
Out[3]: [1.4142135623730951,
         8.0,
         (0.20000000000000018, 3.0),
         2.718281828459045,
         3.141592653589793]
```

3 Sympy

- symbolic math package
- module that can be loaded into any Python and coexist with other code

```
In [4]: from sympy import *

In [5]: x = symbols('x')

In [6]: expand ((5*x+3)*(1-x)*(1+6*x))

Out[6]: -30*x**3 + 7*x**2 + 20*x + 3

In [7]: solve(-30*x**3 + 7*x**2 + 20*x + 3, x)

Out[7]: [-3/5, -1/6, 1]

In [8]: integrate(1/(1+x**3),x)

Out[8]: log(x + 1)/3 - log(x**2 - x + 1)/6 + sqrt(3)*atan(2*sqrt(3)*x/3 - sqrt(3)/3

In [9]: # will solve this below

        expand ((x-2)*(x-3))

Out[9]: x**2 - 5*x + 6
```

4 Example

- solve quadratic equation a*x**2 + b*x + c = 0

```
In [10]: def quad(a,b,c):
             disc = math.sqrt(b*b - 4*a*c)
             return [(-b+disc)/(2*a),
                     (-b-disc)/(2*a)]

         # find the roots of poly expanded above

         quad(1,-5,6)

Out[10]: [3.0, 2.0]

In [11]: # this equation bombs - x**2 + 1 = 0
         # the roots are +i, -i
         # but math.sqrt doesn't work with a negative argument

         quad(1,0,1)


        ---------------------------------------------------------------------------

        ValueError                                Traceback (most recent call last)

        <ipython-input-11-6492b156a353> in <module>()
          3 # but math.sqrt doesn't work with a negative argument
          4
    ----> 5 quad(1,0,1)


        <ipython-input-10-ba2903888381> in quad(a, b, c)
          1 def quad(a,b,c):
    ----> 2     disc = math.sqrt(b*b - 4*a*c)
          3     return [(-b+disc)/(2*a),
          4             (-b-disc)/(2*a)]
          5


        ValueError: math domain error
```

5 Complex math module

- similar to math module, but knows about complex numbers

```
In [3]: import cmath

        cmath.sqrt(-1)
```

```
Out[3]: 1j
```

```
In [13]: # now can handle imaginary roots

         def quadc(a,b,c):
             disc = cmath.sqrt(b*b - 4*a*c)
             return [(-b+disc)/(2*a),
                     (-b-disc)/(2*a)]

         quadc(1, 0, 1)
```

```
Out[13]: [1j, -1j]
```

5.0.1 Euler's famous equation

```
- the five most important numbers in mathematics in the same equation
```

6

$$e^{j*\pi} + 1 = 0$$

```
In [4]: cmath.exp(1j * cmath.pi) + 1
```

```
Out[4]: 1.2246467991473532e-16j
```

```
In [14]: dir(cmath)
```

```
Out[14]: ['__doc__',
          '__file__',
          '__loader__',
          '__name__',
          '__package__',
          '__spec__',
          'acos',
          'acosh',
          'asin',
          'asinh',
          'atan',
          'atanh',
          'cos',
          'cosh',
          'e',
          'exp',
          'isclose',
          'isfinite',
          'isinf',
          'isnan',
          'log',
          'log10',
          'phase',
          'pi',
          'polar',
          'rect',
          'sin',
          'sinh',
          'sqrt',
          'tan',
          'tanh']
```

7 mpmath module

- Python has arbitrary precision integer arithmetic built in
- mpmath does arbitrary precision floating point

```
In [15]: import mpmath
```

```
In [16]: for decs in range(2, 100, 20):
             # dps is number of decimals
             mpmath.mp.dps = decs
             mps = str(mpmath.mpf(1.)/mpmath.mpf(19))
             print(mps)
```

```
0.053
0.05263157894736842105263
0.052631578947368421052631578947368421052632 
```

```
0.0526315789473684210526315789473684210526315789473684210526315789
0.052631578947368421052631578947368421052631578947368421052631578947368421052631578
```

8 random

```
- various flavors of randomness
```

```
In [17]: import random
```

```
In [18]: [random.randint(4, 9) for j in range(25)]
```

```
Out[18]: [4, 8, 5, 6, 9, 7, 4, 7, 8, 7, 7, 8, 4, 6, 7, 7, 8, 7, 4, 7, 5, 5, 9, 5, 8
```

```
In [19]: [random.choice(['sci', 'eng', 'art', 'lit']) for j in range(20)]
```

```
Out[19]: ['eng',
          'sci',
          'eng',
          'art',
          'lit',
          'art',
          'art',
          'eng',
          'sci',
          'sci',
          'art',
          'sci',
          'eng',
          'art',
          'eng',
          'lit',
          'lit',
          'sci',
          'eng',
          'eng']
```

```
In [20]: [random.uniform(10,20) for j in range(20)]
```

```
Out[20]: [19.501591714615294,
          15.006559341030107,
          15.209716219324331,
          16.38801935641188,
          12.797049438605168,
          12.580307121789307,
          17.27747514203035,
          13.989727799576318,
          17.45186452494507,
          12.4211732597255,
          19.201392521137482,
          14.89278157648604,
          17.046619802087044,
          14.574006660069566,
          16.068568197143836,
          12.028044323265126,
          19.391871312513075,
          17.94642725581442,
          17.982716174694705,
          10.39665132966006]
```

```
In [21]: [random.gauss(0, 1) for j in range(20)]
```

```
Out[21]: [1.1112902893808643,
          -0.05524675259929446,
          -0.7000250247920489,
          -1.913546432555811,
          1.2321934665887677,
          1.0952635849158074,
          0.15678947364085383,
          0.29623059801663276,
          -0.03780424654542534,
          -0.4148453357666101,
          -0.013551658640150373,
          0.7486078744027823,
          -1.4961811635425741,
          -0.35674525088013204,
          -0.556321400428154,
          -0.7572503652693081,
          1.7512337933900286,
          -1.0494281032010189,
          2.5017400278571555,
          0.9439899047872942]
```

```
In [22]: random.sample(range(20), 10)
```

```
Out[22]: [4, 10, 11, 13, 18, 15, 16, 6, 7, 0]
```

9 Sage

- large application
- integrates some 90 math packages
- goal is to be an open source Mathematica
- has a web notebook interface similar to Jupyter notebooks

10 SciPy

- extensive collection of math, special functions, linear algebra, statistics, signal processing, numerical analysis
- SciPy doc
- SciKits doc

11 Machine Learning

- Scikit-Learn
  - excellent Machine Learning package
- TensorFlow
  - has a Python API
  - very popular