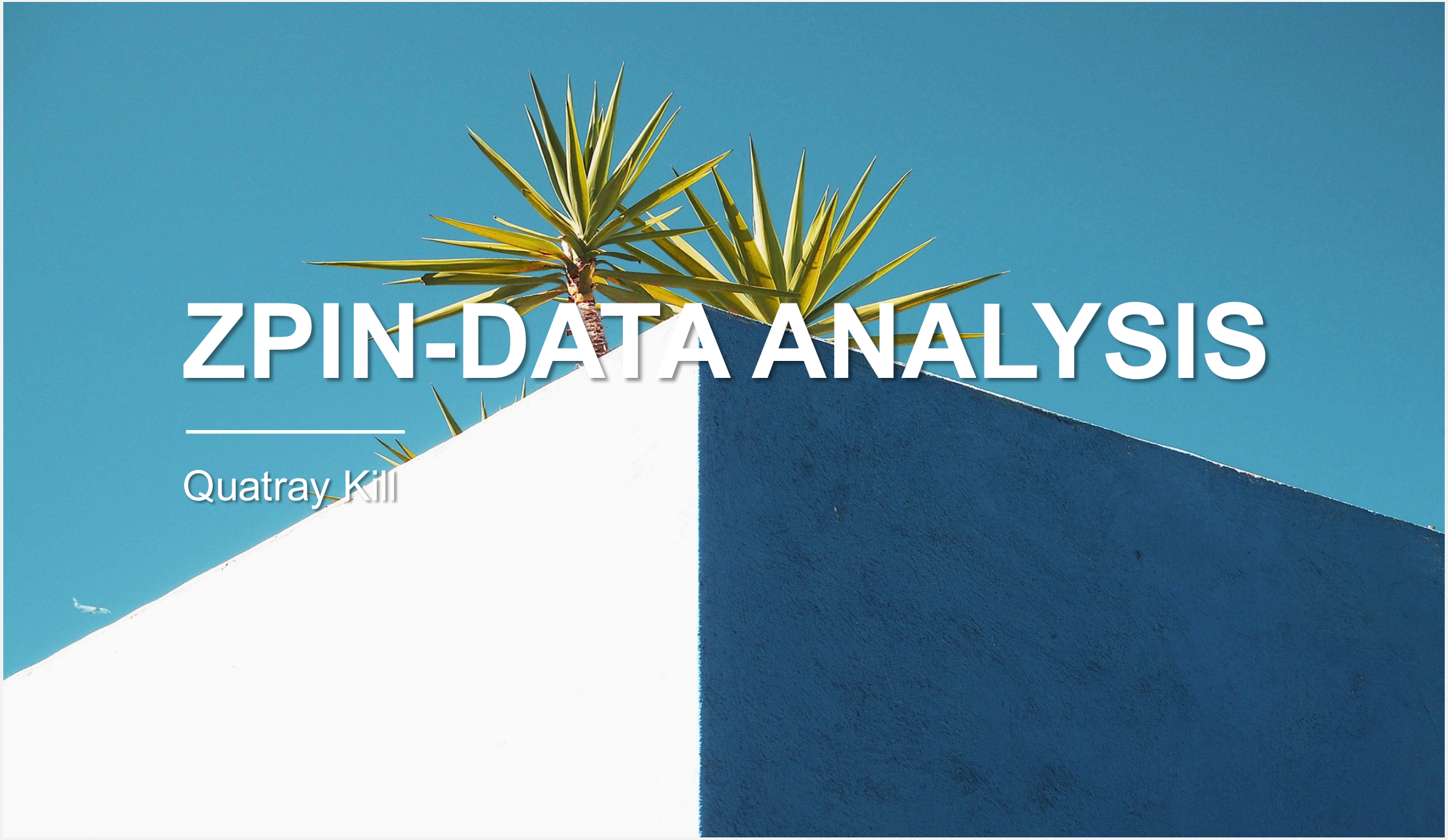


ZPIN-DATA ANALYSIS

Quatray Kill

MEET OUR GROUP! !

徐心娴

夏瑞欣

关若萱

曹怡惠

CONTENTS

WHY

WHAT

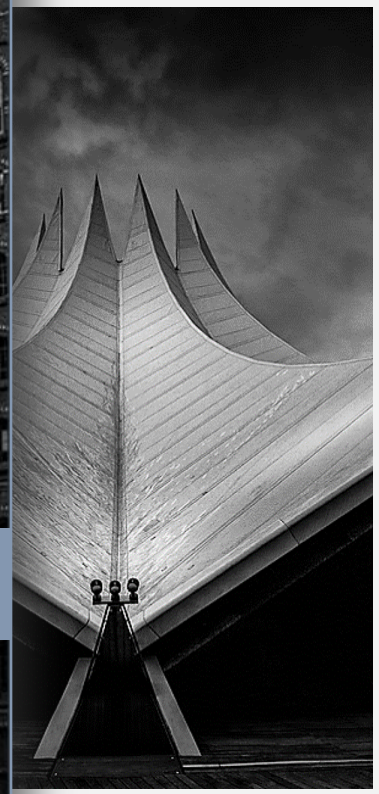
HOW

RESULT

NO

Business is the human activity related to material things. It is necessary for civilization.

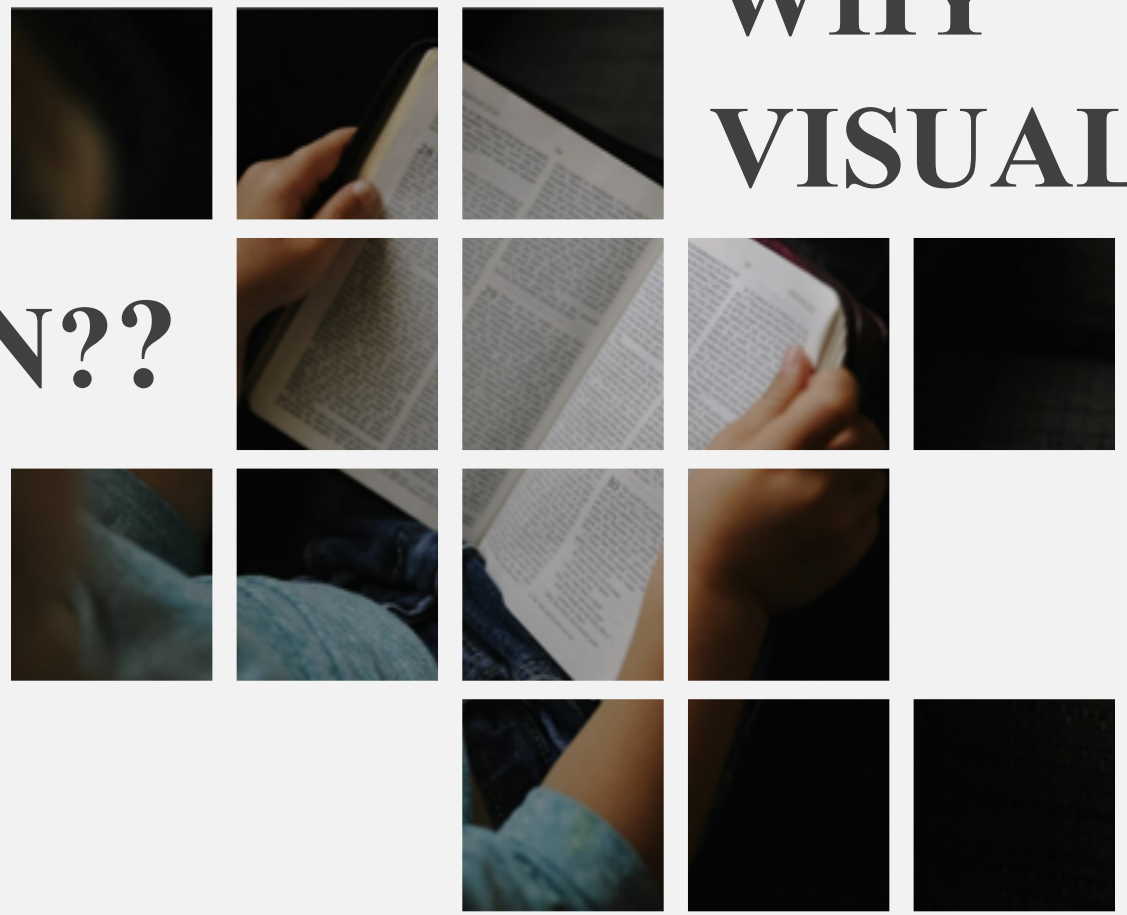
WHY WE DO THE PROJECT



WHY ZPIN??

- a large-scale company providing human resources information and service for various companies in diverse fields..

- an official and professional organization that has both talent service license and labor dispatch license issued by the government.



WHY VISUALIZATION??

- lacks systematic classification and analysis
- troublesome to find targeted information
- lack a clear visualization of job-related issues

NO

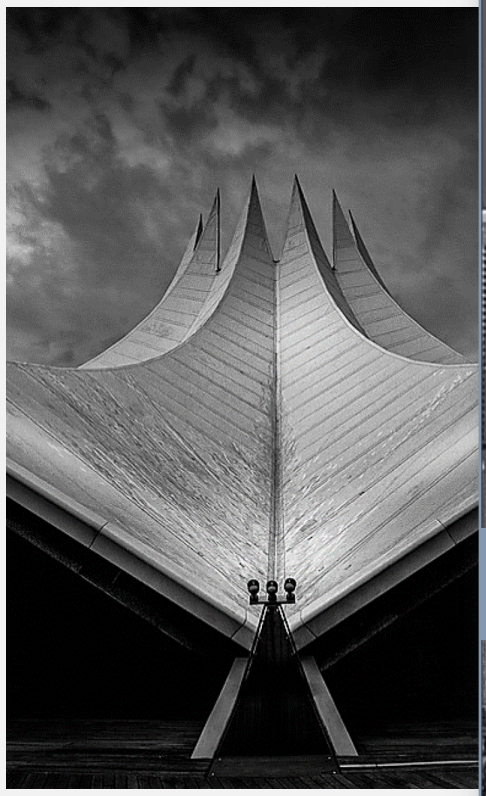
Business is the human activity related to material things. It is necessary for civilization.

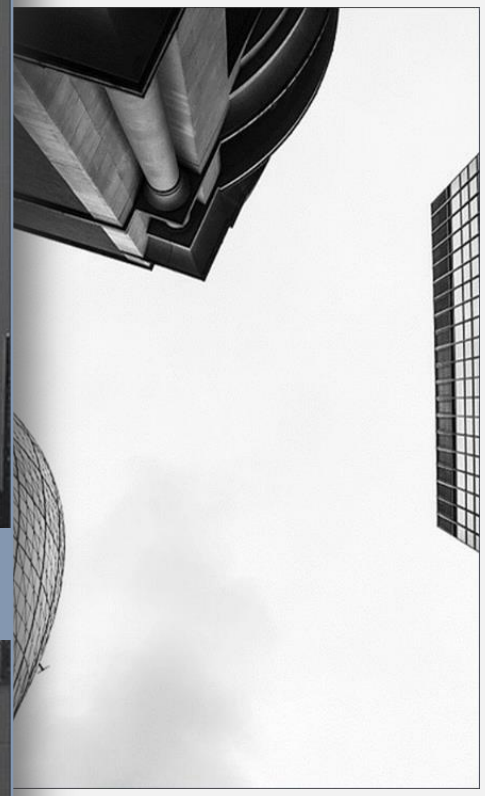
WHAT SHOULD WE DO

- Key Word Categorization
- Data Processing
- Data Analysis & Visualization

NO

Business is the human activity related to material things. It is necessary for civilization.





Key Word Categorization

Firstly we need to crawl date on the target page so as to do the further analysis. There're various methods to crawl data on a web, such as requests, BeautifulSoup and scrapy. As novices here we use <scrapy> to grab data on ZPIN.

Advantages of SCRAPY

- Having a fast and high-level screen grabbing framework
- Anyone can easily modify it according to their needs
- Providing the basic classes of plenty kinds of reptiles (spider, sitemap crawler……)

Key Word Categorization

Four steps of using scrapy

(1) Select a web where you want to crawl data.

(2)Define grabbing Items

(3) Write a spider to grab the data. (Including data crawl rules)

(4) Run the spider you write to grab the data.

·Based on the universal steps, we can use it to help grab data on ZPIN.

Key Word Categorization

- Select a web
- Define Items
- Write a spider
- Run and store

We plan to analyze the data of employment information on investment banking from ZPIN. So we choose a web that contains the searching result:

[http://sou.zhaopin.com/jobs/searchresult.ashx?jl=全国&kw=投行&sm=0&p=1&isfilter=0&fl=489&isadv=0]

Key Word Categorization

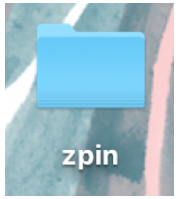
- Select a web
- Define Items
- Write a spider
- Run and store

Before definition, open terminal to pip scrapy and start a program.

- Open terminal

```
pip install scrapy
cd Desktop/    #change the path to find the Folder more easily.
scrapy startproject zpin    #start a project named "zpin"
```

Then you'll find a folder named "zpin" on your desktop. Use PyCharm to open the [items.py] in the folder to definite our 4 aimed grabbing items.



items.py

```
import scrapy
class ZpinItem(scrapy.Item):
    # define the fields for your item here like:
    # name = scrapy.Field()
    salary = scrapy.Field()
    job = scrapy.Field ()
    company = scrapy.Field ()
    place = scrapy.Field ()
    pass
```

(The items to scrapy)

Key Word Categorization

- Select a web
- Define Items
- Write a spider
- Run and store

After the definition of grabbing items, we need to write a spider to grab the data. Open the folder named [spiders], then create a new file [zpinnn.py]

spiders

zpinnn.py

```
import scrapy
import string
from zpin.items import ZpinItem
# input the url
class zpinnn(scrapy.Spider):
    name = "zpinnn"
    allowed_domains = ["zhaopin.com"]
    start_urls = [
        "http://sou.zhaopin.com/jobs/searchresult.ashx?jl=全国&kw=投行
&sm=0&p=1&isfilter=0&fl=489&isadv=0"
    ]

    def parse(self, response):
        '''
        爬虫程序响应函数
        :param response:
        :return:
        '''
        url = response.urljoin (self.start_urls[0])
        yield scrapy.Request (url, callback=self.parse_response)

# Filter to get the work list
    def parse_response(self, response):
```

Key Word Categorization

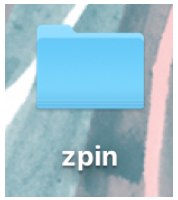
- Select a web
- Define Items
- Write a spider
- Run and store

```
        job_list = response.xpath
("//div[@id='newlist_list_content_table']/table[position()>1]/tr[1]")
# Filter to get the aimed data
        for job_msg in job_list:
            job = job_msg.xpath ("td[@class='zwmc']/div/a").xpath
("string(.)").extract ()[0]
            company = job_msg.xpath ("td[@class='gsmc']/a").xpath
("string(.)").extract ()[0]
            salary = job_msg.xpath ("td[@class='zwyx']").xpath
("string(.)").extract ()[0]
            place =  job_msg.xpath ("td[@class='gzdd']").xpath
("string(.)").extract ()[0]
# Encapsulation into item objects
            item = ZpinItem ()
            item['job'] = job
            item['company'] = company
            item['salary'] = salary
            item['place'] = place
            yield item
# Turn the page of the web
        next_page = response.xpath
("//div[@class='pagesDown']/ul/li/a/@href").extract ()
        for page in next_page:
            page = response.urljoin (page)
            yield scrapy.Request (page, callback=self.parse_response)
```

Key Word Categorization

- Select a web
- Define Items
- Write a spider
- Run and store

So far, we've finished writing all the code of data crawl. To put it into further use, we need to store the data into a json file, which can be more convenient to analyze.



zpin.json

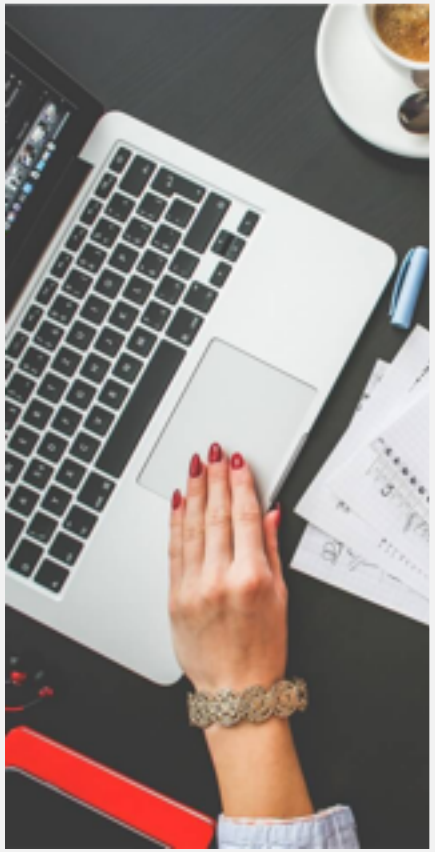
- Open terminal

```
cd Desktop/
cd zpin     # change the path
scrapy crawl zpinnn -o zpin.json -t json
```

"json" is a lightweight data exchange format. It is easy to read and write. It is also easy for machine to parse and generate.

Key Word Categorization

So far, we've gotten the json file we need, storing all the relevant items of different recruitment information.



zpin.json

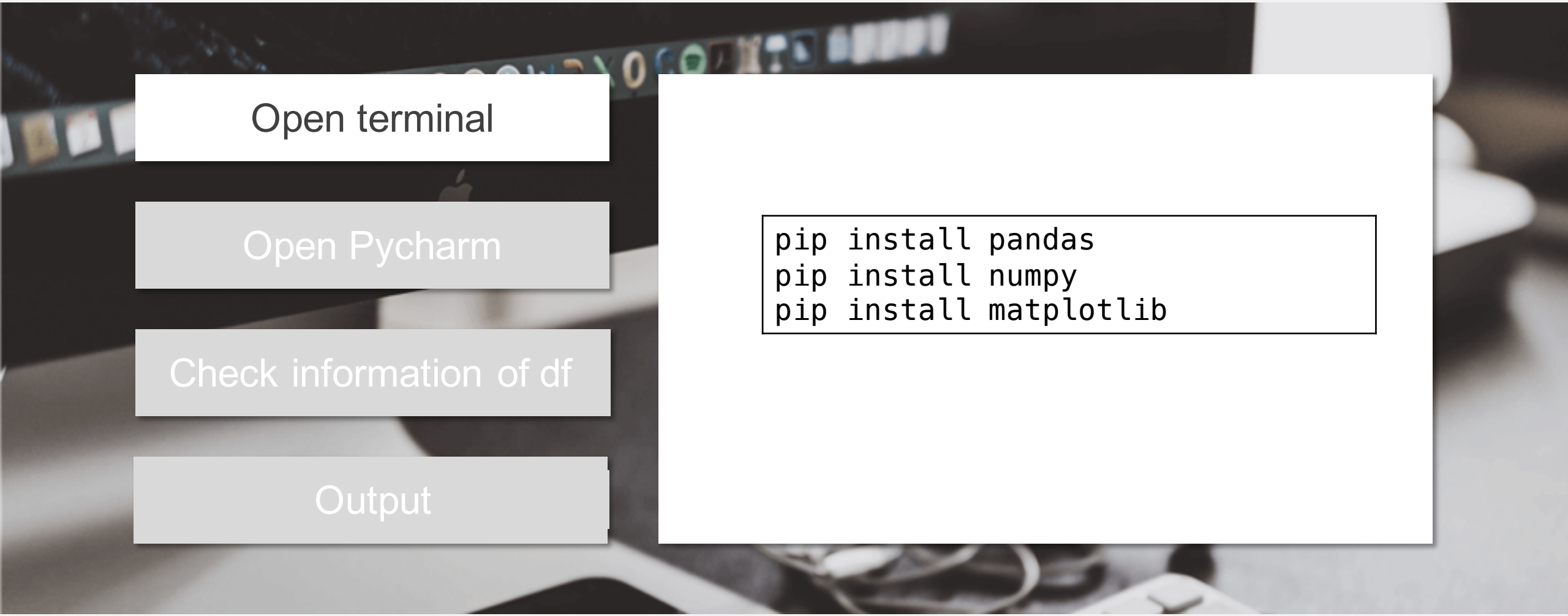
- job
- salary
- company
- place

Data processing- Purpose of Processing

Data we crawled previously is still not ready for **direct analysis**

- Salary data has different forms
- e.g. >=8000 RMB/m
  6000<x<9000 RMB/m
  need consultation (面议)
- Place data are not yet categorized into cities.
- e.g. turning 苏州-姑苏区 >> 苏州

Data processing- Salary Rearrangement



Open terminal

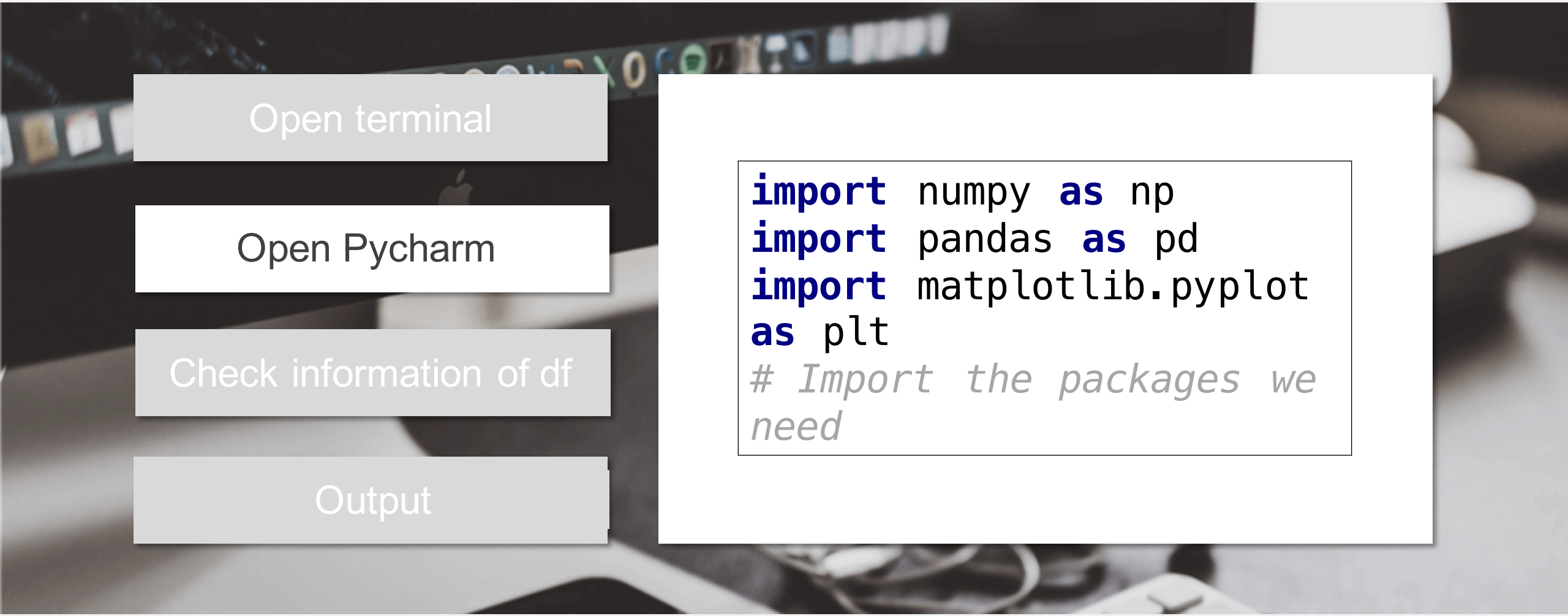
Open Pycharm

Check information of df

Output

```
pip install pandas
pip install numpy
pip install matplotlib
```

Data processing- Salary Rearrangement



Open terminal

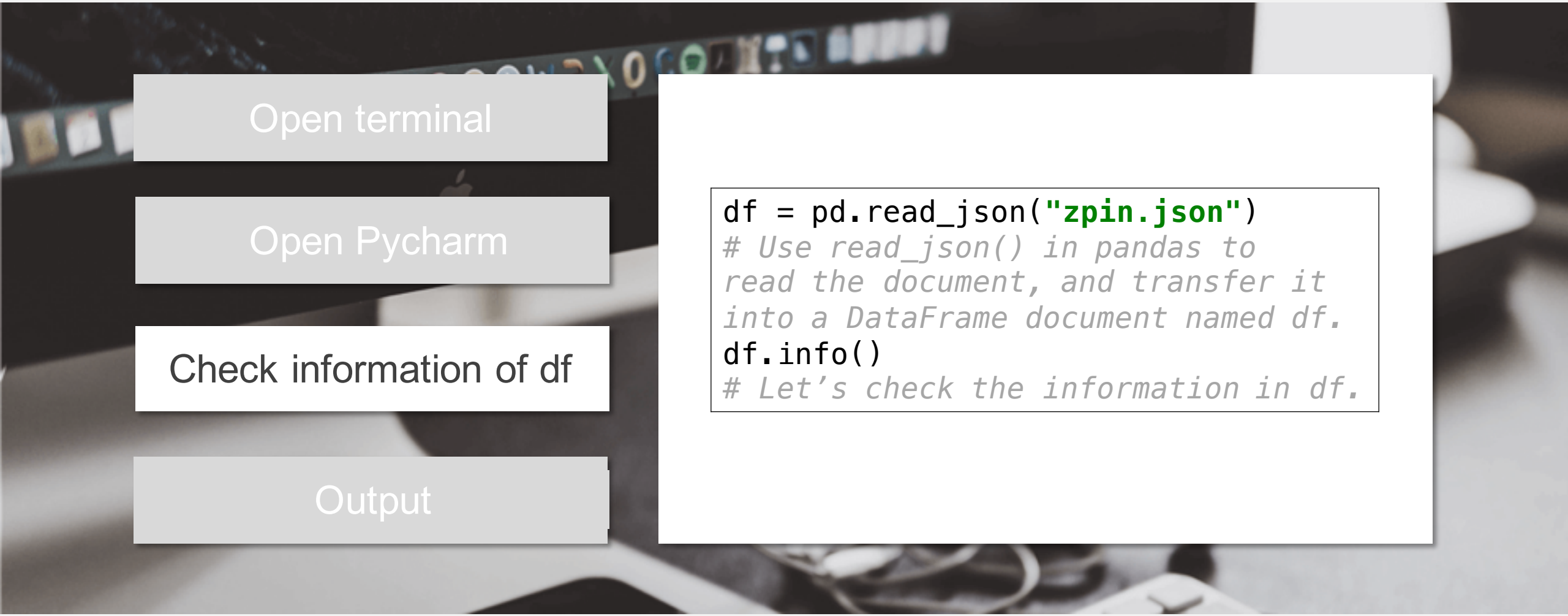
Open Pycharm

Check information of df

Output

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
# Import the packages we need
```

Data processing- Salary Rearrangement



Open terminal

Open Pycharm

Check information of df

Output

```
df = pd.read_json("zpin.json")
# Use read_json() in pandas to
read the document, and transfer it
into a DataFrame document named df.
df.info()
# Let's check the information in df.
```

Data processing- Salary Rearrangement

Open terminal

Open Pycharm

Check information of df

Output

```
Int64Index: 4737 entries, 0 to 4736
Data columns (total 4 columns):
company     4737 non-null object
job         4737 non-null object
place       4737 non-null object
salary      4737 non-null object
dtypes: object(4)
```

Data processing- Salary Rearrangement

- **Serialize** 'salary' column
- **Add 3 columns** 'bottom', 'top', 'average' to store lowest salary, highest salary, and average salary respectively.

- Use q1, q2, q3, q4 to **count the execution times** of each statement in which q1 counts the times of salary such as '6000-8000 RMB/month', and q2, 'not much than 10000 RMB/month'; q3 represents other circumstances such as 'found no element' or 'discuss personally'.

```
import re
df['bottom'] = df['top'] = df['average']
= df['salary']
pattern = re.compile('([0-9]+)')
q1=q2=q3=q4=0
for i in range(len(df['salary'])):
    item = df['salary'].iloc[i].strip()
    result = re.findall(pattern,item)
```

Data processing- Salary Rearrangement

- Use ‘try’ first to execute common circumstances: salary such as ‘6000-8000 RMB/month’ and deal with ‘salary’ column one by one with ‘regular expression’(正则表达式)， storing them into three new columns.

```
  try:
        if result:
            try:
                df['bottom'].iloc[i],df['top'].iloc[i] =
result[0],result[1]
                  df['average'].iloc[i]
=str((int(result[0])+int(result[1]))/2)
                  q1+=1

    except:
                  df['bottom'].iloc[i] = df['top'].iloc[i] =
result[0]
                  df['average'].iloc[i] =
str((int(result[0])+int(result[0]))/2)
                  q2+=1
```

Data processing- Salary Rearrangement

- Use 'else' to execute other circumstances when there 's no numbers such as 'found no element' or 'discuss personally', and keep the original string, that is to keep bottom=top=average='discuss personally'(面议)

```
        else:
        # If this sentence is correctly executed, there's no
numbers in【salary】, such as 'found no element' or 'discuss
personally' .
            df['bottom'].iloc[i] = df['top'].iloc[i] =
df['average'].iloc[i] = item
            q3+=1

    except Exception as e:
        q4+=1
        print(q4,item,repr(e))
```

Data processing- Salary Rearrangement

- Print the result.

```
df[['salary','bottom','top','average']].head(99)
```

	salary	bottom	top	average
0	2001-4000	2001	4000	3000.5
1	8001-10000	8001	10000	9000.5
2	3500-4500	3500	4500	4000.0
3	6000-9000	6000	9000	7500.0
4	8001-10000	8001	10000	9000.5
5	20001-30000	20001	30000	25000.5
6	4001-6000	4001	6000	5000.5
7	6001-8000	6001	8000	7000.5
8	100001-150000	100001	150000	125000.5
9	15000-30000	15000	30000	22500.0
24	6001-8000	6001	8000	7000.5
25	10000-15000	10000	15000	12500.0
26	8000-15000	8000	15000	11500.0
27	面议	面议	面议	面议
28	8000-15000	8000	15000	11500.0
29	5000-8000	5000	8000	6500.0
...	...	...	...	...
69	4001-6000	4001	6000	5000.5
70	15000-30000	15000	30000	22500.0
71	15000-30000	15000	30000	22500.0
72	5000-8000	5000	8000	6500.0

Data processing- City Rearrangement

- Check the [place] df and the related data

```
df.place.value_counts()
```

```
北京          858
上海          525
深圳          324
广州          247
北京-朝阳区     141
杭州          126
成都          124
郑州           89
济南           73
武汉           71
南京           67
深圳-福田区      62
天津           57
苏州           56
厦门           51
青岛           48
广州-天河区      47
北京-海淀区      45
上海-浦东新区    44
西安           43
合肥           40
重庆           37
大连           34
福州           32
石家庄         31
长沙           30
深圳-南山区     29
南昌           28
北京-西城区     28
佛山           27
              ...
上海-金山区      1
平凉             1
双城             1
青岛-黄岛区（新行政区）    1
梧州             1
印度尼西亚       1
南通-港闸区      1
咸宁             1
无锡-南长区      1
晋中             1
邵阳             1
西安-莲湖区      1
盐城-亭湖区      1
成都-天府新区    1
宜宾             1
伊春             1
乌兰察布         1
淮安-清河区      1
安阳             1
松原             1
黄石             1
昆明-盘龙区      1
成都-郫都区      1
成都-龙泉驿区    1
烟台-芝罘区      1
兰州-七里河区    1

Name: place, Length: 404, dtype: int64
```

Data processing- City Rearrangement

- Add 【city】 column
- Compile regular expression pattern to return the pattern of an object
- index by line number. iloc(i) function means gain data of line i.
- re.search function match pattern in the string , and will return when finding the first matching string , if no string matches , it will return None

```
import re
df['city'] = df['place']
pattern2 = re.compile('(.*?)(\-)')
df_city=df['place'].copy()
for i in range(len(df_city)):
    item = df_city.iloc[i]. strip()
    result = re.search(pattern2, item)
    if result:
      print(result.group(1).strip())
      df_city.iloc[i] = result.group(1).strip()
  else:
      print(item.strip())
      df_city.iloc[i] = item.strip()
```

Data processing- City Rearrangement

- To examine if the new column is correct, we print both the new [city] column and the original [place] column to see if they match each other or not.

```
df['city'] = df_city
df[['city',"place"]]
```

	city	place
0	呼和浩特	呼和浩特
1	武汉	武汉
2	武汉	武汉
3	北京	北京
4	广州	广州
5	杭州	杭州
6	南京	南京
7	上海	上海
8	成都	成都
9	武汉	武汉-江岸区
10	上海	上海-普陀区
11	郑州	郑州
12	成都	成都-青羊区
13	郑州	郑州
14	北京	北京
15	天津	天津
16	广州	广州
17	广州	广州
18	广州	广州
19	广州	广州
20	上海	上海
21	成都	成都
22	成都	成都
23	福州	福州-台江区
24	成都	成都
25	深圳	深圳
26	上海	上海-浦东新区
27	深圳	深圳
28	南京	南京
29	重庆	重庆
...	...	...
4724	南通	南通
4725	深圳	深圳-南山区
4726	吉安	吉安
4727	北京	北京
4728	上海	上海
4729	深圳	深圳-福田区
4730	青岛	青岛
4731	吉林市	吉林市
4732	杭州	杭州
4733	上海	上海-闵行区
4734	北京	北京
4735	北京	北京
4736	天津	天津

4737 rows × 2 columns

Data Analysis & Visualization

Introduction

Apart from company names and job names, we pay attention to key words including average monthly salary, working city and related job descriptions.

We try to combine datum with jobs to answer questions like: how job opportunities concerning investment banks distributed among the country, whether most of the job recruitments concentrate in Beijing, Shanghai, Shenzhen and Guangzhou……

Data Analysis & Visualization

4.1 Job Distribution

4.2 Average monthly salary

Select top 15 cities and visualizing in 2 ways

Job Distribution

INTEGRATING & RANKING

Data Slicing

-To slice the previous [city] column and print the sorted numpy to get the data we want.

Visualization-Bar graph

-To use the plot statement to make the corresponding bar graph directly.

Visualization-Pie chart

-To vividly demonstrate the proportion of job opportunities concerning investment banks of each city.

Data Slicing

We **slice** the previous [city] column and **print** the sorted numpy to get the data we want

.

The city ends up to Xi'an, the last city of which the number of job opportunities is **above 50**.

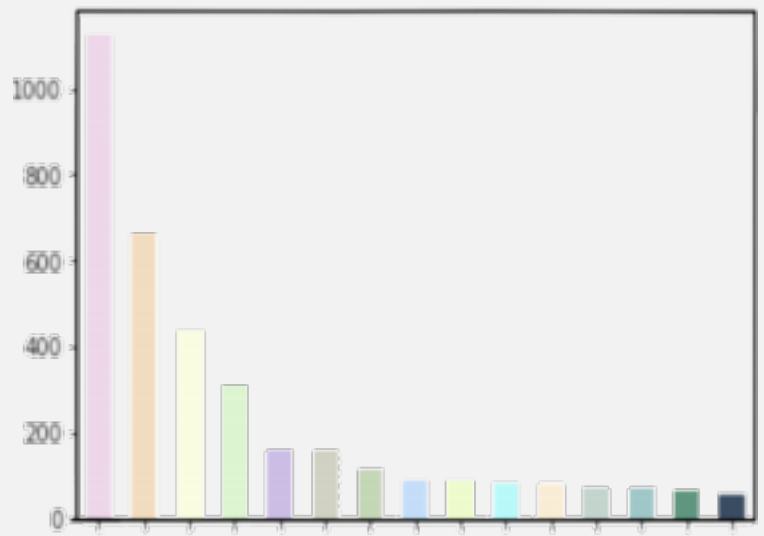
```
df['city'].value_counts()[:15].keys().tolist()
print(df['city'].value_counts()[:15].index.tolist())
print(df['city'].value_counts()[:15].values.tolist())
```

```
['北京', '上海', '深圳', '广州', '成都', '杭州', '郑州', '武汉', '南京', '济南'
, '青岛', '苏州', '天津', '厦门', '西安']
[1124, 662, 432, 311, 156, 154, 113, 87, 86, 79, 75, 72, 72, 66, 54]
```

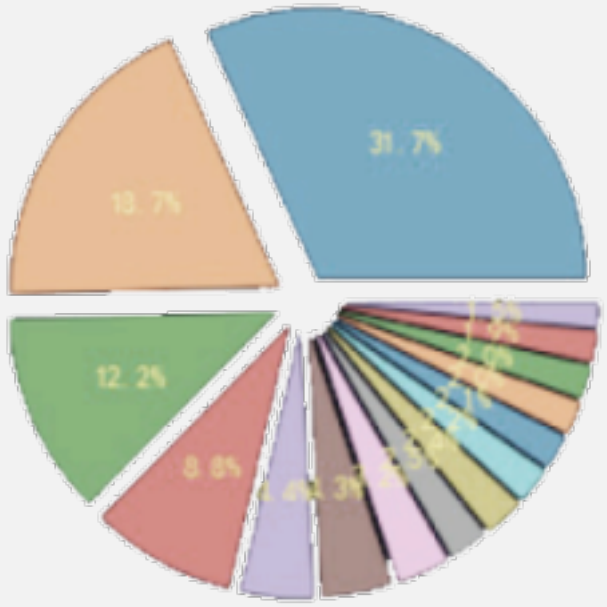
Visualization - Bar graph & Pie chart

BAR GRAPH

```
df['city'].value_counts()[:15].plot.bar()
```



PIE CHART



Data Analysis & Visualization

4.1 Job Distribution

4.2 Average monthly salary

Construct a DataFrame
and observe the corresponding salary

Average monthly salary

GROUPING
& ANALYZING

DataFrame constructing

-To construct a DataFrame including two columns [city] and [average salary], and process it with groupby operation.

Visualization-Histogram

-To set a canvas which is further provided with a subgraph, coordinating axis and its title, a general chart and some related illustrations in the form of histogram.

DataFrame constructing

Change the data from 'str' to 'float'.

```
df_average =
pd.to_numeric(df_average)
```

Divide the average salary into groups, according to the corresponding cities.

```
df4=pd.DataFrame
(data={'city':df_city,'average':df_average})

grouped4 =
df4['average'].groupby(df4['city'])
```

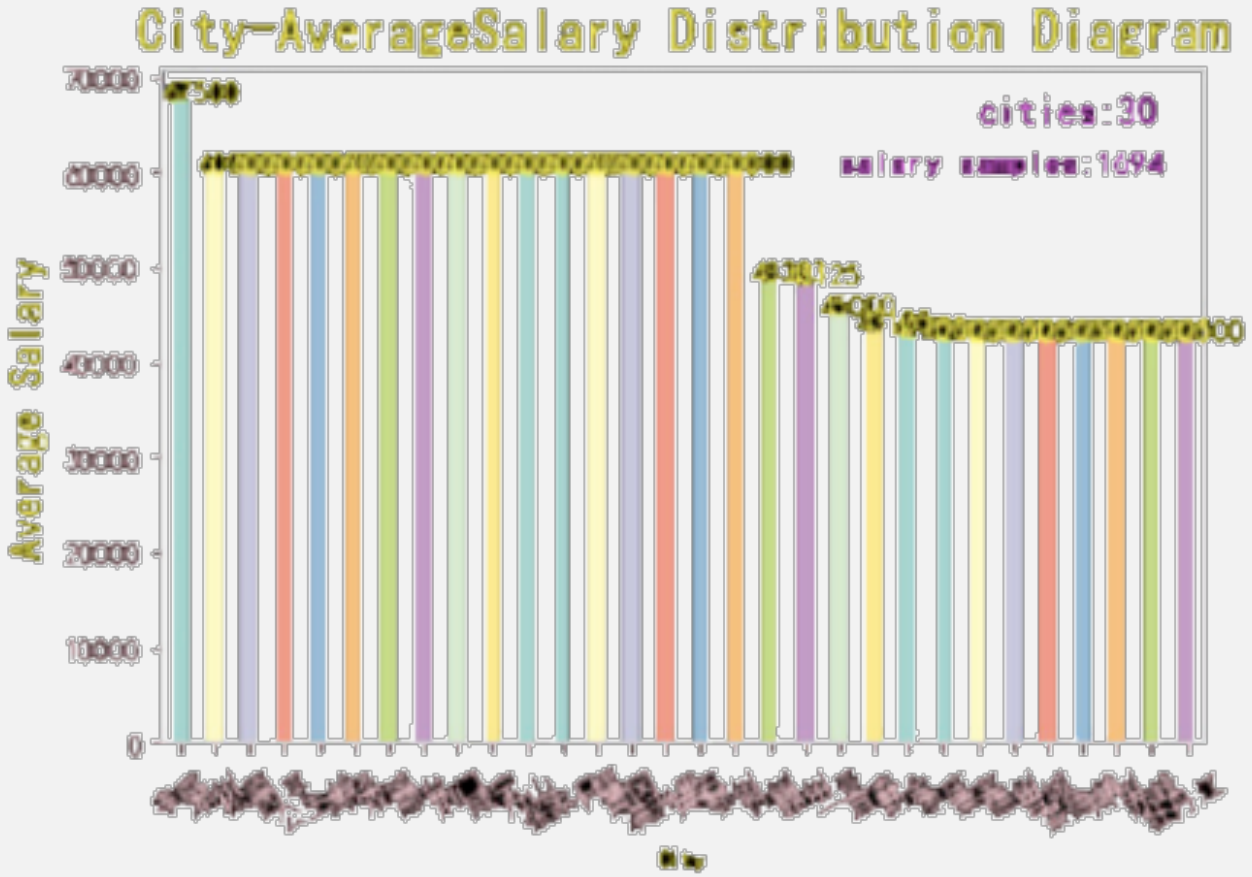
Sort the average salary in the ascending sort.

```
df_ranking=result4.sort_values
(ascending=False).round(1)[:30]
print(df_ranking)
```

Data Analysis & Visualization

Visualization - Histogram

Set the Canvas



Get Output

NO

Business is the human activity related to material things. It is necessary for civilization.

Analysis of Results

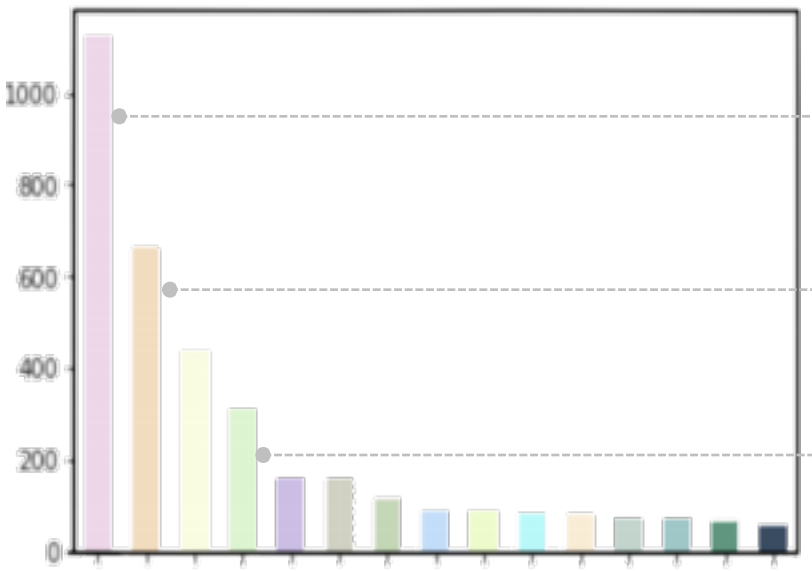
5.1 Job & Cities

5.2 Salary & Cities

What’ s the relationships between jobs and cities?

This part will analyze the results of the relationship between jobs and cities based on the visualization —— bar graph & pie chart

Analysis of Results

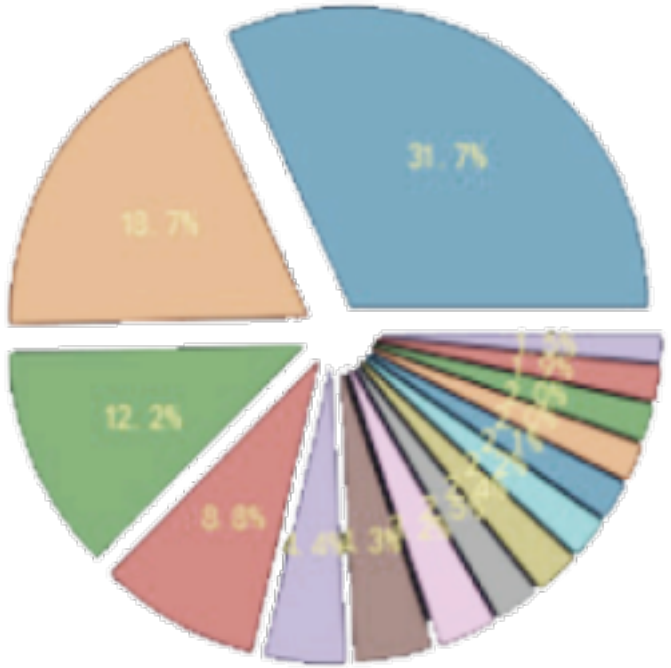


More opportunities !

Beijing tops the list of 1124

Shanghai follows, but roughly half

Beijing, Shanghai, Shenzhen Guangzhou, accounting for 71.4%



Analysis of Results

5.1 Job & Cities

5.2 Salary & Cities

Do more jobs mean higher salary?

This part will concentrate on analyzing the relationship between salary and cities based on the visualization —— histogram.

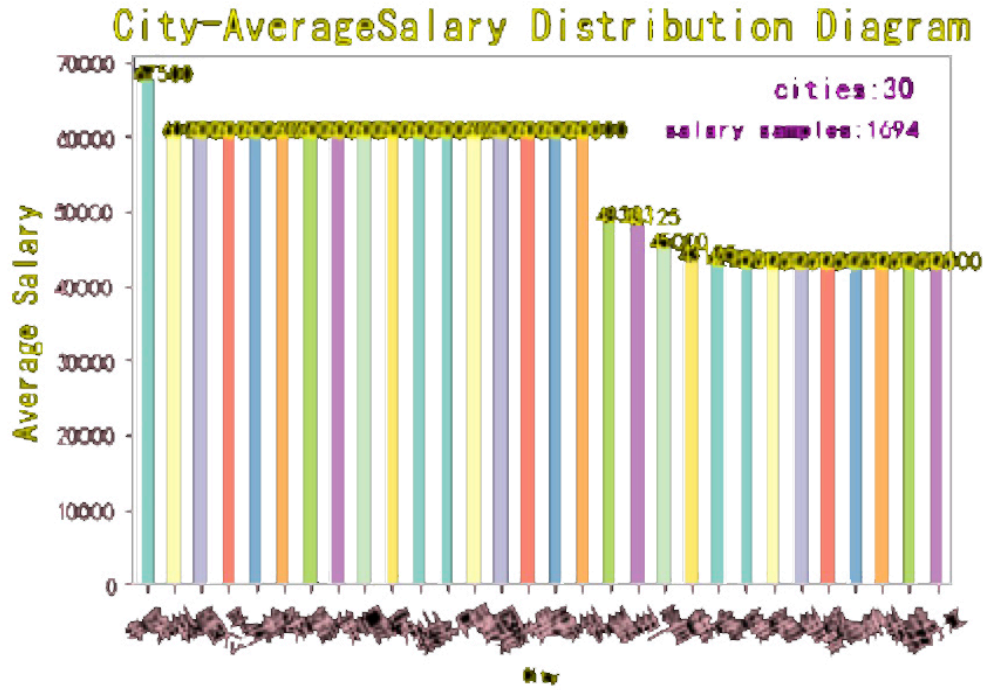
Analysis of Results

More jobs ≠ Higher salary

孝感	67500.2
松原	60000.5
四平	60000.5
公主岭	60000.5
菏泽	60000.5
双城	60000.5
邢台	60000.5
咸宁	60000.5
舟山	60000.5
张家口	60000.5
十堰	60000.5
黄冈	60000.5
湖州	60000.5
满洲里	60000.5
黄石	60000.5
鄂州	60000.5
赣州	48125.4
三亚	45000.2
宜春	43125.4
新余	42750.2
莱芜	42500.5
德州	42500.5
晋城	42500.5
景德镇	42500.5
朔州	42500.5
赤峰	42500.5
枣庄	42500.5
牡丹江	42500.5

dtype: float64

- Beijing, Shanghai, Guangzhou, Shenzhen fail to lead
- Xiaogan, Songyuan, Siping, Gongzhuling, Heze top the 5
- Limited data
- Suggestions
  I. Still relatively decent pay compared to other jobs
  II. Provide opportunity for those always seeking for jobs in first-line cities



Conclusion

Lasting three months, our project comes to an end on June 15th . We are glad to stand here to share our opinions with you, and we've learned a lot though the process of python learning.

Hope this presentation has given you some thoughts of ZPIN data analysis!

Problem Solving

With the help of python, the powerful and accessible programming language, we are able to solve problems more complex and practical than we could imagine.

Teamwork Building

Our teamwork skill and our organization and coordination capabilities are also improved during the project.

Thanks for Tutor

Thank our tutor, Mr. Bao, for giving us abundant, useful suggestions and support whenever we have problems. His profound knowledge sparks our interest in python.


THANK YOU
Quatray Kill