24. Giới thiệu np.arange() và np.linspace()

np.arange()

Mục đích là tạo ra các giá trị có khoảng cách đều trong một khoảng nhất định trong Numpy. Các giá trị được sinh ra nằm trong khoảng [start,stop) (sẽ bao hàm giá trị start nhưng không chứa stop). Với các đối số truyền vào là số nguyên, hàm arange sẽ tương đương với hàm range trong "Python core" nhưng giá trị trả về là ndarray thay vì một list.

Sử dụng **np.arange([start,] stop[, step,], dtype=None)**

Với cú pháp trên, ta có thể gán tường minh 3 thông số đầu vào, start, stop và step. Nếu start không được gán nó sẽ tự động nhận giá trị "0", tương tự như vậy cho step sẽ được gán là 1.0 nếu nó không được gán tường minh bởi người sử dụng và "step" sẽ nhận số thực điều này khác với python core với kiểu list trong hàm "range". Thêm nữa, nếu "step" được gán tường minh thì "start" cũng phải được gán tường minh. Tham số dtype có thể gán tường minh hoặc nhận giá trị None để hệ thống tự nhận giá trị ứng với đầu vào.

Giá trị trả về:

"arange" sẽ trả về một "ndarray" thay vì một "list" như trong "range" thuộc python core. Với các đối số truyền vào là dấu phẩy động (floating point), độ dài của kết quả trả về là ceil((stop-start)/step). (ceil() trả về số integer nhỏ nhất mà lớn hơn hoặc bằng tham số). Việc này có thể dẫn đến một số phần tử cuối của mảng kết quả sẽ lớn hơn giá trị stop.

Các bạn có thể theo dõi ví dụ bên dưới để hiểu rõ hơn.

```
import numpy as np
a = np.arange(1, 10)
print "np.arange(1, 10) = ", a
x = range(1,10)
print "range(1,10) = ",x

x = np.arange(10.4)
print "np.arange(10.4) = ",x
x = np.arange(0.5, 10.4, 0.8)
print "np.arange(0.5, 10.4, 0.8) = ",x
x = np.arange(0.5, 10.4, 0.8, int)
print "np.arange(0.5, 10.4, 0.8, int) = ", np.arange(0.5, 10.4, 0.8, int)

Output:

np.arange(1, 10) =  [1 2 3 4 5 6 7 8 9]

range(1,10) =  [1, 2, 3, 4, 5, 6, 7, 8, 9]

np.arange(10.4) =  [  0.   1.   2.   3.   4.   5.   6.   7.   8.   9.  10.]

np.arange(0.5, 10.4, 0.8) =  [  0.5   1.3   2.1   2.9   3.7   4.5   5.3   6.1   6.9   7.7   8.5   9.3   10.1]

np.arange(0.5, 10.4, 0.8, int) =  [ 0  1  2  3  4  5  6  7  8  9 10 11 12]
```

np.linspace()

linspace cũng tương tự như arange nhưng nó truyền vào số lượng mẫu (num) thay vì kích thước của step như arange.

Sử dụng **np.linspace(start, stop, num=50, endpoint=True, retstep=False)**

Linspace trả về một ndarray, bao gồm các mẫu phân bố bằng nhau (num) trong khoảng đóng [start, stop] hoặc [start, stop). Nếu một khoảng đóng kín [] hoặc [) sẽ được trả về, phụ thuộc vào việc endpoint' là True hay False. Tham số 'start' xác định giá trị bắt đầu của dãy sẽ được tạo ra. 'Stop' sẽ là giá trị kết thúc của dãy, trừ khi 'endpoint' được đặt thành False. Trong trường hợp thứ hai, trình tự kết quả sẽ bao gồm tất cả trừ các mẫu được chia đều cho 'num + 1'. Điều này có nghĩa là 'stop' được loại trừ. Lưu ý rằng kích thước bước thay đổi khi 'stop' là False. Số lượng mẫu được tạo ra có thể được đặt bằng 'num', mặc định là 50. Nếu tham số tùy chọn 'endpoint' được đặt thành True (mặc định), 'stop' sẽ là mẫu cuối cùng của chuỗi. Nếu không, nó không được bao gồm.

```
import numpy as np
print "50 values between 1 and 10:"
print "np.linspace(1, 10) = ", np.linspace(1, 10)

print "7 values between 1 and 10:"
print "np.linspace(1, 10, 7)", np.linspace(1, 10, 7)

print "excluding the endpoint:"
print "np.linspace(1, 10, 7, endpoint=False) = ", np.linspace(1, 10, 7, endpoint=False)
print "comparing to"
print "np.linspace(1, 10, 8) = ", np.linspace(1, 10, 8)



Output:

50 values between 1 and 10:

np.linspace(1, 10) =  [  1.          1.18367347   1.36734694   1.55102041   1.73469388

   1.91836735   2.10204082   2.28571429   2.46938776   2.65306122

   2.83673469   3.02040816   3.20408163   3.3877551    3.57142857

   3.75510204   3.93877551   4.12244898   4.30612245   4.48979592

   4.67346939   4.85714286   5.04081633   5.2244898    5.40816327

   5.59183673   5.7755102    5.95918367   6.14285714   6.32653061

   6.51020408   6.69387755   6.87755102   7.06122449   7.24489796

   7.42857143   7.6122449    7.79591837   7.97959184   8.16326531

   8.34693878   8.53061224   8.71428571   8.89795918   9.08163265

   9.26530612   9.44897959   9.63265306   9.81632653  10.        ]

7 values between 1 and 10:

np.linspace(1, 10, 7) [  1.    2.5   4.    5.5   7.    8.5  10. ]

excluding the endpoint:

np.linspace(1, 10, 7, endpoint=False) =  [ 1.          2.28571429  3.57142857  4.85714286  6.14285714  7.42857143

  8.71428571]

comparing to

np.linspace(1, 10, 8) =  [  1.          2.28571429   3.57142857   4.85714286   6.14285714

   7.42857143   8.71428571  10.        ]
```

Một tham số thú vị đó là “retstep”. Nếu tham số tùy chọn 'retstep' được thiết lập, hàm cũng sẽ trả lại giá trị khoảng cách giữa các giá trị liền kề. Vì vậy, hàm sẽ trả về một tuple (‘ndarray', 'step'):

Ví dụ sau:

```
import numpy as np
samples, spacing = np.linspace(1, 10, retstep=True)
print "spacing = ",spacing
samples, spacing = np.linspace(1, 10, 20, endpoint=True, retstep=True)
print "spacing = ",spacing
samples, spacing = np.linspace(1, 10, 20, endpoint=False, retstep=True)
print "spacing = ",spacing



Output:



spacing =  0.183673469388

spacing =  0.473684210526

spacing =  0.45
```

Kết luận

Hai hàm **np.arange()** và **np.linspace()** đều là những hàm cơ bản trong thư viện numpy hỗ trợ tạo ra các giá trị có khoảng cách đều nhau trong một khoảng nhất định cho trước. **np.linspace()** truyền vào số lượng mẫu (num) thay vì kích thước của step như **np.arange()**. Khi đối số step là non-integer (ví dụ 0.1) kết quả trả về của arrange thường không chính xác. Với những trường hợp như vậy, tốt hơn là nên sử dụng linspace. Trong bài tiếp theo chúng tôi sẽ giới thiệu về mảng trong numpy.