Python Programming Techniques

Eliot Feibush

PICSciE
Princeton Institute for Computational Science and Engineering

Log in with your netID
Teaching Assistants

Allen Lu
Brianna Yao
Versatile

Very efficient for user / programmer.
sample1.py

```
x = 0.
xmax = 10.
xincr = 2.

while x < xmax:  # Here is a block of code
    y = x * x
    print(x, y)
x += xincr
```
Example

No variable declaration.
No memory allocation.
No compiling, no .o or .obj files
No linking.

No kidding - Just run.
Mac
Magnifying glass: idle (idle.app)

Python 3.6
IDLE (Python GUI)

Command line from terminal also possible.

Windows
Start Menu

Python
IDLE
(Python GUI)
Browser based IDE

https://repl.it/languages/python3
Interpreter
Integrated Development Environment -- idle

Everything that a program can have:

Variables
Strings
Lists
Expressions
Import modules

Great for learning & trying new lines of code
idle

IDE – Integrated Development Environment
   Color-coded syntax
   Statement completion
   Interpreter retains “scope” after program ends

Written in Python with tkinter GUI module.

IDLE ➔ Preferences
   Font, Keys
   History-previous: up-arrow
   History-next: down-arrow
Try out the interpreter

Python 3.6.5
>>> 2+3
5
>>> a = 5.1
>>> b = 6.2
>>> print (a*b)
31.62
help()

dir()

type()

>>> help()

# interpreter

help> keywords

# if, else, for ...

help> symbols

# + - = / ...

help> modules

# math, os, sys

help> topics

# USE UPPER CASE

Python Rosetta Stone
Variables

Case sensitive

- start is not the same as Start
- count is not the same as Count
- \( R = \frac{1}{r} \)

Start with a letter, not a number
Long names OK
Types and Operators

int          # scalar variable, holds a single value
float
long
complex     a = (3 + 4j)       # type(a)

+    -    *    /    %    //    **    # Arithmetic operators
+=    -=    *=    /=

<    <=    >    >=    ==    !=    # Comparison operators
+    # has magic overload abilities!
Casts

int()
long()
float()

hex()       # string representation
oct()       # string representation

str()       # for printing numbers + strings
## Built-in Constants

<table>
<thead>
<tr>
<th>Value</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>&lt;type ‘bool’&gt;</td>
</tr>
<tr>
<td>False</td>
<td>&lt;type ‘bool’&gt;</td>
</tr>
<tr>
<td>None</td>
<td>&lt;type ‘NoneType’&gt;</td>
</tr>
</tbody>
</table>
Indenting Counts!

Indent 4 spaces or a tab -- be consistent

: at end of line indicates start of code block
  requires next line to be indented

Code block ends with an outdent

Code runs but not as desired – check your indents
Program

Loops
Conditionals, Control
Functions
Keywords

Control
if  else  elif
while  break  continue
and  or  not

>>> help()
help > keywords
idle: File ➔ New File
Save ➔ command-s
Run ➔ Run Module ➔ F5 ➔ key

Python 3.6.2 (v3.6.2:5fd33b5926, Jul 16 2017, 20:11:06)
[GCC 4.2.1 (Apple Inc. build 5666) (dot 3)] on darwin
Type "copyright", "credits" or "license()" for more information.
>>> WARNING: The version of Tcl/Tk (8.5.9) in use may be unstable.

-------------- RESTART: /Users/efeibush/Documents/Untitled.py ---------------
15
>>>
Programming Exercise

Write a python program that converts degrees to radians for:

0, 10, 20, 30, ... 180 degrees

edit and save: deg.py
Run F5: deg.py

radians = degrees * 3.14 / 180.
print(degrees, radians)
Debugging Tip

IDLE shell retains variables in scope after running program:

```python
dir()
print(degree)
```
Comments

in line text after # is ignored

# can be in any column

Text within triple quotes

""" This is a multi-line comment that will be compiled to a string but will not execute anything. It is code so it must conform to indenting """"
s = "shrubbery"
print(s)

len(s)
Strings

Sequence of characters such as  \( s = \text{“abcdefg”} \)
Indexed with [ ] starting at [0]
  \( s[0] \) is a,  \( s[1] \) is b

\( s[\ -1 \ ] \) refers to last character in string.
  Negative indexing starts at last character.

Use \( s[p:q] \) for string **slicing**.
  \( s[3:] \) evaluated as “defg”
  \( s[:3] \) evaluated as “abc” *up to but not 3*
  \( s[1:-2] \) evaluated as “bcde”
    *up to but not including -2*
String Concatenation

first = ‘John’
last = ‘Cleese’

full = first + " " + last

sp = " "
full = first + sp + last
+ Operator is Operand “Aware”

>>> "water" + "fall"   # concatenate

>>> 3 + 5           # addition

________________________________________________________________________________________

>>> 3 + "George"    # unsupported type

>>> "George" + 3    # TypeError
pi = 3.14159
print ('The answer is ' + str(pi))
# cast float to string to avoid TypeError
The Immutable String

Can’t replace characters in a string.

```python
s = "abcd"
```

```python
s[1] = "g"  # Object does not support item assignment
```

```python
s = "agcd"  # re-assign entire string
```
Automatic Memory Management

malloc() — realloc() — free()

char name[32]

name = “as long as you want”

len(name)  # len() function is part of __builtins__
a = 3

if a > 0:
    print ("a is positive")
elif a < 0:
    print( "a is negative")
else:
    print ("a = 0")
String Exercise

Degrees to radians:

Print column titles
Right align degree values
Limit radians to 7 characters

Reminder: len(s)
str  Under the Hood

str – is a Class! Not just a memory area of characters
Object oriented programming
Encapsulated data and methods
Use the dot . to address methods and data
a = "hello"
a.upper()  # returns "HELLO"

`type(a)`
`dir(str)`
`help(str)`

hidden methods start with __
import math
dir(math)

math.sqrt(x)
math.sin(x)
math.cos(x)

from math import *
dir()

sqrt(x)

from math import pi
dir()

print pi
Keywords for Inclusion

import    from    as
import math

Exercise

Degrees to radians and now cosine:

Use `math.pi` for defined constant
Use `math.cos(radian)` to compute cosine
Print cosine in 3\textsuperscript{rd} column

\textit{Align cosine to decimal point}

\textit{(Do not truncate the cosine)}
Data Structures

*Resemble arrays in other languages*

List `[]`  # ordered sequence of stuff

Tuple `( )`  # n-tuple, immutable

Dictionary `{ }`  # key-value pairs
Lists [ ]

Indexed from [0]
Last index is [-1] or length - 1

Class object with its own methods, e.g.
  .append()
  .sort()

Magic slice operator :
Magic iter() function actually __iter__()

min() max() are builtins
Declare a List

x = [59, 50, 42, 34, 23, 14]

x.append(4)  # works in place, no return

Identify the sequence? Next item?

x.append(“Spring St”, “Canal St”)

x[3] = “Penn Station”

# list is mutable, can replace values

x = []  # create empty list, then append to it

x = list()
List methods

append()
extend()
insert()
remove()
sort()  # in place, does not return a new list
reverse()  # in place
index()
count()

cList = aList + bList  # concatenate lists
range() Function

range(stop)  # assumes start=0 and incr=1
range(start, stop)  # assumes incr=1
range(start, stop, incr)

Returns sequence of integers, up to, but not including stop.

Python 2 returns a list.
Python 3 returns a "range class" to save memory.
Both give you an iterable sequence.

range() is a built-in function:  dir(__builtins__)
Keywords Looping with range()

for i in range(10):
    for i in dayList:
List Techniques

d = list(range(4))  # [0, 1, 2, 3]
d = [0] * 4  # [0, 0, 0, 0]

d = [-1 for x in range(4)]  # [-1, -1, -1, -1]

List Comprehension
Lists Exercise

Degrees to radians, cosines, and now lists:

Create a list of radians and a list of cosines
Print the lists
Use a range() loop instead of while
Plot Exercise

Degrees to radians, cosines, lists, now plot:

Plot a curve: x axis: radians, y axis: cosines

import matplotlib.pyplot as plt
plt.plot(radiansL, cosinesL)
plt.show()  # displays on screen
matplotlib + LaTeX

```python
import matplotlib.pyplot as plt
plt.rc("text", usetex=True)
    # set config to draw text with Tex
plt.xlabel(r"\textbf{Time}" )
    # draw x label “Time” in bold font
    # compare to: plt.xlabel(“Time”)

s = r"\n" # raw string has \n, not linefeed
latex.py example - requires latex installation
```
$y = \sin \frac{1}{x^2}$
del keyword

del a[3]  # deletes element at index 3

del a[2:4]  # deletes element 2 and 3
    # list slicing

del a      # deletes entire list. a is gone.
Unpack a list into variables

```python
name = ["Abe", "Lincoln"]

first, last = name

# multiple variables on left side of =
# number of variables must be len(name)
```
d = [ [0]*4 for y in range(3) ]

[[0, 0, 0, 0],
[0, 0, 0, 0],
[0, 0, 0, 0]]
N-dimensional Arrays

```python
import numpy

ndarray class – optimized to be very fast.
 Integrated with matplotlib for graphing.
```

[link to www.princeton.edu/~efeibush/Python Programming mini-course numpy numpy2016.pdf]
numpy.arange()

Note: arange can use floats for interval & step

```python
import numpy
radA = numpy.arange(1.5, 2.5, .1)
    # Returns numpy array of evenly spaced floats
    # min, max, step
for x in radA:    # can iterate on numpy array
```
numpy.linspace()

Note: linspace can use floats for interval integer for number of steps

```python
import numpy
a = numpy.linspace(1.5, 2.5, 11)
# Returns numpy array of evenly spaced floats
# min, max, number of steps
a = list(a)  # cast array to list

for x in a:
```
python Runs Your Program
Command Line version

```python
python sample1.py
```

sample1.py source code is run directly instead of compile, link, run.

No `.obj` nor `.o` files of compiled code.
No `.exe` nor `a.out` of executable code.

```python
python  -i  exdeg.py
```
import sys
print (sys.argv)

sys.argv is a list
sys.argv[0] has the name of the python file.
Subsequent locations have command line args.
Does not apply in interpreter.

>>> help(sys)
import os

fileL = []  # set up a list

for f in os.listdir('.'):  
    if f.endswith('.py'):
        print( f )
        fileL.append(f)

fileL.sort()  # list function, sort in place

print( fileL )

# much better text handling than csh or bash; shell independent

import subprocess  # Advanced
    # then use the Popen class for running programs
Defining a Function

Block of code separate from main.

Define function before calling it.

```python
def myAdd(a, b):
    # define before calling
    return a + b
```

```python
p = 25  # main section of code
q = 30
```

```python
r = myAdd(p, q)  # case sensitive
```
Keywords

Functions (methods, subroutines)

    def
    return
Define a Function Exercise

Degrees to radians, cosines, lists, now function:

Format the radians using a function call
import math  # knows where to find it

import sys
sys.path.append("/Users/efeibush/spline")
import cubic.py  # import your own code

reload – debugging your own module from the interpreter
n-Tuple

Immutable List
   Saves some memory
   Cannot be modified when passed to subroutine

aTuple = tuple(aList)  # Create from a list
   # No append, no assignment; OK to extract slice
cTuple = aTuple + bTuple  # OK to concatenate

print aTuple[0]  # index using brackets
Dictionary {} 

Key : Value
Look up table
Index by key -- Any hashable (immutable) type

\texttt{print d[key]}  \# prints value for specified key

Order of key:value pairs is not guaranteed. Good for command line arguments
name list files, nicknames, etc.

\texttt{d[key] = value}  \# to add a key-value pair
\texttt{such as d["New Jersey"] = "Trenton"}
Dictionary methods

d = { }  
d = dict()

eDict.update(gDict)  # combine dictionaries

del eDict[key]

if key in eDict:  
    print (eDict[key])

d.keys()  # returns set of all keys

d.items()  # returns set of all key:value pairs as tuples
gFile = open("myfile.txt", "r")  # built-in function

for j in gFile:  # python magic: text file iterates on lines
    print j  # print each line

gFile.close()

see readsplit.py  
str.split() 
    .split() method parses a line of text into list of words
Write a Text File

```python
f = open("myfile.txt", "w")
    # open is a built-in function
a = 1
b = 2

f.write("Here is line " + str(a) + "\n");
f.write("Next is line " + str(b) + "\n");

f.close()
    # write() and close() are file object methods
```
1. **Read, Parse, Store, Write**

```python
import sys

inF = open(sys.argv[1], "r")  # open the file specified on the command line
linesL = inF.readlines()     # read all lines of text into a list of Strings
inF.close()                  # no longer needed

from collections import OrderedDict
kvD = OrderedDict()          #kvD = {}         # does not preserve order

for lineS in linesL:
    # iterate through each line of text in the list
    wL = lineS.split()         # parse the line into words
    keyS = wL[0]               # first word is the key
    valueS = wL[2]             # third word is the value, assume w[1] is =
    kvD[keyS] = valueS         # add key-value pair to dictionary; items are strings
    print keyS, valueS

print " "
print kvD.keys()
print kvD.values()
print " "
print kvD.viewitems()
```
import datetime

outF = open("log", "w")  # open new file; will replace existing file

for k in kvD:  # iterate through each key in dictionary
    v = kvD[k]  # get the value for the key; it's a string

    logTime = datetime.datetime.now()  # generate a date-time object
    # cast to str for printing
    s = str(logTime) + ": " + k + " " + v + "\n"

    outF.write(s)  # write entire line to file

outF.close()
Keywords for Exception Handling

try
extcept
finally
Summary – Elements of Python

Scalar variables, operators
Strings - Class with methods
List [ ], tuple ( ), dictionary { }
Control
Comments, indenting
def your own functions
import modules – use functions
Plotting
Text File I/O
Built-in Classes

str, list, tuple, dict, file

dir(str)
help(str)

hidden methods start with __
Built-in Functions

len()
range()
type()
input()            # read from standard input
                  # Python 2: raw_input()

print()
open()            # file I/O
help()            # interpreter

abs()    round()    complex()
min()     max()   sum()    pow()

dir()       dir(__builtins__)
e.g.      help(input)
Interpreter help()

>>> help()
# go into help mode
help>
help>
keywords
symbols
topics
modules
# enter topic UPPERCASE
q

>>>
Python at princeton.edu

ssh nobel.princeton.edu

% which python

/usr/bin/python
    version 2.7.5
/usr/bin/python3
    version 3.6.8

module load anaconda3/2019.10

python 3.7.4
More Info & Resources

python.org

docs.python.org

princeton.edu/~efeibush/python

“notes3” folder has exercises

Princeton University Python Community

princetonpy.com

PICSciE walk-in help sessions: Lewis 245
Tuesday 10:30 – 11:30 am   Thursday  2 – 3 pm
Resources

University library: O'Reilly books on-line

*Python in a Nutshell*

Where to?

Anaconda distribution of python

matplotlib – draw graphs
numpy – arrays & math functions
scipy – algorithms & math tools
PIL - Image Processing
Multiprocessing
Pycuda → GPU, CUDA
GUI – Tkinter, pyqt, wxpython
Visualization toolkit – python scripting
# Here is my python source code

```python
a = 3
b = 5
c = a * b
print(c)
```

---

```
15
```
Write a pgm (world’s simplest) image file:
   Replace my line for a gradient with your code to make an image.

   Change maxIntensity to your scale.

Display your picture:
   python pgmdisplay.py
Reading a netCDF File

Structured, scientific data file format
Can read from URL

scipy – netcdf_file class for read/write
numpy – multi-dimensional data arrays