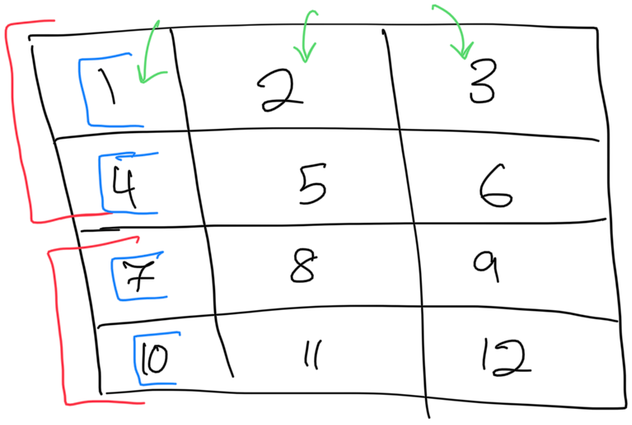
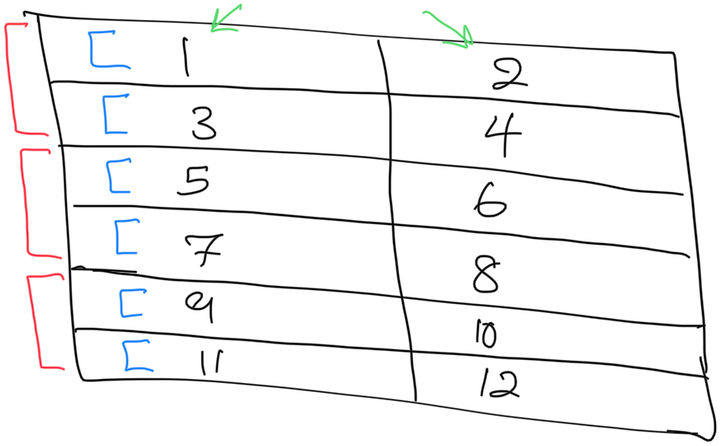
4- N-Dimensional Array

شرح مفصل عن المصفوفات متعددة الأبعاد.

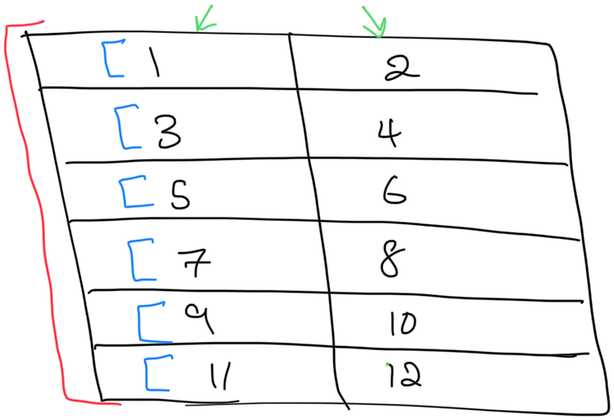
$(2, 2, 3) = 12$



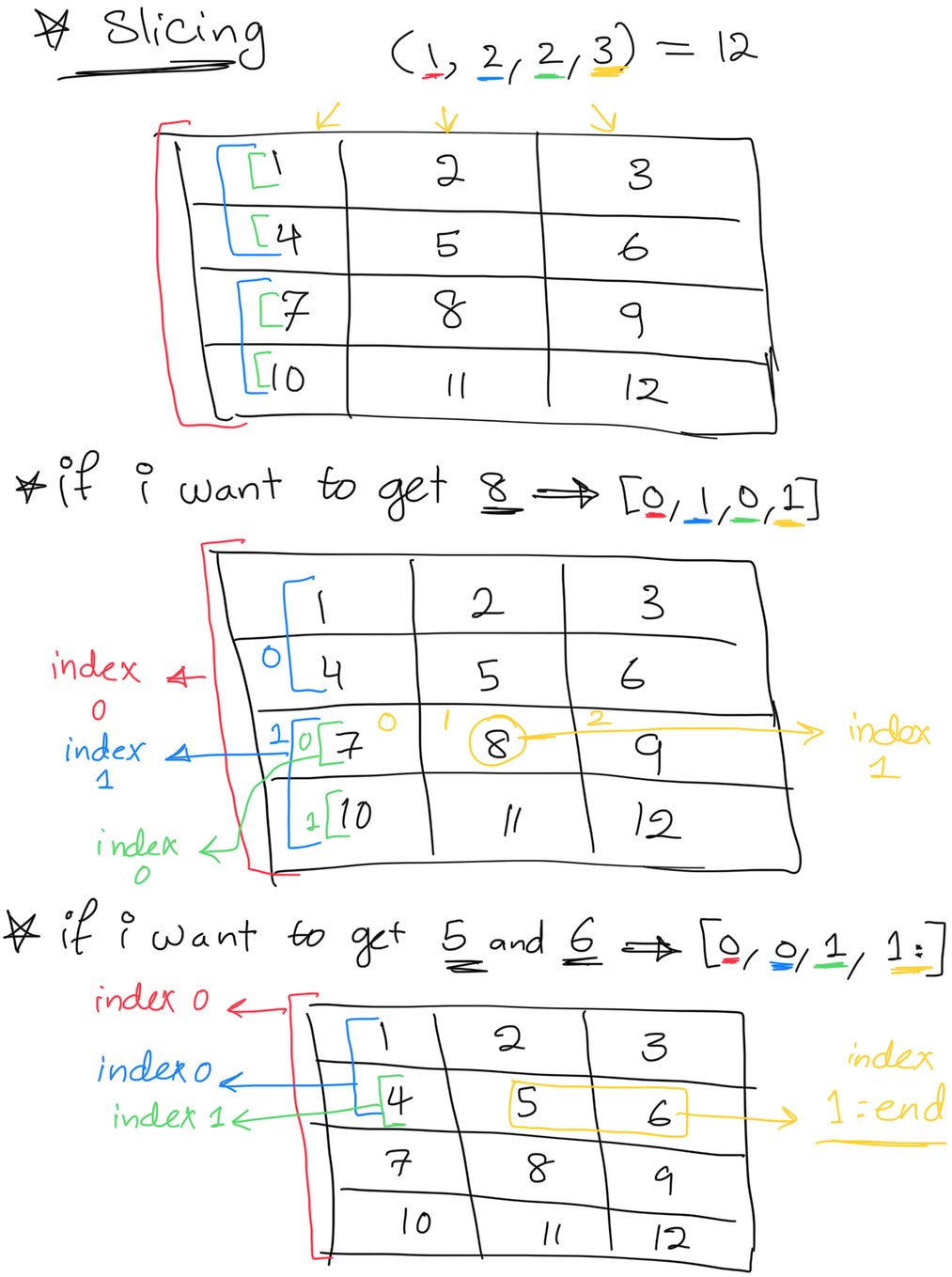
$(3, 2, 2) = 12$



$(1, 6, 2) = 1$



شرح مفصل عن الوصول للعناصر داخل المصفوفات متعددة الأبعاد.



اولاً المصفوفة ثلاثية الابعاد :

- هي مصفوفة تتكون من ثلاث أبعاد

```
arr = np.arange(90)

print("array shape is : ",arr.shape)
print("array dimensions is :",arr.ndim)
```

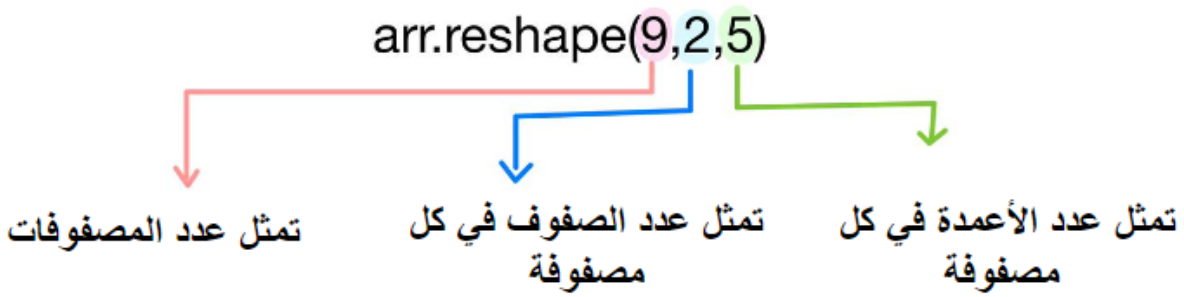
```
array shape is :  (90,)
array dimensions is : 1
```

- تم تعريف مصفوفة ثنائية البعد بإستخدام arange()
- طباعة shape , dimension

```
d3_arr = arr.reshape(9,2,5)

# 9 = arrays
# 2 = rows
# 5 = columns
```

- تم تعريف متغير جديد يحمل إسم d3_arr من المصفوفة القديمة arr
- لكن بإستخدام reshape(9,2,3) ← اي انه تم تحويل arr من مصفوفة ثنائية البعد الى ثالثية البعد و تخزينها في المتغير الجديد d3_arr



- يكون ناتج reshape :

```
array([[[ 0,  1,  2,  3,  4],
        [ 5,  6,  7,  8,  9]],

       [[10, 11, 12, 13, 14],
        [15, 16, 17, 18, 19]],

       [[20, 21, 22, 23, 24],
        [25, 26, 27, 28, 29]],

       [[30, 31, 32, 33, 34],
        [35, 36, 37, 38, 39]],

       [[40, 41, 42, 43, 44],
        [45, 46, 47, 48, 49]],

       [[50, 51, 52, 53, 54],
        [55, 56, 57, 58, 59]],

       [[60, 61, 62, 63, 64],
        [65, 66, 67, 68, 69]],

       [[70, 71, 72, 73, 74],
        [75, 76, 77, 78, 79]],

       [[80, 81, 82, 83, 84],
        [85, 86, 87, 88, 89]]])
```

- نتج عن reshape تسع مصفوفات وكل مصفوفة مكونة من (n=2,m=5) اي صفين و خمس أعمدة.

الوصول الى قيمة معينة في **3D array:**

لنفترض أننا نريد طباعة المصفوفة الخامسة من المصفوفة d3_arr

```
d3_arr[4]
```

```
array([[40, 41, 42, 43, 44],
       [45, 46, 47, 48, 49]])
```

- هنا index يبدأ من صفر , فسوف تكون المصفوفة الخامسة في index 4 .

```
d3_arr[4][0]
```

```
array([40, 41, 42, 43, 44])
```

```
d3_arr[4][1]
```

```
array([45, 46, 47, 48, 49])
```

- [0] هنا يشير إلى row vector فسيتم إرجاع الصف الأول كامل
- [1] هنا يشير إلى row vector فسيتم إرجاع الصف الثاني كامل

الأن لنفترض اننا نريد الوصول الى 41 في الصف الاول العمود الثاني

```
d3_arr[4][0][1]
```

```
41
```

الأن لنفترض اننا نريد الوصول الى 48 في الصف الثاني العمود الرابع

```
d3_arr[4][1][3]
```

```
48
```

لنفترض الأن اننا نريد طباعة العددين المتتابعين 6 و 7 ,[6,7,16,17,26,27,...]

```
array([[[ 0,  1,  2,  3,  4],
        [ 5,  6,  7,  8,  9]],

       [[10, 11, 12, 13, 14],
        [15, 16, 17, 18, 19]],

       [[20, 21, 22, 23, 24],
        [25, 26, 27, 28, 29]],

       [[30, 31, 32, 33, 34],
        [35, 36, 37, 38, 39]],

       [[40, 41, 42, 43, 44],
        [45, 46, 47, 48, 49]],

       [[50, 51, 52, 53, 54],
        [55, 56, 57, 58, 59]],

       [[60, 61, 62, 63, 64],
        [65, 66, 67, 68, 69]],

       [[70, 71, 72, 73, 74],
        [75, 76, 77, 78, 79]],

       [[80, 81, 82, 83, 84],
        [85, 86, 87, 88, 89]]])
```

- الارقام متواجدة في جميع المصفوفات
- جميعها في الصف الثاني لكل مصفوفة
- جميعها في العمود الثاني و الثالث

```
d3_arr[:: ,1:  ,1:3]
```

```
array([[[ 6,  7]],

       [[16, 17]],

       [[26, 27]],

       [[36, 37]],

       [[46, 47]],

       [[56, 57]],

       [[66, 67]],

       [[76, 77]],

       [[86, 87]]])
```

نلاحظ : "::" تعني جميع الطبقات , " 1:" تعني جميع الصف الثاني , "1:3" اي من العمود الثاني الى الثالث

مثال اخر لنفترض اننا نريد طباعة الارقام [55, 56, 57, 58] و [65, 66, 67, 68]

```
d3_arr[5:7 ,1: ,:4]
```

```
array([[[55, 56, 57, 58]],

       [[65, 66, 67, 68]]])
```

نريد الأن طباعة الرقم 55 و 66 فقط !!!

```
d3_arr[5:7 ,1 ,0:2]
d3_arr[5:7 ,1 ,1:2]
```

```
array([[56],
       [66]])
```

طرق إضافة عمود/صف جديد للمصفوفة لـ 3D:

```
d3_add = np.array([[[1,2,3],
                    [4,5,6],
                    [7,8,9]]])
```

```
print("Array's type:" , d3_add.dtype)
print("Array's size:" , d3_add.size)
print("Array's shape:" , d3_add.shape)
print("Array's dimention:" , d3_add.ndim)
```

```
Array's type: int32
Array's size: 9
Array's shape: (1, 3, 3)
Array's dimention: 3
```

1. بإستخدام append():

```
new_row = np.array([[[10,11,12],
                    [13,14,15]]])

new1 = np.append(d3_add, new_row, axis=1)
new1
```

```
array([[[ 1,  2,  3],
        [ 4,  5,  6],
        [ 7,  8,  9],
        [10, 11, 12],
        [13, 14, 15]]])
```

في (المحور "0: للأعمدة 1:للصفوف",المصفوفة المراد جمعها,المصفوفة الاصلية)append

```
print("Array's shape:" , new1.shape)
```

```
Array's shape: (1, 5, 3)
```

ايضاً بإستخدام append نستطيع إضافة layer او بعد جديد:

```
new_col = np.array([[[16,17,18],
                     [19,20,21],
                     [22,23,24]]])
new = np.append(d3_add, new_col, axis=0)
new
```

```
array([[[ 1,  2,  3],
        [ 4,  5,  6],
        [ 7,  8,  9]],

       [[16, 17, 18],
        [19, 20, 21],
        [22, 23, 24]]])
```

```
print("Array's shape:" , new.shape)
```

```
Array's shape: (2, 3, 3)
```

2. بإستخدام ()concatenate:

```
new_row = np.array([[[10,11,12],
                     [13,14,15]]])

new1 = np.concatenate([d3_add, new_row], axis=1)
new1
```

```
array([[[ 1,  2,  3],
        [ 4,  5,  6],
        [ 7,  8,  9],
        [10, 11, 12],
        [13, 14, 15]]])
```

```
print("Array's shape:" , new1.shape)
```

```
Array's shape: (1, 5, 3)
```

إضافة rows للمصفوفة

```
new_col = np.array([[[16,17,18],
                     [19,20,21],
                     [22,23,24]]])
new = np.concatenate([d3_add, new_col], axis=0)
new
```

```
array([[[ 1,  2,  3],
        [ 4,  5,  6],
        [ 7,  8,  9]],

       [[16, 17, 18],
        [19, 20, 21],
        [22, 23, 24]]])
```

```
print("Array's shape:" , new.shape)
```

```
Array's shape: (2, 3, 3)
```

هنا ايضاً لأضافة بُعد أو layer جديد للمصفوفة