KEY_Practice09_Packages

July 12, 2019

1 Practice with packages!

Remember: * Functions and methods take an input, do something with the input, and return an output * Functions can take arguments that modify the output of the function * Methods are specific to ceratin object types * Packages are collections of functions

First, we have to import numpy:

[1]:
```
# command Python to import numpy
import numpy as np
```

Now, let's use a numpy function to get random numbers from a normal distribution (the bell curve):

[2]:
```
# use np.random.normal to get random numbers from a normal distribution
# the 0 is the mean of the distribution and the 1 is the standard deviation.␣
 ↪the 100 is the number of samples
numbers = np.random.normal(0, 1, 100)

# print numbers
print(numbers)
```

```
[-1.39824431 -1.76166251  0.83669333  1.33360894 -0.49462094  0.48796171
 -1.06458152  1.17580621  1.50728168  0.51071567 -0.53156599 -0.73132039
 -1.20048062  0.34644299 -0.15259685  1.3806484  -0.438835   -0.66199107
 -1.38793918 -0.5826494  -0.87847718  1.02316764 -1.57031353  0.08549053
  0.42805798 -1.38042177  0.76638705 -0.96425888  1.52762172  0.61726025
 -1.20799678  0.39308299  2.30805412  0.58539944 -3.06280901 -0.32682174
  1.21076048  0.21104442 -1.09344276 -0.44282578  1.05969761 -0.58336075
  1.12012205  1.19334622  0.04389964  1.77846379 -2.21769015 -0.57741489
 -0.88719918 -0.87502049 -0.29861258 -0.24046768 -0.10085904  0.58088138
  0.76729548 -1.37138301  0.06116189  0.55618946 -0.66409905 -0.17966806
 -0.536019   -2.24568329 -1.24587218 -0.02894768 -0.03916526 -1.29917822
 -1.09798789 -0.20351238  0.06703675  0.52252771  0.62698796 -0.22718316
 -0.13256246  0.13017293  0.47935942  0.94389704  0.04757612  0.09725786
  0.6342669   1.34136594 -0.61623609  0.46616108  1.67502907  3.49134311
 -0.98592029 -1.6212825   0.24048973 -0.55281765  2.97495292  1.71777094
  0.72910466 -2.34129259  0.62385808  1.08726304 -0.32394229 -1.2396672
 -0.03296779  0.16765308  0.12932121  0.77008837]
```

How many elements are in `numbers`?

[3]:
```
# get the number of elements in numbers
print(len(numbers))
```

```
100
```

What is the mean of `numbers`?

[4]:
```
# command Python to get the mean of numbers
np.mean(numbers)
```

[4]:
```
-0.012398409645250607
```

Get the absolute value of `numbers` and save it to `abs_num`:

[5]:
```
# command Python to get the absolute value of numbers
abs_num = np.abs(numbers)
# print abs_num
print(abs_num)
```

```
[ 1.39824431  1.76166251  0.83669333  1.33360894  0.49462094  0.48796171
  1.06458152  1.17580621  1.50728168  0.51071567  0.53156599  0.73132039
  1.20048062  0.34644299  0.15259685  1.3806484   0.438835    0.66199107
  1.38793918  0.5826494   0.87847718  1.02316764  1.57031353  0.08549053
  0.42805798  1.38042177  0.76638705  0.96425888  1.52762172  0.61726025
  1.20799678  0.39308299  2.30805412  0.58539944  3.06280901  0.32682174
  1.21076048  0.21104442  1.09344276  0.44282578  1.05969761  0.58336075
  1.12012205  1.19334622  0.04389964  1.77846379  2.21769015  0.57741489
  0.88719918  0.87502049  0.29861258  0.24046768  0.10085904  0.58088138
  0.76729548  1.37138301  0.06116189  0.55618946  0.66409905  0.17966806
  0.536019    2.24568329  1.24587218  0.02894768  0.03916526  1.29917822
  1.09798789  0.20351238  0.06703675  0.52252771  0.62698796  0.22718316
  0.13256246  0.13017293  0.47935942  0.94389704  0.04757612  0.09725786
  0.6342669   1.34136594  0.61623609  0.46616108  1.67502907  3.49134311
  0.98592029  1.6212825   0.24048973  0.55281765  2.97495292  1.71777094
  0.72910466  2.34129259  0.62385808  1.08726304  0.32394229  1.2396672
  0.03296779  0.16765308  0.12932121  0.77008837]
```

What is the mean of `abs_num`?

[6]:
```
# command Python to print the mean of abs_num
print(np.mean(abs_num))
```

```
0.869598950082
```

Now, let's sort `numbers`:

[7]:
```
# sort numbers
numbers.sort()
# print numbers
print(numbers)
```

```
[-3.06280901 -2.34129259 -2.24568329 -2.21769015 -1.76166251 -1.6212825
 -1.57031353 -1.39824431 -1.38793918 -1.38042177 -1.37138301 -1.29917822
 -1.24587218 -1.2396672  -1.20799678 -1.20048062 -1.09798789 -1.09344276
 -1.06458152 -0.98592029 -0.96425888 -0.88719918 -0.87847718 -0.87502049
 -0.73132039 -0.66409905 -0.66199107 -0.61623609 -0.58336075 -0.5826494
 -0.57741489 -0.55281765 -0.536019   -0.53156599 -0.49462094 -0.44282578
 -0.438835   -0.32682174 -0.32394229 -0.29861258 -0.24046768 -0.22718316
 -0.20351238 -0.17966806 -0.15259685 -0.13256246 -0.10085904 -0.03916526
 -0.03296779 -0.02894768  0.04389964  0.04757612  0.06116189  0.06703675
  0.08549053  0.09725786  0.12932121  0.13017293  0.16765308  0.21104442
  0.24048973  0.34644299  0.39308299  0.42805798  0.46616108  0.47935942
  0.48796171  0.51071567  0.52252771  0.55618946  0.58088138  0.58539944
  0.61726025  0.62385808  0.62698796  0.6342669   0.72910466  0.76638705
  0.76729548  0.77008837  0.83669333  0.94389704  1.02316764  1.05969761
  1.08726304  1.12012205  1.17580621  1.19334622  1.21076048  1.33360894
  1.34136594  1.3806484   1.50728168  1.52762172  1.67502907  1.71777094
  1.77846379  2.30805412  2.97495292  3.49134311]
```

Now let's round `numbers` so each element has 2 decimal places:

[8]:
```
# command Python to round all elements in numbers to 2 decimal places
np.round(numbers,2)
```

[8]:
```
array([-3.06, -2.34, -2.25, -2.22, -1.76, -1.62, -1.57, -1.4 , -1.39,
       -1.38, -1.37, -1.3 , -1.25, -1.24, -1.21, -1.2 , -1.1 , -1.09,
       -1.06, -0.99, -0.96, -0.89, -0.88, -0.88, -0.73, -0.66, -0.66,
       -0.62, -0.58, -0.58, -0.58, -0.55, -0.54, -0.53, -0.49, -0.44,
       -0.44, -0.33, -0.32, -0.3 , -0.24, -0.23, -0.2 , -0.18, -0.15,
       -0.13, -0.1 , -0.04, -0.03, -0.03,  0.04,  0.05,  0.06,  0.07,
        0.09,  0.1 ,  0.13,  0.13,  0.17,  0.21,  0.24,  0.35,  0.39,
        0.43,  0.47,  0.48,  0.49,  0.51,  0.52,  0.56,  0.58,  0.59,
        0.62,  0.62,  0.63,  0.63,  0.73,  0.77,  0.77,  0.77,  0.84,
        0.94,  1.02,  1.06,  1.09,  1.12,  1.18,  1.19,  1.21,  1.33,
        1.34,  1.38,  1.51,  1.53,  1.68,  1.72,  1.78,  2.31,  2.97,  3.49])
```

Challenge: Get the square root of the `abs_num` function. Save this to sqrt_abs_num. *Hint:* Use the function `sqrt` in the numpy package.

[9]:
```
# command Python to get the square root of the numbers in abs_num
sqrt_abs_num = np.sqrt(abs_num)
# print sqrt_abs_num
print(sqrt_abs_num)
```

```
[ 1.18247381  1.32727635  0.91470942  1.15481988  0.70329292  0.69854256
  1.0317856   1.08434598  1.227714    0.71464374  0.72908572  0.85517272
  1.09566446  0.58859408  0.39063647  1.17500996  0.66244622  0.81362834
  1.17810831  0.76331474  0.93727113  1.01151749  1.25312151  0.29238764
  0.6542614   1.17491351  0.87543535  0.98196684  1.23596995  0.78565912
  1.09908907  0.62696331  1.51922813  0.765114    1.75008829  0.57168325
  1.10034562  0.45939571  1.04567814  0.66545156  1.02941615  0.76378056
```

```
 1.05835819  1.09240387  0.20952242  1.33359056  1.4891911   0.75987821
 0.94191251  0.9354253   0.54645456  0.49037504  0.31758313  0.76215575
 0.87595404  1.17106064  0.24730931  0.74578111  0.81492272  0.42387269
 0.73213318  1.49856041  1.11618644  0.17014016  0.19790214  1.13981499
 1.04784917  0.45112347  0.25891455  0.72286078  0.79182571  0.47663735
 0.36409127  0.36079486  0.69235787  0.97154364  0.21811952  0.31186192
 0.79640875  1.15817354  0.78500707  0.6827599   1.29422914  1.86851361
 0.99293519  1.27329592  0.49039752  0.74351708  1.72480518  1.31063761
 0.85387625  1.53012829  0.78984687  1.04271906  0.56915928  1.11340343
 0.18157034  0.40945461  0.35961259  0.87754679]
```

What is the max of the square roots? The min?

[10]:
```
# print the max of the square roots
print(max(sqrt_abs_num))
# print the min of the square roots
print(min(sqrt_abs_num))
```

```
1.86851360945
0.170140164357
```

Nice job! You just practiced: * Importing packages in Pyton * Using functions in packages * Saving things to variables * Doing math with numpy