Mid Journey V4 prompt: a lot of dictionaries --niji --ar 3:2 --q 2

LECTURE 3-2: PYTHON BASICS II - DICTIONARY AND LOOP

Dr. James Chen, Animal Data Scientist, School of Animal Sciences

2023 APSC-5984 SS: Agriculture Data Science

Multi-Axis Operation

6

3

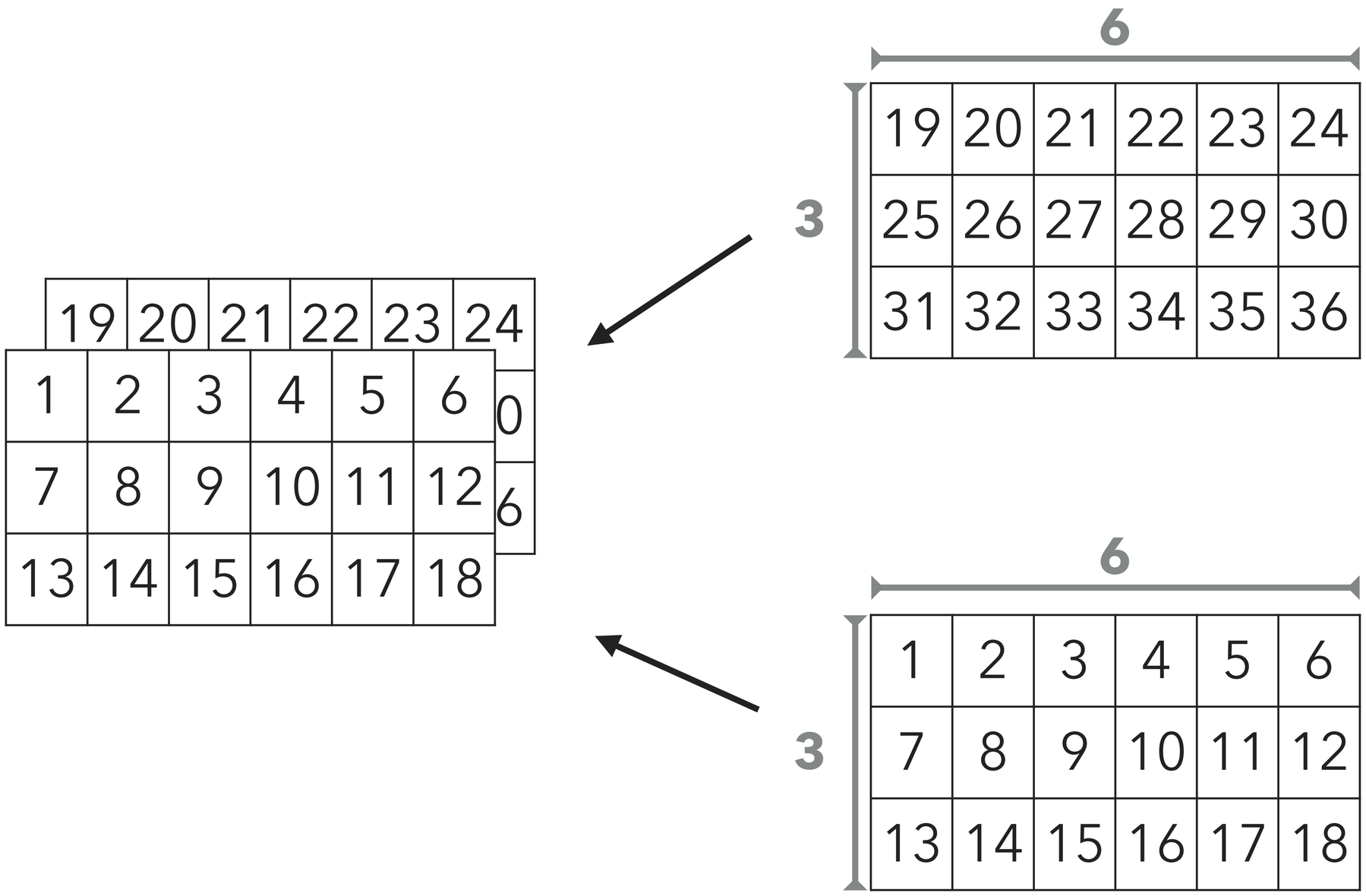
19	20	21	22	23	24
25	26	27	28	29	30
31	32	33	34	35	36

6

3

1	2	3	4	5	6
7	8	9	10	11	12
13	14	15	16	17	18

Multi-Axis Operation



Multi-Axis Operation

3D matrix: 2 x 3 x 6

Dim. 2

Dim. 0

6

2

19	20	21	22	23	24

Dim. 1 **3**

1	2	3	4	5	6	0
7	8	9	10	11	12	6
13	14	15	16	17	18	

6

3

19	20	21	22	23	24
25	26	27	28	29	30
31	32	33	34	35	36

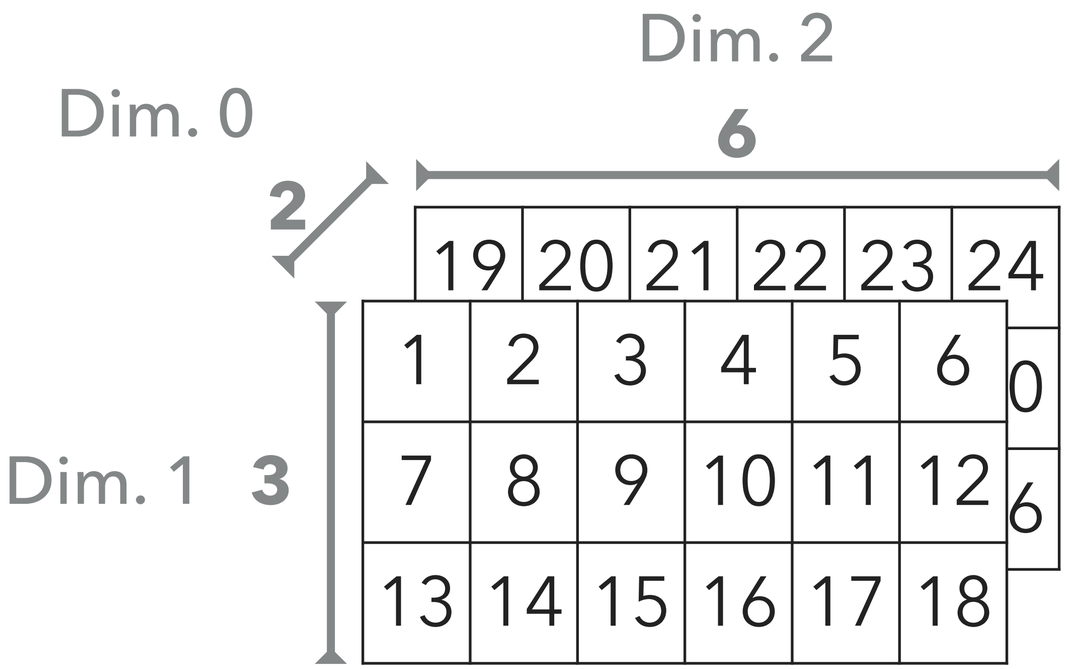
6

3

1	2	3	4	5	6
7	8	9	10	11	12
13	14	15	16	17	18

Multi-Axis Operation (Np.Mean())

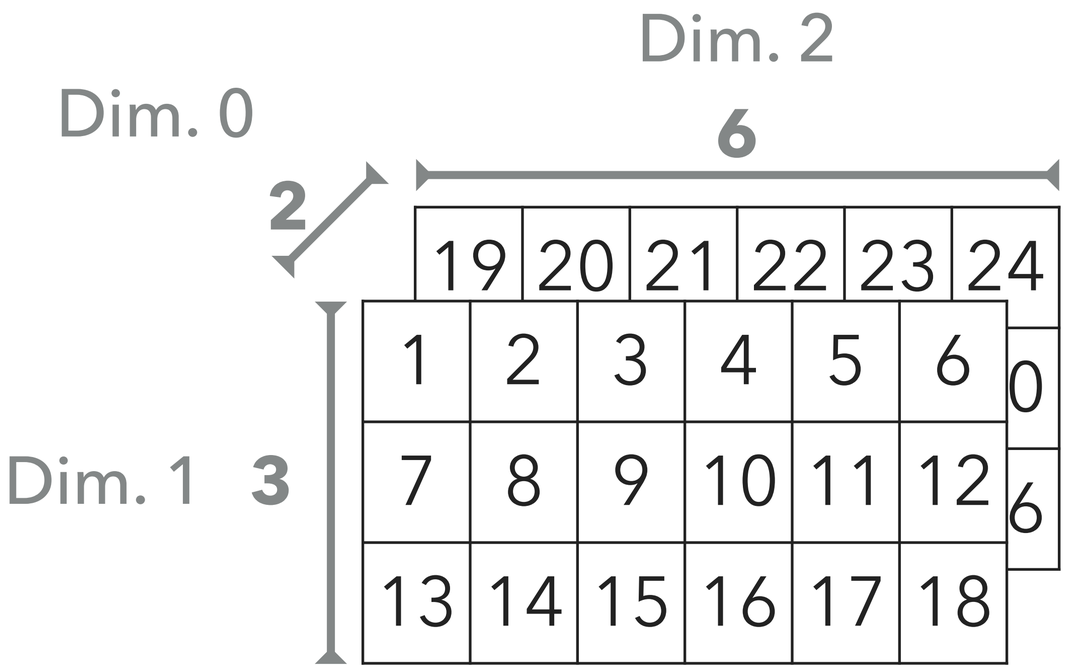
3D matrix: 2 x 3 x 6



e.g., np.mean(X, axis = 0)

Multi-Axis Operation (Np.Mean())

3D matrix: 2 x 3 x 6



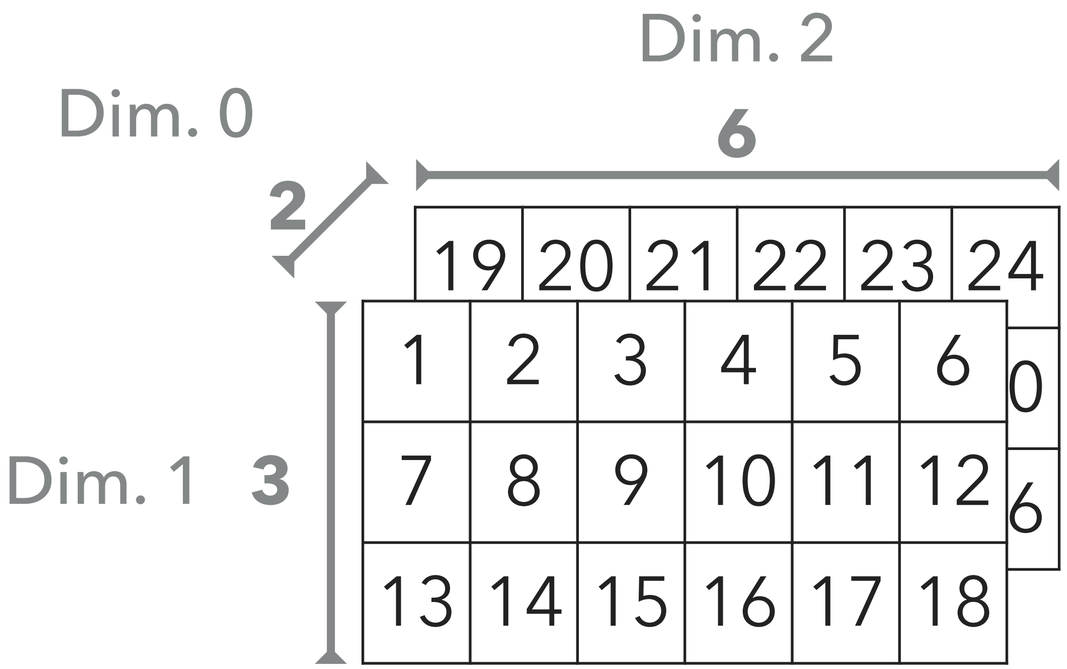
Axis = 0 ––––> 3 x 6

10	11	12	13	14	15
16	17	18	19	20	21
22	23	24	25	26	27

e.g., np.mean(X, axis = 0)

Multi-Axis Operation (Np.Mean())

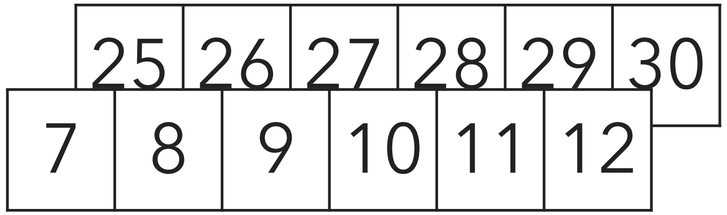
3D matrix: 2 x 3 x 6



Axis = 0 ––––> 3 x 6

10	11	12	13	14	15
16	17	18	19	20	21
22	23	24	25	26	27

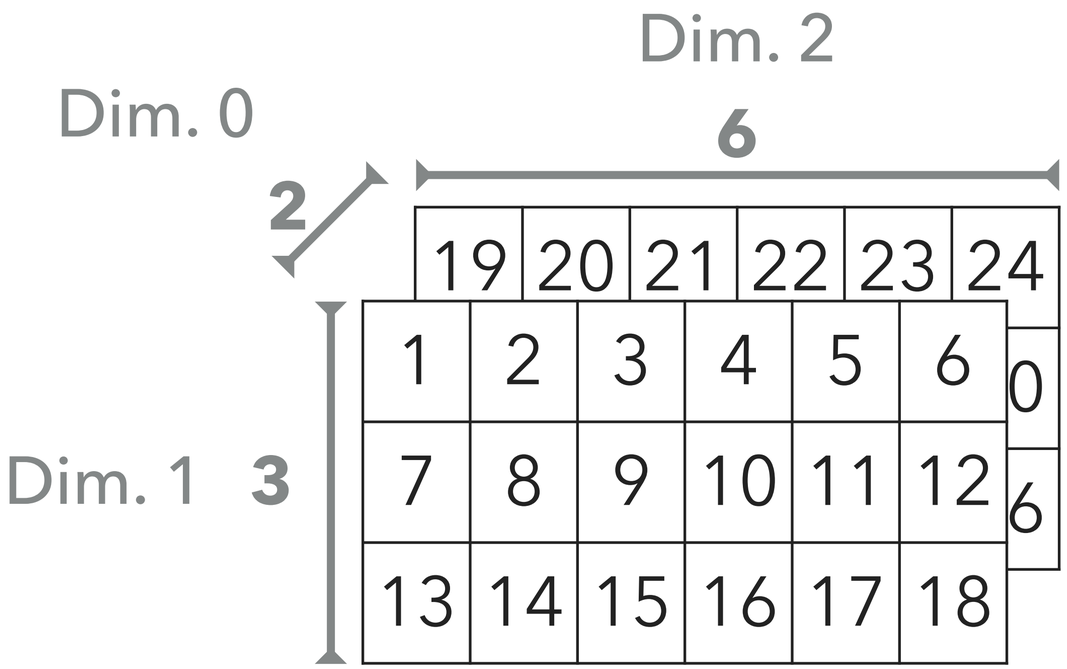
Axis = 1 ––––> 2 x 6



e.g., np.mean(X, axis = 0)

Multi-Axis Operation (Np.Mean())

3D matrix: 2 x 3 x 6

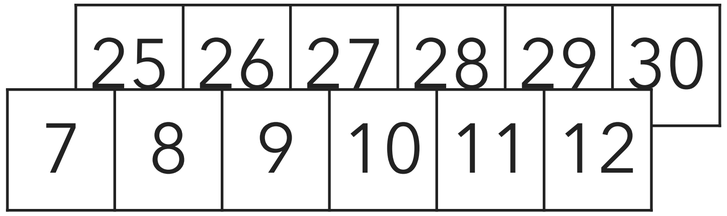


e.g., np.mean(X, axis = 0)

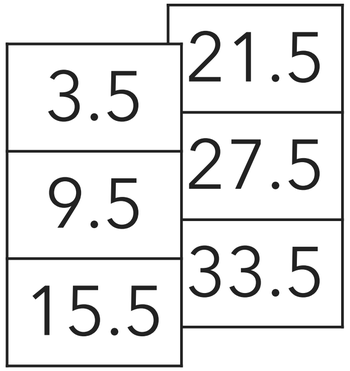
Axis = 0 ––––> 3 x 6

10	11	12	13	14	15
16	17	18	19	20	21
22	23	24	25	26	27

Axis = 1 ––––> 2 x 6

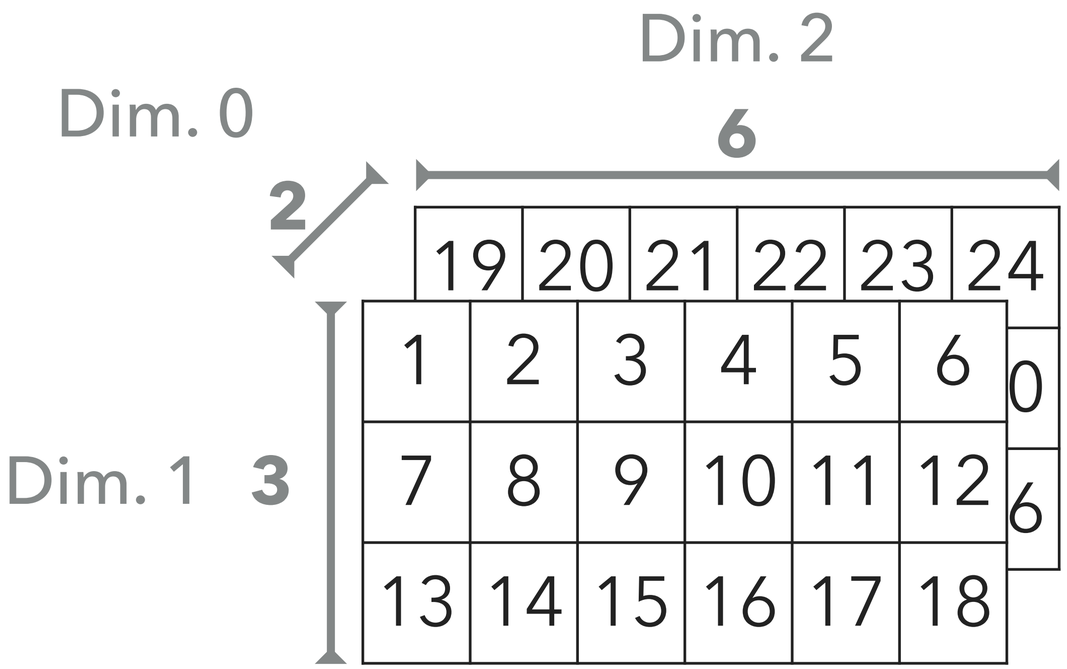


Axis = 2 ––––> 2 x 3



Multi-Axis Operation (Np.Mean())

3D matrix: 2 x 3 x 6



e.g., np.mean(X, axis = 0)

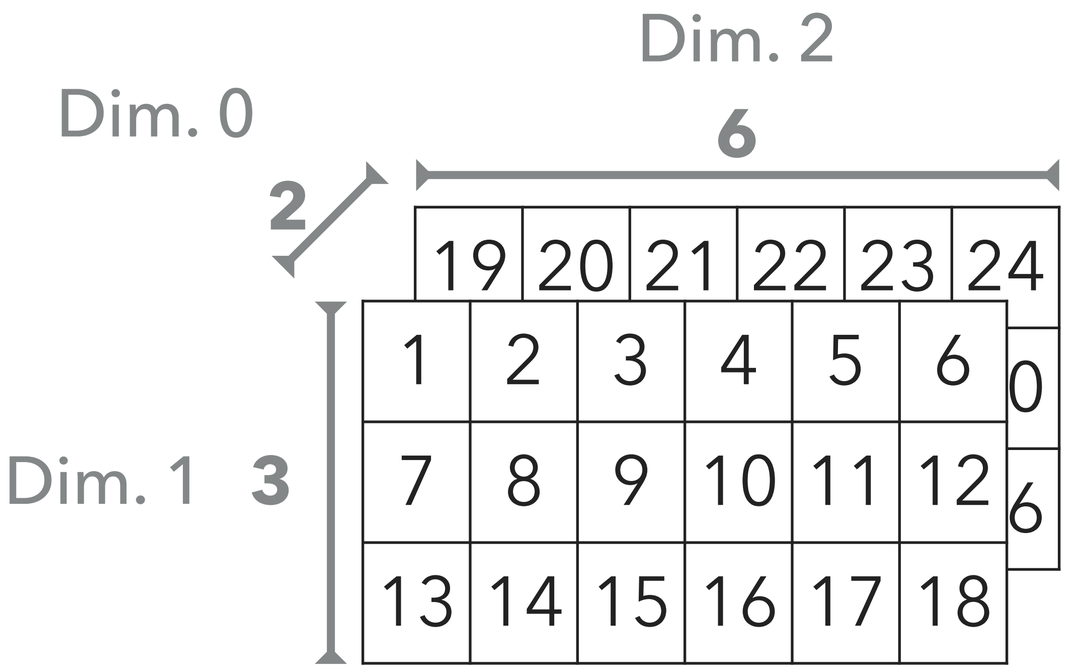
Axis = 0 ––––> 3 x 6

10	11	12	13	14	15
16	17	18	19	20	21
22	23	24	25	26	27

Axis = (0, 1)

Multi-Axis Operation (Np.Mean())

3D matrix: 2 x 3 x 6



e.g., np.mean(X, axis = 0)

Axis = 0 ––––> 3 x 6

10	11	12	13	14	15
16	17	18	19	20	21
22	23	24	25	26	27

Axis = (0, 1) ––––> 1 x 6

16	17	18	19	20	21

Multi-Axis Operation (Np.Mean())

3D matrix: 2 x 3 x 6

Dim. 2

Dim. 0

6

2

19	20	21	22	23	24

Dim. 1 **3**

1	2	3	4	5	6	0
7	8	9	10	11	12	6
13	14	15	16	17	18	

e.g., np.mean(X, axis = 0)

Axis = 0 ––––> 3 x 6

10	11	12	13	14	15
16	17	18	19	20	21
22	23	24	25	26	27

Axis = (0, 1) ––––> 1 x 6

16	17	18	19	20	21

Axis = (0, 2) ––––> 1 x 3

12.5
18.5
24.5

Overview

Dictionary

Dictionary is a collection of key-value pairs

In computer science, an **associative array**, **map**, **symbol table**, or **dictionary** is an abstract data type that stores a collection of (key, value) pairs, such that each possible key appears at most once in the collection. In mathematical terms an associative array is a function with *finite* domain.[1] It supports 'lookup', 'remove', and 'insert' operations.

insert

lookup

remove

https://en.wikipedia.org/wiki/Associative_array

Dictionary - Instantiate a Dictionary

List [value1, value2, value3]

```
nested_list = [['John', 20], ['Mary', 25], ['Michael', 30]]
```

Dictionary {key1 : value1, key2 : value2, key3 : value3}

```
person_age = {'John': 20, 'Mary': 25, 'Michael': 30, 'Elizabeth': 35}
```

Dictionary - Insertion

Dictionary {key1 : value1, key2 : value2, key3 : value3}

```
person_age = {'John': 20, 'Mary': 25, 'Michael': 30, 'Elizabeth': 35}
```

Use squared brackets (indexing)

```
person_age['David'] = 40
print(person_age) # {'John': 20, 'Mary': 25, 'Michael': 30, 'Elizabeth':
35, 'David': 40}
```

Use .update() method

```
person_age.update({'David': 40})
print(person_age) # {'John': 20, 'Mary': 25, 'Michael': 30, 'Elizabeth':
35, 'David': 40}
```

Dictionary - Lookup

Dictionary {key1 : value1, key2 : value2, key3 : value3}

```
person_age = {'John': 20, 'Mary': 25, 'Michael': 30, 'Elizabeth': 35}
```

Use squared brackets (indexing)

```
john_age = person_age['John']
print(john_age) # 20
```

Use .get() method

```
john_age = person_age.get('John')
print(john_age) # 20
```

Dictionary - Obtain all Keys or Values

Dictionary {key1 : value1, key2 : value2, key3 : value3}

```
person_age = {'John': 20, 'Mary': 25, 'Michael': 30, 'Elizabeth': 35}
```

Get all keys

```
keys = person_age.keys()
print(keys) # dict_keys(['John', 'Mary', 'Michael', 'Elizabeth', 'David'])
```

Get all values

```
values = person_age.values()
print(values) # dict_values([20, 25, 30, 35, 40])
```

Challenge: How To Fetch the Last Key-Value Pair?

Challenge: How To Fetch the Last Key-Value Pair?


```
# step 1: get a list of keys in a dictionary
keys = person_age.keys()
print(keys) # dict_keys(['John', 'Mary', 'Michael', 'Elizabeth', 'David'])

# step 2: access the last key in the dictionary
last_key = list(keys)[-1]
print(last_key) # David

# step 3: access the value of the last key
last_value = person_age[last_key]
print(last_value) # 40
```

Loop - For Loops

For loop

Iterator

```
for i in [0, 1, 2]:
    print(i)
```

Output

```
# 0
# 1
# 2
```

Loop - For Loops

For loop

Iterator

```
for i in [0, 1, 2]:
    print(i)
```

Output

```
# 0
# 1
# 2
```

Iterator

```
for i in range(3):
    print(i)
```

Output

```
# 0
# 1
# 2
```

Loop - For Loops

For loop

```
person_age = {'John': 20, 'Mary': 25, 'Michael': 30, 'Elizabeth': 35}
```

```
for i in person_age:
    print(i)
```

Loop - For Loops

For loop

```
person_age = {'John': 20, 'Mary': 25, 'Michael': 30, 'Elizabeth': 35}
```

Only "**keys**" were printed out

```
for i in person_age:
    print(i)
# John
# Mary
# Michael
# Elizabeth
```

Loop - For Loops

For loop

```
person_age = {'John': 20, 'Mary': 25, 'Michael': 30, 'Elizabeth': 35}
```

Only "**keys**" were printed out

```
for i in person_age:
    print(i)
# John
# Mary
# Michael
# Elizabeth
```

Use an **index** to fetch the corresponding value

```
for key in person_age:
    print(person_age[key])
# 20
# 25
# 30
# 35
```

Loop - for Loops (Slow Motion)


```
person_age = {'John': 20, 'Mary': 25, 'Michael': 30, 'Elizabeth': 35}
```

```
for key in person_age:
    print(person_age[key])
# 20
# 25
# 30
# 35
```

```
print(person_age['John'])
print(person_age['Mary'])
print(person_age['Michael'])
print(person_age['Elizabeth'])

# 20
# 25
# 30
# 35
```

Loop - For Loops


```
person_age = {'John': 20, 'Mary': 25, 'Michael': 30, 'Elizabeth': 35}
```

```
for key, value in person_age.items():
    print(key, value)
# John 20
# Mary 25
# Michael 30
# Elizabeth 35
```

Loop - While Loops

While loop

```
i = 0
while i < 3:
    print(i)
    i = i + 1
```

Output

```
# 0
# 1
# 2
```

Loop - While Loops

While loop

```
i = 0
while i < 3:
    print(i)
    i = i + 1
```

```
i = 0
if i < 3:
    print(i)
    i = i + 1
    if i < 3:
        print(i)
        i = i + 1
        if i < 3:
            print(i)
            i = i + 1
            # ... and so on
        else:
            pass
    else:
        pass
else:
    pass
```

Output

```
# 0
# 1
# 2
```

Loop - Break and Continue

break

```
for i in range(5):
    if i == 2:
        break
    print(i)
# 0
# 1
```

continue

```
for i in range(5):
    if i == 2:
        continue
    print(i)
# 0
# 1
# 3
# 4
```