CS357: Python Tutorial

August 30, 2018

Python

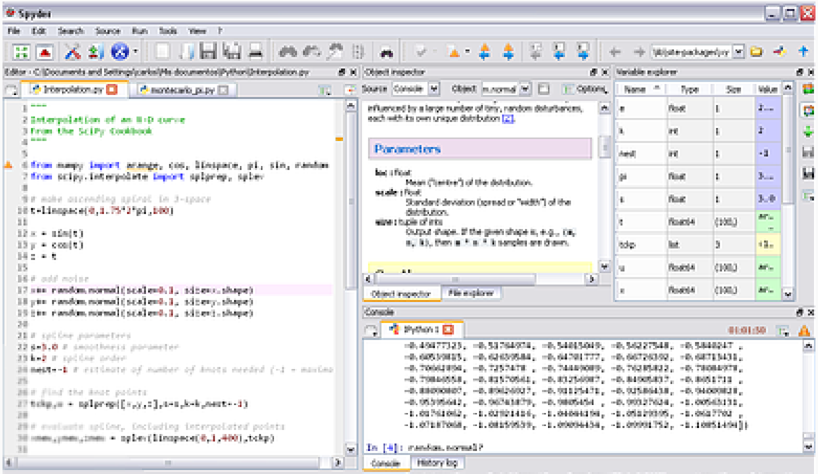
- High Level
- Interpreted
- Excellent libraries

Tools

- Interpreter
- Code editor
- Optional: IDE

Optional Tool: IDE

- Spyder



```
"""
Interpolation of an N-D curve
From the SciPy Cookbook
"""

from numpy import arange, cos, linspace, pi, sin, random
from scipy.interpolate import splprep, splev

# make ascending spiral in 3-space
t=linspace(0,1.75*2*pi,100)

x = sin(t)
y = cos(t)
z = t

# add noise
x+= random.normal(scale=0.1, size=x.shape)
y+= random.normal(scale=0.1, size=y.shape)
z+= random.normal(scale=0.1, size=z.shape)

# spline parameters
s=3.0 # smoothness parameter
k=2 # spline order
nest=-1 # estimate of number of knots needed (-1 = maximal

# find the knot points
tckp,u = splprep([x,y,z],s=s,k=k,nest=-1)

# evaluate spline, including interpolated points
xnew,ynew,znew = splev(linspace(0,1,400),tckp)
```

influenced by a large number of tiny, random disturbances, each with its own unique distribution [2].

Parameters

loc : float
    Mean ("centre") of the distribution.
scale : float
    Standard deviation (spread or "width") of the distribution.
size : tuple of ints
    Output shape. If the given shape is, e.g., (m, n, k), then m * n * k samples are drawn.

Name	Type	Size	Value
e	float	1	2....
k	int	1	2
nest	int	1	-1
pi	float	1	3....
s	float	1	3.0
t	float64	(100,)	ar...
tckp	list	3	<li...
u	float64	(100,)	ar...
x	float64	(100,)	ar...

```
       -0.49477323, -0.53764974, -0.54815049, -0.56227548, -0.5848247 ,
       -0.60539835, -0.62639584, -0.64781777, -0.66726392, -0.68713431,
       -0.70662894, -0.7257478 , -0.74449089, -0.76285822, -0.78084978,
       -0.79846558, -0.81570641, -0.83256087, -0.84905837, -0.8651713 ,
       -0.88090807, -0.89626927, -0.91125471, -0.92586438, -0.94009828,
       -0.95395642, -0.96743879, -0.9805454 , -0.99327624, -1.00563131,
       -1.01761062, -1.02921416, -1.04044194, -1.05129395, -1.0617702 ,
       -1.07187068, -1.08159539, -1.09094434, -1.09991752, -1.10851494])

In [4]: random.normal?
```

Permissions: RW   End-of-lines: LF   Encoding: UTF-8-GUESSED   Line: 17   Column: 18

Python Comments

- Single line
- Multi-line

Python Datatypes

- Integers
- Floating-Point Numbers
- Complex Numbers
- Booleans
- Strings
- None

Variables

- Dynamic types
- Easy conversion
  - Numeric types convert implicitly
  - You can typecast to convert to/from strings
- No arbitrary limit to a number's magnitude

Printing

- print is a function, so it needs parentheses `print()`
- Don't need to tell print what type is coming
- Format strings are your friend
    - Put variable values in a larger statement
    - Precise control of appearance
    - `https://docs.python.org/3.6/library/string.html`

Data Structures

- Lists
- Tuples

Lists

- Ordered collection of arbitrary items
- Grows and shrinks
- Mutable (items can change)
- Common operations:
  - Get an item
  - Add/remove/set items
  - Check for inclusion

Tuples

- Fixed, ordered collection of items
- Immutable (cannot change)
- Common operations:
    - Get an item
    - Unpacking

Simple Conditions

If Statement

```
if <condition>:
    <body>
```

- relational ops: `>`, `<`, `>=`, `<=`, `==`, `!=`
- boolean ops: `and`, `or`, `not`

```
>>> 3 > 5
False
>>> 2 == 1 or 'spam' == 'spam'
True
>>> (not (3 != 3)) and (5 >= 5)
True
```

Example

```
>>> x1 = 1; x2 = 0; x3 = 4
>>> maxval = x1
>>> if x2 > maxval:
...     maxval = x2
... if x3 > maxval:
...     maxval = x3
...
>>> maxval
4
```

- Note: whitespace matters

Two-Way Decisions

```
>>> x1 = 1; x2 = 0; x3 = 4
>>> if x >= x2:
...     if x1 > = x3:
...         maxval = x1
...     else:
...         maxval = x3
... else:
...     if x2 >= x3:
...         maxval = x2
...     else:
...         maxval = x3
...
>>> maxval
4
```

Multi-Way Decisions

```
>>> x1 = 1; x2 = 0; x3 = 4
>>> if x1 >= x2 and x1 >= x3:
...     maxval = x1
... elif x2 >= x1 and x2 >= x3:
...     maxval = x2
... else:
...     maxval = x3
...
>>> maxval
4
```

Definite Loops

For Loop

```
for <var> in <sequence>:
    <body>
```

```
>>> seq = ['egg', 'and', 'spam']
>>> for item in seq:
...     print(item, end=' ')
...
egg and spam
```

For Loops and `range()`

- Range function
    - Range is its own type
    - Typically when used for looping
    - `range(stop)` Looping from 0 (inclusive) to stop (exclusive)
    - `range(start, stop[, step])` Looping from start (inclusive) to stop (exclusive) by step
    - Can typecast to list to get list of integers
      `list(range(stop)) -> list of integers`

```
>>> for i in range(3):
...     print(i)
...
0
1
2
```

Nesting

```
>>> for i in range(3):
...     for j in range(2):
...         print('({}, {})'.format(i,j))
...
(0, 0)
(0, 1)
(1, 0)
(1, 1)
(2, 0)
(2, 1)
```

Indefinite Loops

While Loop

```
while <condition>:
    <body>
```

- `break`: breaks out of immediate containing loop
- `continue`: continues with next iteration
- `else`: executes upon exhaustion of for loop or when while condition becomes false

Example

```
>>> for n in range(2, 7):
...     for x in range(2, n):
...         if n % x == 0:
...             msg = '{} equals {} * {}'
...             print(msg.format(n, x, n/x)
...             break
...     else:
...         msg = '{} is a prime number'
...         print(msg.format(n))
...
2 is a prime number
3 is a prime number
4 equals 2 * 2
5 is a prime number
6 equals 2 * 3
```

Sentinel Loop

```
r = .5
n = 0
sol = 2.
tol = 1e-5
curr = 0
while abs(sol - curr) > tol:
    curr += r**n
    n += 1
print('Converged in {} iterations'.format(n))
```

[4]

Functions

Function Definition

```
def <name>(<formal-parameters>):
    <body>
```

- All arguments are pass by value
- Some objects are mutable

```
>>> def knight(reps):
...     for i in range(reps):
...         print('Ni!')
...
>>> knight(3)
Ni!
Ni!
Ni!
```

Example

```
def convert(deg, celsius=True):
    if celsius:
        return (9 / 5.) * (deg + 32)
    else:
        return (5 / 9.) * (deg - 32)
```

```
>>> convert(0)
32
>>> convert(100, False)
212
```

Example

```
def convert_all(degs, celsius=True):
    condegs = []
    for item in degs:
        condegs.append(convert(degs, celsius))

    return condegs
```

```
>>> lst = [0, 100]
>>> f = convert_all(lst)
>>> f
[32, 212]
>>> convert_all(convert_all(lst), False)
[0, 100]
```

Example

```
def convert_all(degs, celsius=True):
    for i in range(len(degs)):
        degs[i] = convert(degs[i], celsius)
```

```
>>> lst = [0, 100]
>>> convert_all(lst)
>>> lst
[0, 212]
>>> convert_all(lst, False)
>>> lst
[0, 100]
```

Numpy

- What does it provide?
    - Arrays
    - Important functions - dot product, matrix multiplication, transpose, etc.
    - Methods that operate on entire arrays
- More efficient than lists - very fast
- Examples

Scipy

- Nice numerical functions
  - Linear algebra
  - Interpolation
  - Optimization
  - Signal Processing
  - FFT
  - Integration
  - Sparse matrices
- Input/output functions

Matplotlib

- Useful plotting library
- Plotting similar to Matlab
- What can it do?
  - Plot basic graphs
  - Multiple lines per graph
  - Titles, labels, legends
  - Multiple graphs in a plot
- Code examples

References

References

[1] matplotlib: python plotting. URL `http://matplotlib.org/`.

[2] Numpy and scipy documentation. URL `http://docs.scipy.org/doc/`.

[3] Python docs. URL `http://docs.python.org/2/`.

[4] John M. Zelle. *Python Programming: An Introduction to Computer Science*. Franklin, Beedle & Associates, Inc., Wilsonville, Oregon, 2004.