

# SESSION 4

## Pandas

1. What is Pandas?
2. Pandas Series
3. Datatype handling in Panda Series
4. Appending in Panda Series
5. Concatenation in Panda Series

### 1. What is Pandas?

- **Pandas is a Python library used for working with data sets.**
- **It has functions for analyzing, cleaning, exploring, and manipulating data.**

Basically Pandas will store your array data into something that looks like excel sheets.

### Why Use Pandas?

- **Pandas allows us to analyze big data and make conclusions based on statistical theories.**
- **Pandas can clean messy data sets, and make them readable and relevant.**
- **Relevant data is very important in data science.**

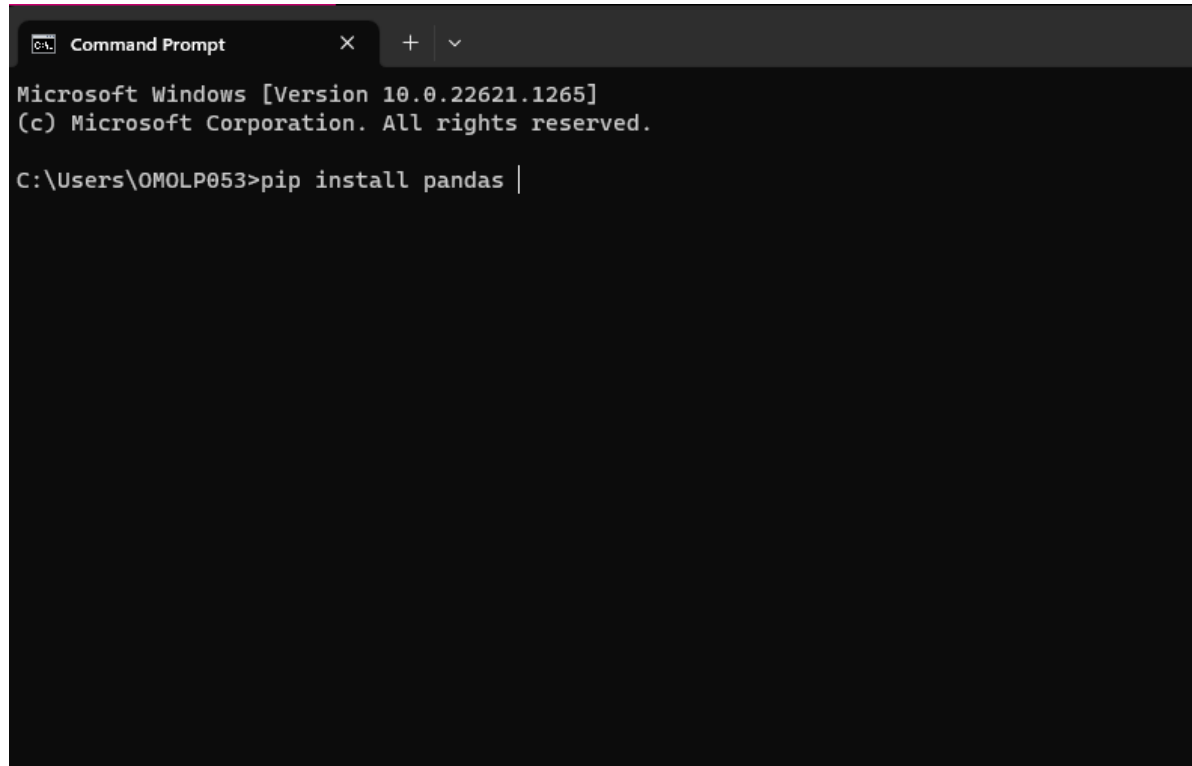
### Installation of Pandas

For installing any module in python we can use python installation package i.e pip we can use the following command any module in python.

```
pip install module_name
```

Now we will install pandas

- pip install pandas

A screenshot of a Windows Command Prompt window. The title bar shows 'Command Prompt' with standard window controls. The text inside the window reads: 'Microsoft Windows [Version 10.0.22621.1265] (c) Microsoft Corporation. All rights reserved. C:\Users\OMOLP053>pip install pandas |'. The cursor is at the end of the command line.

```
Microsoft Windows [Version 10.0.22621.1265]
(c) Microsoft Corporation. All rights reserved.

C:\Users\OMOLP053>pip install pandas |
```

## Import Pandas

```
In [ ]: import pandas
```

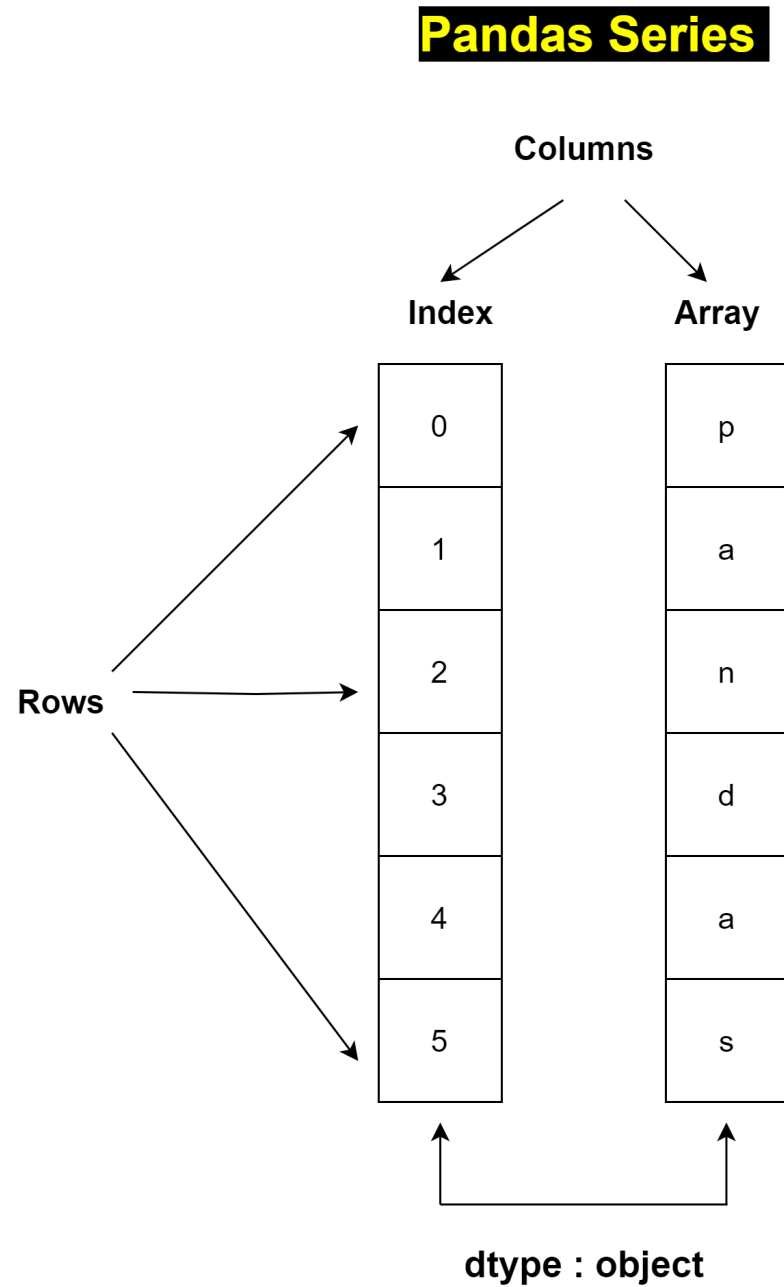
## Pandas as pd

Pandas is usually imported under the pd alias.

```
In [ ]: import pandas as pd
import numpy as np
info = np.array(['P', 'a', 'n', 'd', 'a', 's'])
```

```
a = pd.Series(info)
print(a)
```

```
0    P
1    a
2    n
3    d
4    a
5    s
dtype: object
```



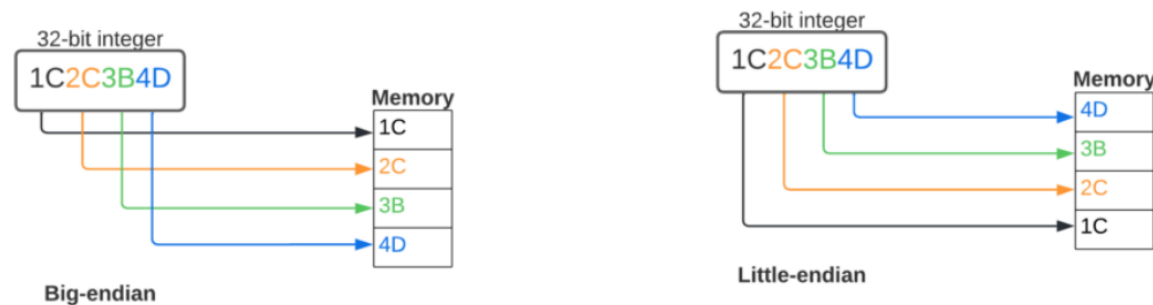
# Data type Object (dtype) in NumPy Python

Every ndarray has an associated data type (dtype) object. This data type object (dtype) informs us about the layout of the array. This means it gives us information about:

- Type of the data (integer, float, Python object, etc.)
- Size of the data (number of bytes)
- If the data type is a sub-array, what is its shape and data type?
- The byte order of the data (little-endian or big-endian)

## Define little-endian & big-endian:

We define endianness as the order of bytes inside a word of data stored in computer memory. Big-endian and little-endian are the two main ways to represent endianness. Big-endian keeps the most significant byte of a word at the smallest memory location and the least significant byte at the largest. On the other hand, little-endian keeps the least significant address at the smallest memory location.



## 2. Pandas Series

### What is a Series?

- A Pandas Series is like a column in a table.
- It is a one-dimensional array holding data of any type.

```
In [ ]: # Create a simple Pandas Series from a List:
```

```
import pandas as pd
```

```
a = [1, 7, 2]
```

```
myvar = pd.Series(a)
```

```
print(myvar)
```

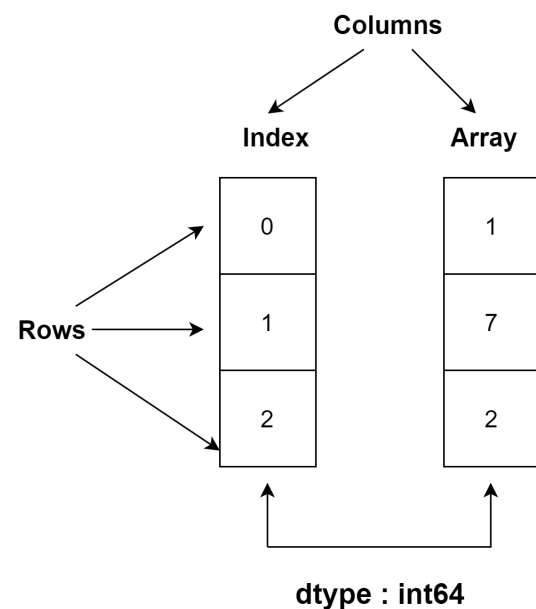
```
0    1
```

```
1    7
```

```
2    2
```

```
dtype: int64
```

## Pandas Series



```
a = [1, 7, 2]
```

As we have passed a list to our `pd.Series(a)` method ,  
you can notice that it is return us `int64` as dtype

```
In [ ]: import pandas as pd
```

```
a = [1, 7, 2]
```

```
myvar = pd.Series(a)

print(myvar[0])
```

1

## Labels Creation

**With the index argument, you can name your own labels.**

In [ ]:

```
# Create your own labels:

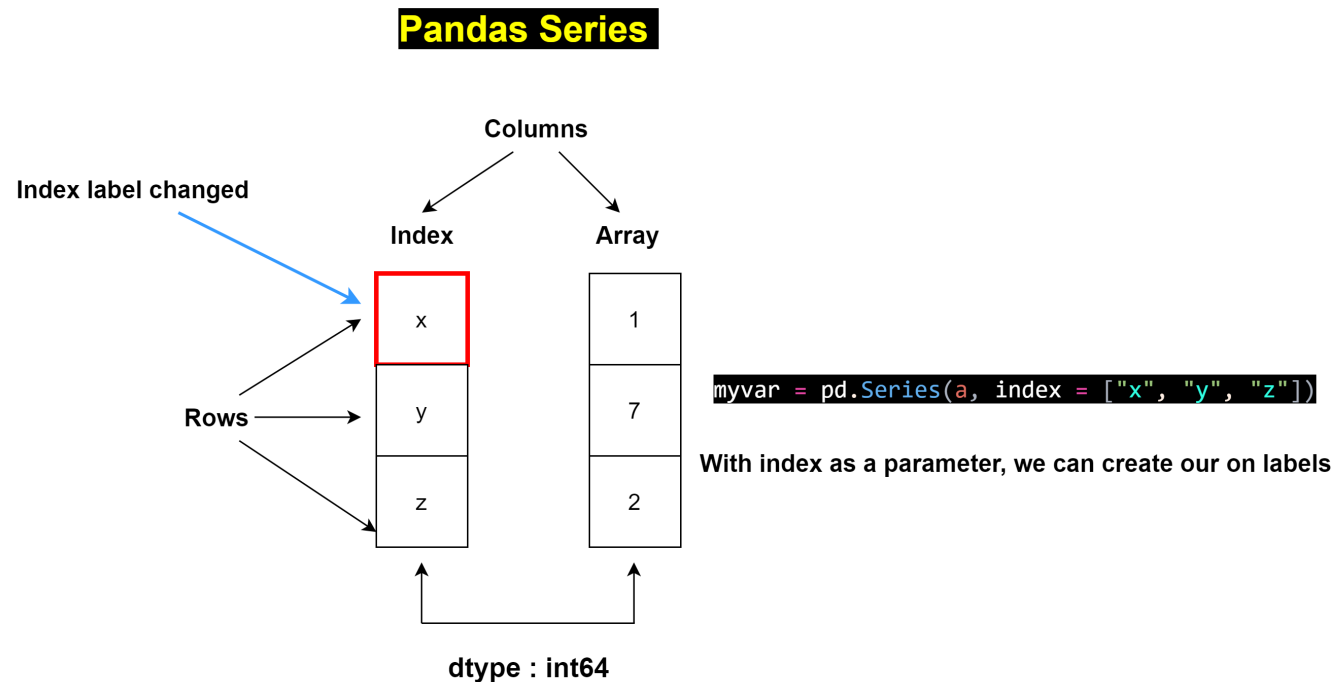
import pandas as pd

a = [1, 7, 2]

myvar = pd.Series(a, index = ["x", "y", "z"])

print(myvar)
```

```
x    1
y    7
z    2
dtype: int64
```



```
In [ ]: import pandas as pd

a = [1, 7, 2]

myvar = pd.Series(a, index = ["x", "y", "z"])

print(myvar["y"])
```

7

### 3. Datatype handling in Panda Series

#### Key/Value Objects as Series

- You can also use a key/value object, like a dictionary, when creating a Series.

```
In [ ]: import pandas as pd

calories = {"day1": 420, "day2": 380, "day3": 320}
```

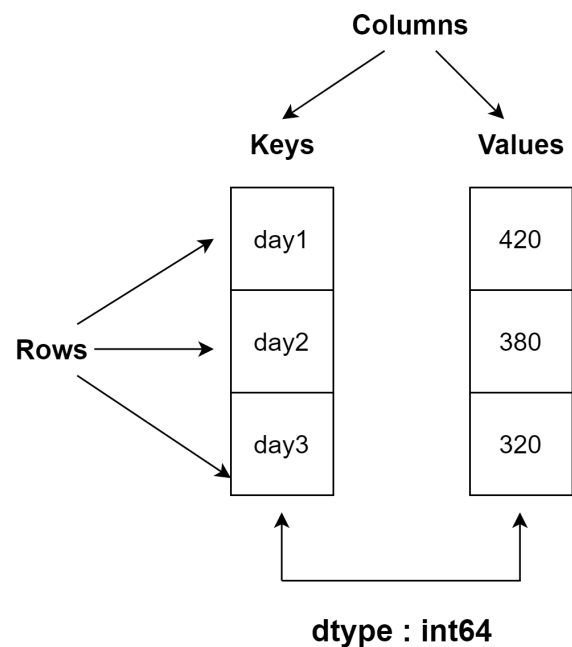


```
myvar = pd.Series(calories)

print(myvar)
```

```
day1    420
day2    380
day3    390
dtype: int64
```

## Pandas Series



Passing Dictionary to `pd.Series()` method

```
calories = {"day1": 420, "day2": 380, "day3": 390}

myvar = pd.Series(calories)
```

- To select only some of the items in the dictionary, use the index argument and specify only the items you want to include in the Series.

### Example

- Create a Series using only data from "day1" and "day2":

```
In [ ]: import pandas as pd
```

```
calories = {"day1": 420, "day2": 380, "day3": 320}

myvar = pd.Series(calories, index = ["day1", "day2"])

print(myvar)
```

```
day1    420
day2    380
dtype: int64
```

## Pandas Series

Passing Dictionary to `pd.Series()` method



## Create a Series from dict

- A dict can be passed as input and if no index is specified, then the dictionary keys are taken in a sorted order to construct index.

In [ ]:

```
import pandas as pd
import numpy as np
data = {'a' : 0., 'b' : 1., 'c' : 2.}
data = pd.Series(data)
print(data)
```

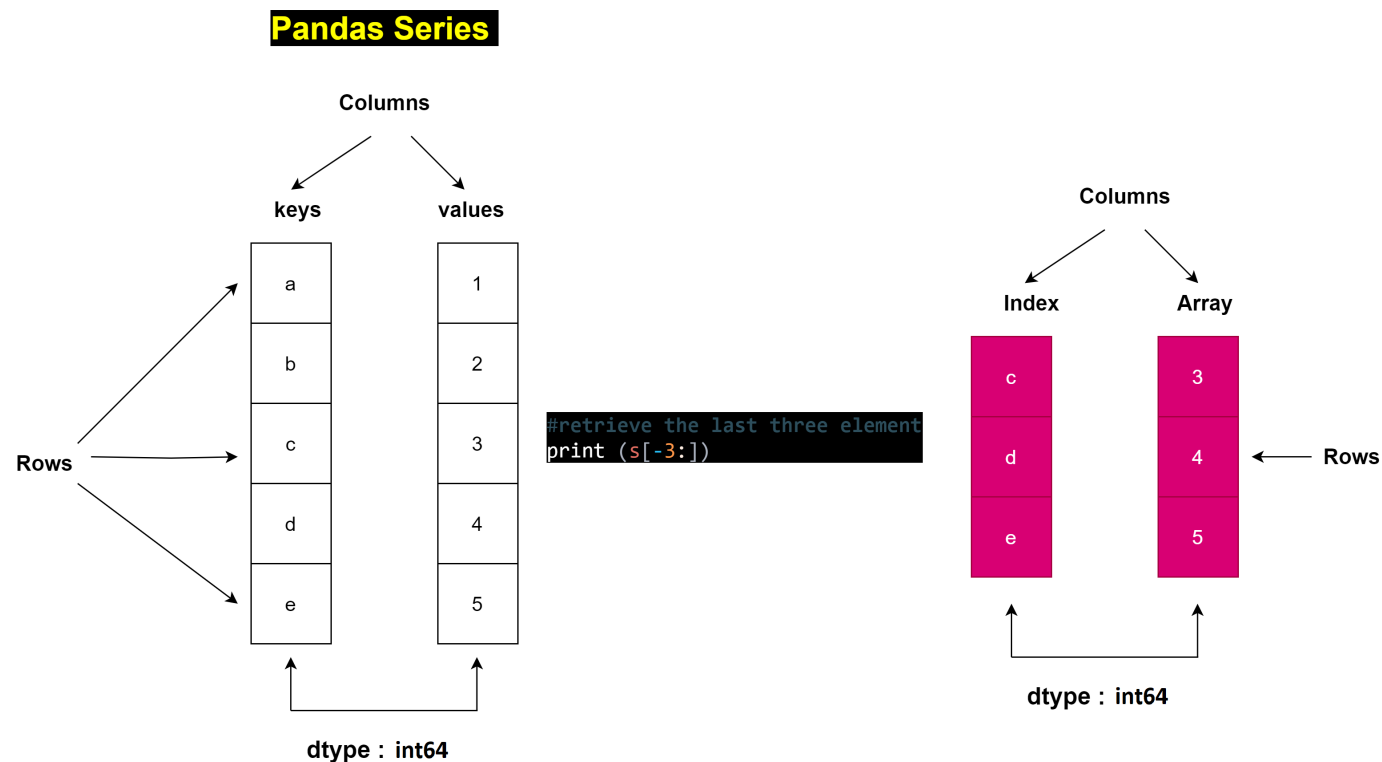
```
a    0.0  
b    1.0  
c    2.0  
dtype: float64
```

## Indexing on Series:

- *If index is passed, the values in data corresponding to the labels in the index will be pulled out.*

```
In [ ]: # Retrieve the last three elements.  
import pandas as pd  
s = pd.Series([1,2,3,4,5],index = ['a','b','c','d','e'])  
  
#retrieve the last three element  
print (s[-3:])
```

```
c    3  
d    4  
e    5  
dtype: int64
```



## 4. Appending in Panda Series

Add a panda series to another series using `append()`

- It appends one series object at the end of another series object and returns an appended series.
- The attribute, `ignore_index=True` is used when we do not use index values on appending, i.e the resulting index will be 0 to n-1. By default, the value of the `ignore_index` attribute is `False`

In [ ]:

```
# importing the module
import pandas as pd

# create 2 series objects
series_1 = pd.Series([2, 4, 6, 8])
series_2 = pd.Series([10, 12, 14, 16])

# adding series_2 to series_1 using the append() function
```

```
series_1 = series_1.append(series_2, ignore_index=True)

# displaying series_1
print(series_1)
```

## Pandas Series append() method

**series\_1**

2	4	6	8
---	---	---	---

**series\_2**

10	12	14	16
----	----	----	----

```
# adding series_2 to series_1 using the append() function
series_1 = series_1.append(series_2, ignore_index=True)
```

0	2
1	4
2	6
3	8
0	10
1	12
2	14
3	16

## 5. Concatenation in Panda Series

### Using the concat() function

*It takes a list of series objects that are to be concatenated as an argument and returns a concatenated series along an axis, i.e., if axis=0, it will concatenate row-wise and if axis=1, the resulting Series will be concatenated column-wise.*

In [ ]:

```
# importing the module
import pandas as pd

# create 2 series objects
series_1 = pd.Series([2, 4, 6, 8])
series_2 = pd.Series([10, 12, 14, 16])

# adding series_2 to series_1 using the concat() function
series_1 = pd.concat([series_1, series_2], axis=0)

# displaying series_1
print(series_1)
```

```
0    2
1    4
2    6
3    8
0   10
1   12
2   14
3   16
dtype: int64
```

## Pandas Series concat() method

**series\_1**

2	4	6	8
---	---	---	---

**series\_2**

10	12	14	16
----	----	----	----

```
series_1 = pd.concat([series_1, series_2], axis=0)
```

axis = 0, joins the data row-wise

0	2
1	4
2	6
3	8
0	10
1	12
2	14
3	16

In [ ]:

```
# importing the module
import pandas as pd

# create 2 series objects
series_1 = pd.Series([2, 4, 6, 8])
series_2 = pd.Series([10, 12, 14, 16])

# adding series_2 to series_1 using the concat() function
series_1 = pd.concat([series_1, series_2], axis=1)

# displaying series_1
print(series_1)
```

```
0    1
0    2    10
```

```
1  4 12
2  6 14
3  8 16
```

### Pandas Series concat() method

**series\_1**

2	4	6	8
---	---	---	---

**series\_2**

10	12	14	16
----	----	----	----

```
series_1 = pd.concat([series_1, series_2], axis=1)
```

axis = 1, joins the data column-wise

	0	1
0	2	10
1	4	12
2	6	14
3	8	16

← Data joined column wise

## Homework Questions

1. Write a Python program to create a simple DataFrame with two columns ('Name' and 'Age') and populate it with data. Then print the entire DataFrame.

1. Create a pandas dataframe with the following data:

	Name	Age	Gender
0	John	25	Male
1	Jane	30	Female
2	Mark	20	Male

1. To select only some of the items in the dictionary, use the index argument and specify only the items you want to include in the Series.

```
vitamin = {"day1": 420, "day2": 380, "day3": 390}
```

1. Add a panda series to another series using append()

```
a=[12,23,65,87]
```

```
b=[78,52,33,61]
```



1. use the concat() to combine the following arrays

```
series_1 = pd.Series([2, 4, 6, 8])  
series_2 = pd.Series([10, 12, 14, 16])
```

1. Create a pandas series with the following data:

```
0    10  
1    20  
2    30  
3    40  
dtype: int64
```

1. Create a pandas series with the following data and index labels:

```
a    10  
b    20  
c    30  
d    40  
dtype: int64
```

1. Create a pandas series from a dictionary with the following data:

```
{ 'a': 10, 'b': 20, 'c': 30, 'd': 40 }
```

---

For solutions of Homework questions, please refer to the `HomeworkSolution.ipynb` file