Python reference

Laurel Farris

May 4, 2016

```
#!/usr/bin/env python

#export PYTHONPATH="${PYTHONPATH}:/new/path

# not compiled, execute with cl> python program_name.py
#   or cl> ./program_name.py IF you have the top line in your program
#     (must be the first line, literally)

# Resources
astropy.org
stackoverflow
rgex

# 'pickle' is the equivalent of IDL's 'save'

# To allow import:
def main():
    # program statements
    print "hello world."
if __name__=="__main__":  # flexible way of running routines
    main()


# Things to import (packages?)
import math [as shortername]
print math.sqrt(4)
circumference = 2*math.pi*radius
print math.exp(2) --> get e^2

import pdb
pdb.set_trace() #equivalent of 'STOP' in IDL

# line continuation:
implicit continuation using expression in parentheses...?

# variables: not declared

# lists... similar to arrays, but can't do mathematical operations
x = [0,1,2,3,4]
x = range(5) # Same as above... note that there are 5 elements, not
             # including the number 5
```

```
# numerical arrays
import numpy as np
np.zeros(n,m)
np.array(n) # creates an array that consists of n, NOT LENGTH n
np.array([[a,b,c,d][e,f,g]])
np.arange(n) # [0,1,2,...,n-1]
np.ndarray(...) # two-dimensional array
np.argmax(x) # returns INDEX of max value of x array
np.append(array,what_to_append)
np.linspace
np.logspace
np.sum(array)
np.roll(x,2) # (~IDL's SHIFT); shifts array elements 2 to the right
  --> [3,4,0,1,2] # note x is still the same, unless do x = np.roll(...)

# lists
array.append(newValue)
A = [1,2,3] # --> [1,2,3]
print A*2 # --> [1,2,3,1,2,3]... not [2,4,6] as expected
B = np.array([1,2,3]) # --> [1 2 3]
B*2  # --> [2 4 6]
<<<<<<< HEAD
=======
np.ndarray(...) # two-dimensional array
np.append...?
x = np.linspace(0,9,100) #0-9 with 100 increments
y1 = x**2
# just like with the math package, can't simply do sqrt(x). Need
# np.sqrt (or math.sqrt if not dealing with arrays).
y2 = np.sqrt(x)
# Also this doesn't work:
x = [1,2,3]
y = x**2
# because x is a list, not an array... even though it's just numbers.

>>>>>>> e65b08cb7892365af863d0202be5fe03115aa38c

num_elements = len(Array)

# dynamic memory allocation

# structures
(dictionaries, see also lists)
var={'name':'test', ra:1.}
print var['name'], var['test']
dicname = { 'A':[1,2,3], 'B':[2,4,6], 'C':[3,9,15] }
print dicname
# dictionaries are indexed by keys, rather than numbers.
# Keys can be any immutable type. Can't use lists, since they can be modified.
profile = {'density':rho_r, 'mass':mass, ...}
density = np.array(profile['density'])

# structure arrays
```

```
sarray = np.zeros(nelem,
 dtype=[('name','a12'),('ra','f4'),('dec','f4')])
sarray=np.zeros(nelem,
 dtype={'names': ('name', 'ra', 'dec'),
  'formats': ('S12','f4','f4')})

# operators (add, subtract, multiply, divide, exponent)
+,-,*,/,**,+=,-=,*=,/=,++,--
# bitwise operators (and, or, xor, not)
&,|,^,!
# string operators
+, str[i1:i2] (string slice, but string is immutable)

#matrix operations

# conditionals
if (condition):
    statements
else:
    statements
    (extent of conditional statements specified by indentation)
    pass # need to have 'something' here.
if (condition):
    break
else:
    continue
''' Conditions (logical operators) '''
==, !=, >, >=, <, <=, and, or, not

''' Looping '''
for i in (list): # i = value in list, NOT the index of each value
    statements
for i in range(1,x):  # ~ for i=1,x-1, i IS the index here

''' PYTHON IS NOT INCLUSIVE! '''


''' Looping with condition '''
while (condition):
    statements

''' Simple output '''
print 'characters', string, variable

'''
Formatted output
 > print 'the number is {:.#e|:nd|:n.nf|:ns}'.format(x)
    exponential, integer, float, string
 General syntax:
'''
template.format(var_1,var_2,...var_n)
# template:
#'{[field][!conversion]:[spec]}'
field = index of variables listed in .format()
```

```
conversion = int, float, string, etc.
spec = specifier
[[fill]align][sign][#][0][minwidth][.prec][type]
align:
<(left,default)
>(right)
=(padding after sign, before digits)
^(center)
0: zero padding (same as '=' and fill char of 0)
type: e,f,g(general)

# Including text:
print 'Variable 2 is {1} and variable 1 is {0}'.format(var_1,var_2)

'{:8d}asdfad{:8.2f}'.format(i,f)
formats: {:nd},{:n.nf},{:ns}
#    n-number (width.precision), d-integer, f-float, s-string
#-------------- How to use a variable for width?? ------------------#

# simple reading from terminal and file
terminal:
s=raw_input('prompt')
file:
f=open(file,'[rwa]') (read,write,append)
s=f.readline()
f.close(),
alternatively:
for line in f:
  print line

# higher level file reading routines
data = numpy.loadtxt(file,
 dtype=[('name','a12'),('ra',f4),('dec','f4')])

 data = astropy.io.ascii(file)

'''
-------------------Plotting-----------------------------
'''
import matplotlib.pyplot as plt

.csv file --> ?
pyplot.show()
pyplot.savefig('tmp.pdf')
  --> rewritten each time!
plt.xlabel('labelname',fontsize=14,color='red')

color='#eeefff'

# axes objects

fig = plt.figure()
for i in range(0,some_number):
    ax = fig.add_subplot(n,m,i+1)
```

```
    # n - vertical; m - horizontal; i+1 - plot being defined as 'ax'

fig.suptitle('bold figure suptitle',fontsize=14,fontweight='bold')
plt.xlabel('xlabel') # whole figure
plt.ylabel('xlabel') # whole figure
ax.tick_params(axis=['x'|'y'|'both'],labelsize='large')
ax.ticklabel_format(style='sci',
```