Python Data Science Handbook by Jake VanderPlas

Fancy Indexing

Array များကို indexing လုပ်သည့် နောက်စတိုင် တမျိုးမှာ *fancy indexing* ဖြစ်သည်။

Fancy indexing ဆိုသည်မှာ ပြီးခဲ့သည့် အခန်းတွင် ဖော်ပြပြီး ဖြစ်သည့် ရိုးရိုး indexing များကဲ့သို့ပင်ဖြစ်သည်။ သို့သော် ရိုးရိုး indexing တွင် ကိန်းတစ်ခု သို့မဟုတ် ဂဏန်းတစ်ခု(single scalar)ကိုသာ ထည့်ပြီး indexing လုပ်သည်။ Fancy indexing လုပ်သည့်အခါ တစ်ခုထက် ပိုများသည့် ကိန်းများ array ပုံစံဖြင့်(arrays of indices) indexing လုပ်သည်။

ထို့သို့ ပြုလုပ်နိုင်ခြင်းကြောင့် လျှင်မြန်စွာ access လုပ်နိုင်သည်။ array အကြီးကြီးထဲမှ array ငယ်များကို လျှင်မြန်လွယ်ကူစွာ ခွဲထုတ်နိုင်သည်။ လိုသလို ပြောင်းလဲ နိုင်သည်။ (very quickly access and modify complicated subsets of an array's values.)

Exploring Fancy Indexing

Fancy indexing လုပ်ရန် တစ်ခုထက်ပိုများသည့် ကိန်းများကို access လုပ်ရန် index ကိန်းများကို array ပုံစံဖြင့် ထည့်ပေးခြင်းဖြစ်သည်။ (passing an array of indices to access multiple array elements at once.)

ဥပမာဖြင့် ရှင်းပြရန် x ဟုသည့် array ကို random ကိန်းများ ဖြင့် တည်ဆောက်ပါသည်။ `RandomState` method ကို random နံပါတ်များ ထုတ်ပေးရန် အသုံးပြုသည်။ (methods for generating random numbers drawn from a variety of probability distributions)

`rand.randint(100, size=10)` သည် ၁၀၀ ထက်နည်းသည့် random နံပါတ် (၁၀)ခု ထုတ်ပေးရန် ဖြစ်သည်။

In [1]:
```
import numpy as np
rand = np.random.RandomState(42)

x = rand.randint(100, size=10)
print(x)
```

```
[51 92 14 71 60 20 82 86 74 74]
```

ဥပမာအဖြစ် အောက်တွင် စတုတ္ထ နေရာမှ ကိန်း ၊ အဋ္ဌမ နေရာမှကိန်းနှင့် တတိယနေရာမှ ကိန်းတို့ကို access လုပ်ပုံကို နည်းသုံးမျိုးဖြင့် ဖော်ပြထားသည်။

x[3] = စတုတ္ထနေရာမှ ကိန်း , x[7] = အဋ္ဌမနေရာမှကိန်း၊ x[2] = တတိယနေရာမှ ကိန်း

In [2]:
```
[x[3], x[7], x[2]]
```

Out[2]:
```
[71, 86, 14]
```

ကိန်းတစ်ခုချင်းစီ မထည့်ပေးဘဲ (pass မလုပ်ဘဲ) ကိန်းသုံးခုစလုံးတွက် index များကို list တစ်ခု သို့မဟုတ် array တစ်ခု ပုံစံမျိုးဖြင့် ထည့်ပေးပြီး access လုပ်နိုင်သည်။ (pass a single list or array of indices to obtain the same result)

`ind = [3, 7, 4]` သည် access လုပ်လို့သည့် နေရာ(index)များကို ပေါင်း၍ list တစ်ခု ပြုလုပ်သည်။ ထို ind ကို သုံး၍ x array ကို access လုပ်သည်။

In [3]:
```
ind = [3, 7, 4]
x[ind]
```

Out[3]:
```
array([71, 86, 60])
```

Fancy indexing ပြုလုပ်သည့်အခါ shape of the result သည် shape of the *index arrays* ဖြစ်သည်။ shape of the *array being indexed*: နှင့် သက်ဆိုင်မှု မရှိပေ။ (the shape of the result reflects the shape of the *index arrays* rather than the shape of the *array being indexed*)

တစ်နည်းအားဖြင့် ထွက်လာသည့် array ၏ shape သည် index array အတိုင်း ဖြစ်သည်။

In [4]:
```
ind = np.array([[3, 7],
                [4, 5]])
x[ind]
```

Out[4]:
```
array([[71, 86],
       [60, 20]])
```

multiple dimension အတွက်လည်း Fancy indexing ကို အသုံးပြုနိုင်သည်။ (Fancy indexing also works in multiple dimensions.)

In [5]:
```
X = np.arange(12)
X
```

Out[5]:
```
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11])
```

`np.arange(12)` သည် ၀ မှ ၁၁ အထိ ကိန်း ၁၂ လုံးထုတ်ပေးရန် ဖြစ်သည်။ ထို့အစဉ်လိုက် ကိန်း (၁၂)လုံးကို `.reshape((3, 4))` ဖြင့် 3 x 4 array ဖြစ်အောင် ပြောင်းသည်။

In [6]:
```
X = np.arange(12).reshape((3, 4))
X
```

Out[6]:
```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11]])
```

အထက်ပါ X array ကို standard indexing ဖြစ်သည့် row, col ဖြင့် access လုပ်သည်။ `X[row, col]` row index များကို `row = np.array([0, 1, 2])` ဖြင့် row array တစ်ခု ဖြစ်အောင် ပြုလုပ်သည်။ ကော်လံ index များကို `col = np.array([2, 1, 3])` ဖြင့် colarray တစ်ခု ဖြစ်အောင် ပြုလုပ်သည်။

Like with standard indexing, the first index refers to the row, and the second to the column:

```
In [7]: row = np.array([0, 1, 2])
        col = np.array([2, 1, 3])
        X[row, col]
```

```
Out[7]: array([ 2,  5, 11])
```

2 သည် `X[0, 2]` ဖြစ်ပြီး , 5 သည် `X[1, 1]` နှင့် , 11 onf `X[2, 3]` ဖြစ်သည်။

fancy indexing တွင် index များကို တွဲခြင်း(pairing) လုပ်သည့်အခါ broadcasting rules ကို လိုက်နာသည်။ ဥပမာ index များ၏ column vector တစ်ခုနှင့် row vector တစ်ခုကို ပေါင်းသည့်အခါ two-dimensional index array တစ်ခုရသည်။

```
In [8]: X[row[:, np.newaxis], col]
```

```
Out[8]: array([[ 2,  1,  3],
               [ 6,  5,  7],
               [10,  9, 11]])
```

row value တစ်ခုချင်းစီဖြင့် column vector တစ်ခုချင်းစီကို match လုပ်ခြင်းသည် broadcasting of arithmetic operations ဖြစ်သည်။ Here, each row value is matched with each column vector, exactly as we saw in broadcasting of arithmetic operations. For example:

```
In [9]: row[:, np.newaxis] * col
```

```
Out[9]: array([[0, 0, 0],
               [2, 1, 3],
               [4, 2, 6]])
```

fancy indexing တွင် return value သည် *broadcasted shape of the indices* အတိုင်းဖြစ်သည်။ shape of the array being indexed အတိုင်း ဖြစ်လိမ့်မည် မဟုတ်ပေ။

It is always important to remember with fancy indexing that the return value reflects the *broadcasted shape of the indices*, rather than the shape of the array being indexed.

Combined Indexing

fancy indexing ကို တခြားသော indexing scheme များဖြင့် တွဲသုံးသည့် ပိုအစွမ်းထက်သည်။ (more powerful operations)

In [10]:
```
print(X)
```

```
[[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]]
```

fancy indexing နှင့် simple indexing တွဲသုံးပုံကို အောက်တွင် ဥပမာဖြင့် ပြထားသည်။ `[2, [2, 0, 1]]` သည် တတိယ row နှင့် col index array ဖြစ်သည်။ 2,2 2,0 နှင့် 2,1 တို့ဖြစ်သည်။

In [11]:
```
X[2, [2, 0, 1]]
```

Out[11]:
```
array([10,  8,  9])
```

Fancy indexing ကို slicing လုပ်ခြင်းနှင့် လည်း တွဲ၍ အသုံးပြုနိုင်သည်။

In [12]:
```
X[1:, [2, 0, 1]]
```

Out[12]:
```
array([[ 6,  4,  5],
       [10,  8,  9]])
```

Fancy indexing ကို masking နှင့် လည်း တွဲ၍ အသုံးပြုနိုင်သည်။

In [13]:
```
mask = np.array([1, 0, 1, 0], dtype=bool)
X[row[:, np.newaxis], mask]
```

Out[13]:
```
array([[ 0,  2],
       [ 4,  6],
       [ 8, 10]])
```

Array တစ်ခုအတွင်းရှိ value များအား access လုပ်သည့်အခါ နှင့် modify လုပ်သည်အခါများတွင် fancy indexing ကို တခြားသော နည်းများဖြင့် တွဲ၍ အသုံးပြုခြင်းကြောင့် ပို၍ လွယ်ကူ လျှင်မြန်သည်။