Python Tutorial

Credits to CS357 (UIUC)

January 7, 2020

Python

- High Level
- Interpreted
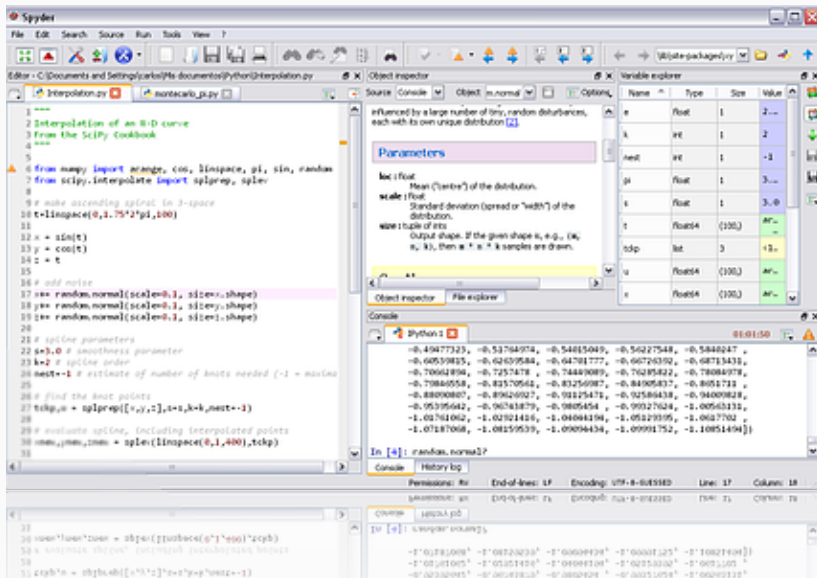- Excellent libraries

Tools

IP[y]:
IPython

- Interpreter
- Code editor
- Optional: IDE

Vim

emacs

Optional Tool: IDE

- Spyder

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

- 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)
```

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

[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.