Data preprocessing

In [1]:
```
#Some preparation
import pandas as pd
import numpy as np
from data_preprocessing import *
from data_plot import *
from team_bulding import *

#Load the data
df = load_data('database\Skill Mapping (Responses).xlsx')

#Check the shape of data
print('Shape of the original data: ', df.shape)

#Remove the columns with missing values of data
df = remove_missing_value(df)
print('Shape of the preprocessed data: ', df.shape)
```

```
Shape of the original data:  (28, 71)
Shape of the preprocessed data:  (28, 71)
```

We should transfer the skill mapping from text result to a new numeric evaluation:

'Never heard of it': 1

'Willing to learn': 2

'Basic understanding': 3

'I can do this with very little guidance': 4

'Skillful': 5

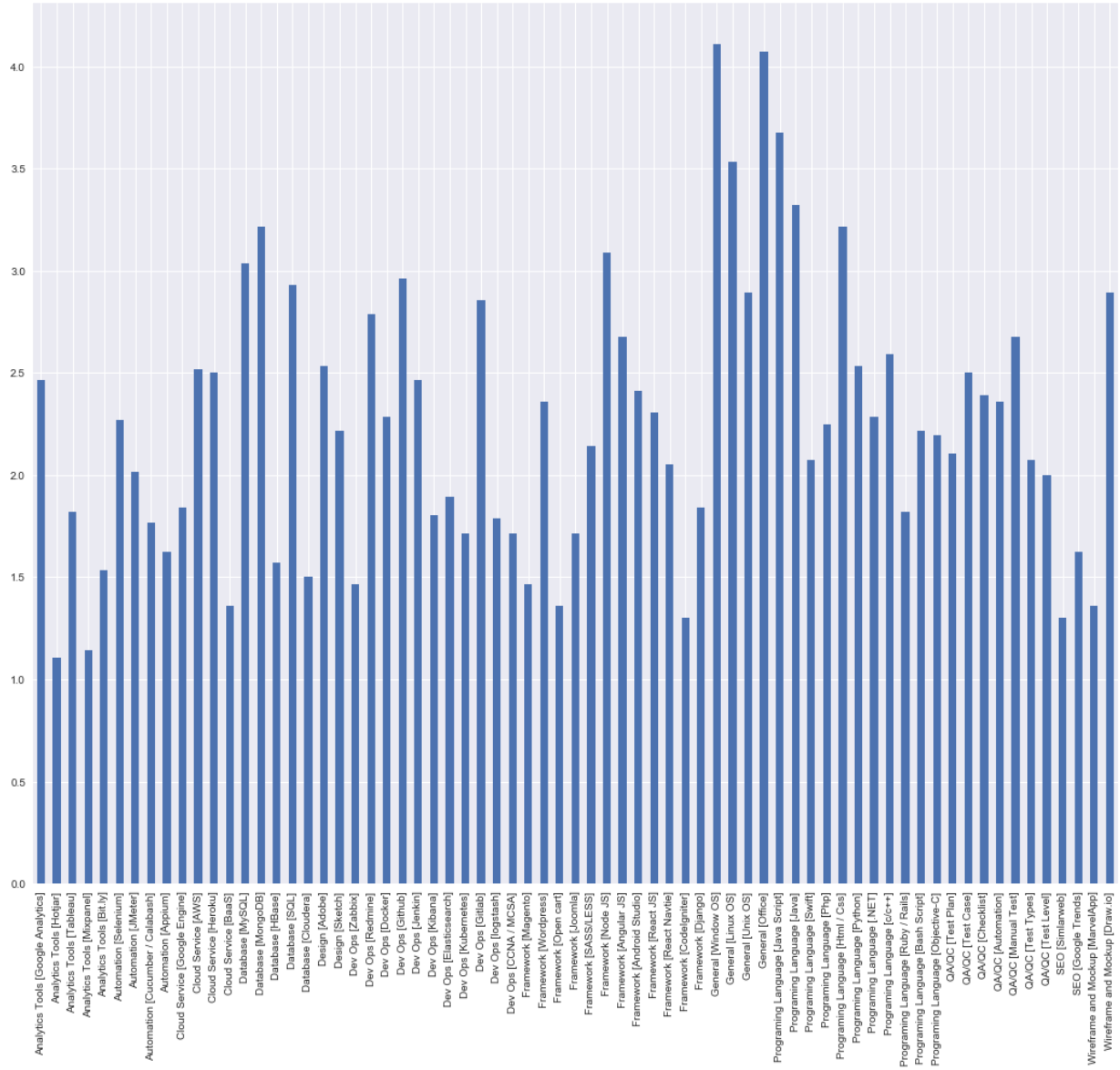
'Can teach other': 6

In [2]:
```
#Transform from categorical to ordinal features:
df = transform(df)
```

I. How would you visualize this data? Is there any insights you can derive from this data?

In [3]:
```
#Remove some redundant features in visualization step: Time stamp and the employe
df1 = df.drop(['Employee', 'Timestamp'], 1)
```

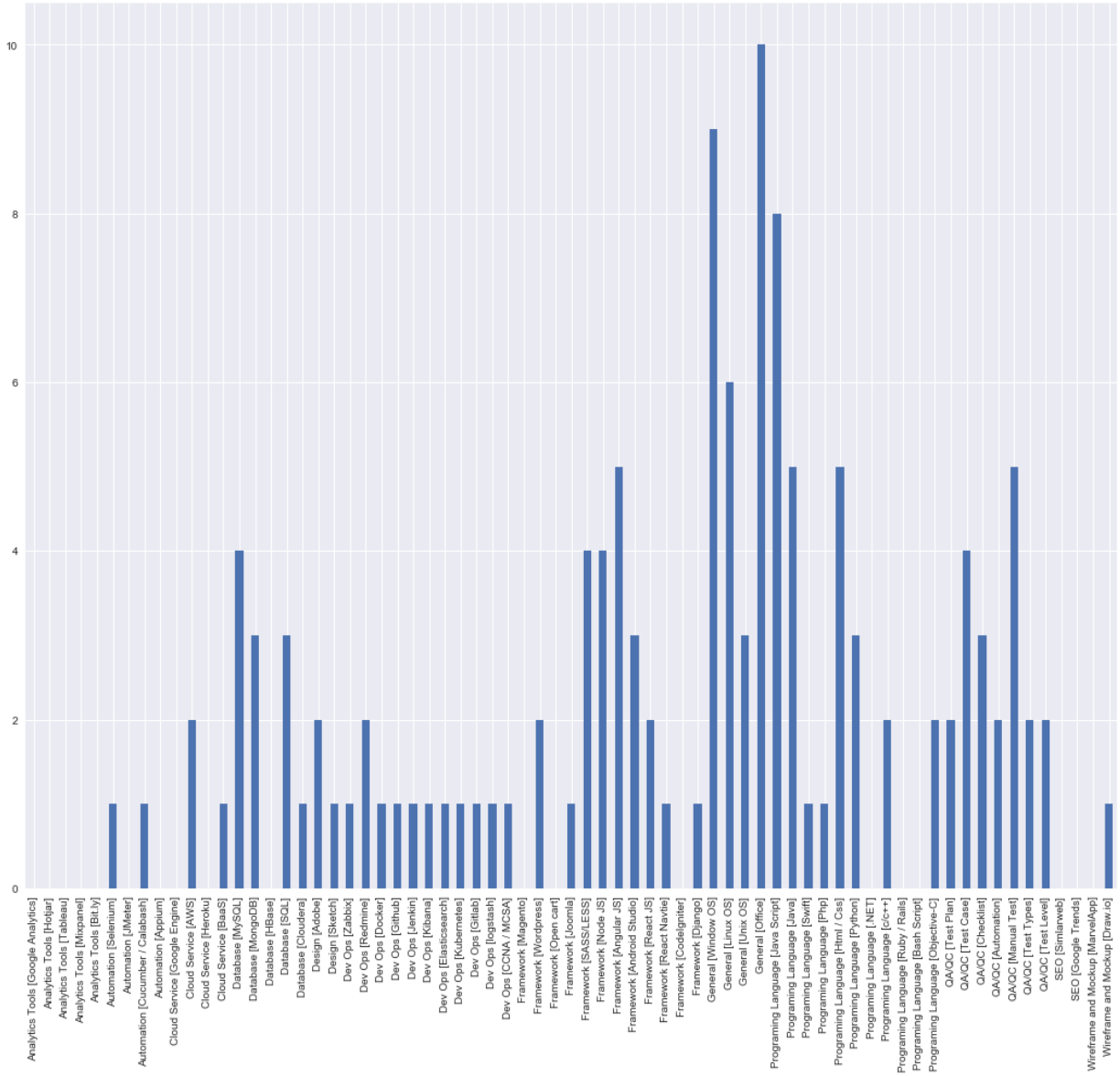
In [4]:
```
oned_plot(df1, criteria = 'mean', condition = None)
```




```
The first 10 skills with largest mean:
['General [Window OS]', 4.107142857142857]
['General [Office]', 4.071428571428571]
['Programing Language [Java Script]', 3.6785714285714284]
['General [Linux OS]', 3.5357142857142856]
['Programing Language [Java]', 3.3214285714285716]
['Database [MongoDB]', 3.2142857142857144]
['Programing Language [Html / Css]', 3.2142857142857144]
['Framework [Node JS]', 3.0892857142857144]
['Database [MySQL]', 3.0357142857142856]
```

=> We can see the first 10 strongest skills of the company.

In [5]:
```
oned_plot(df1, criteria = None, condition = '>4')
```




```
The first 10 skills with largest number of employee >4:
['General [Office]', 10]
['General [Window OS]', 9]
['Programing Language [Java Script]', 8]
['General [Linux OS]', 6]
['Framework [Angular JS]', 5]
['Programing Language [Java]', 5]
['Programing Language [Html / Css]', 5]
['QA/QC [Manual Test]', 5]
['Database [MySQL]', 4]
```

=>There are many highly skilled employee in this skill list. In the case the company wants to improve some employee's skills on this list, they can take advantage of these highly skilled employees for training, teaching each other and something like that.

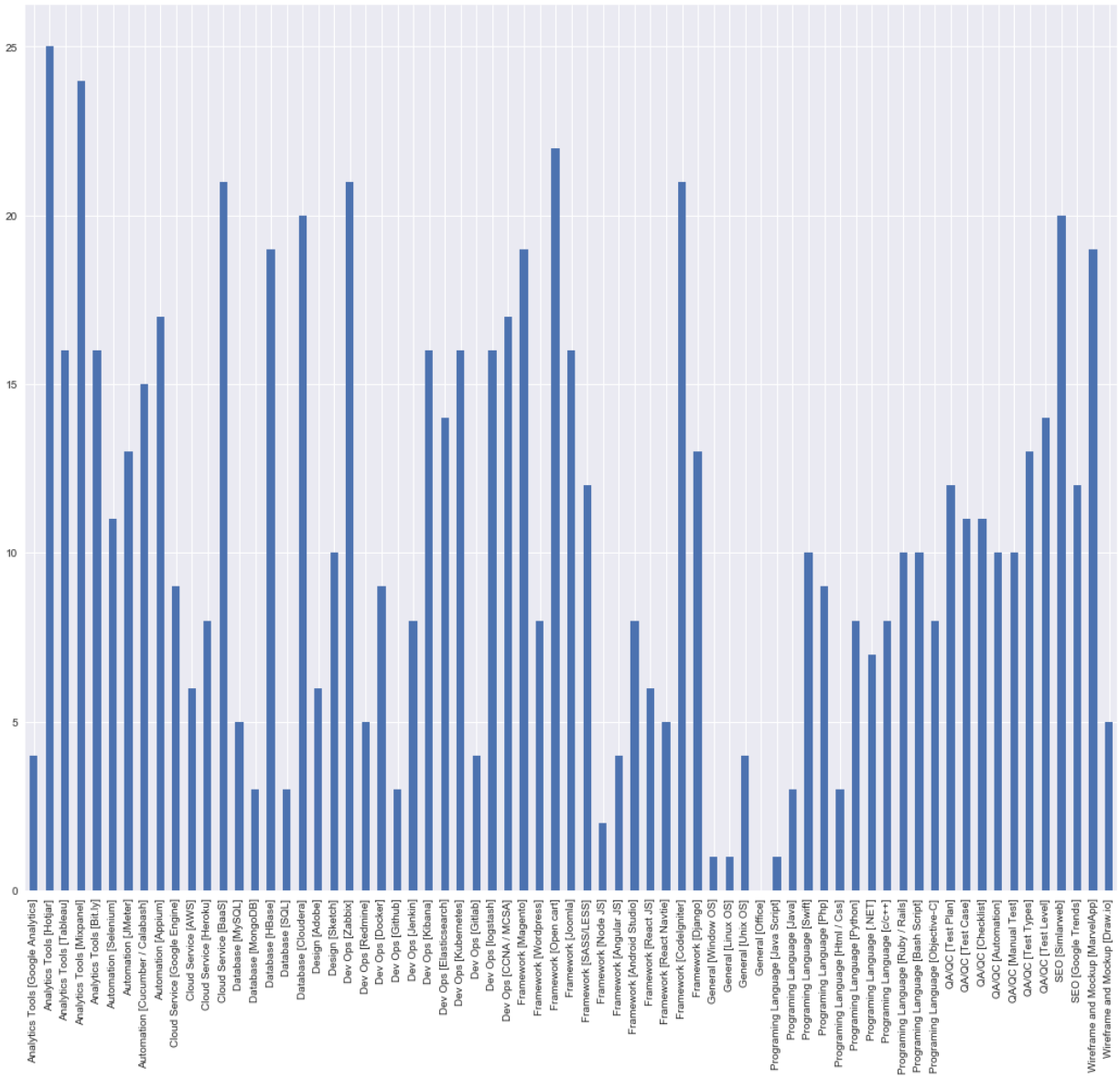
In [6]:
```
oned_plot(df1, criteria = 'mean', condition = '< 1.5')
```

```
The skills with mean < 1.5:
['Dev Ops [Zabbix]', 1.4642857142857142]
['Framework [Magento]', 1.4642857142857142]
['Cloud Service [BaaS]', 1.3571428571428572]
['Framework [Open cart]', 1.3571428571428572]
['Wireframe and Mockup [MarvelApp]', 1.3571428571428572]
['Framework [CodeIgniter]', 1.3035714285714286]
['SEO [Simlarweb]', 1.3035714285714286]
['Analytics Tools [Mixpanel]', 1.1428571428571428]
['Analytics Tools [Hotjar]', 1.1071428571428572]
```

=> These skills which your employees are very poor with mean eval < 1.5.

In [7]:
```
oned_plot(df1, criteria = None, condition = '<2')
```

```
<matplotlib.figure.Figure at 0x27853b2d6a0>
```




```
The first 10 skills with largest number of employee <2:
['Analytics Tools [Hotjar]', 25]
['Analytics Tools [Mixpanel]', 24]
['Framework [Open cart]', 22]
['Cloud Service [BaaS]', 21]
['Dev Ops [Zabbix]', 21]
['Framework [CodeIgniter]', 21]
['Database [Cloudera]', 20]
['SEO [Similarweb]', 20]
['Database [HBase]', 19]
```

=> Some skills, which are really new to your employee.

II. How would you structure and build a data framework so that each member can track their skill progress over time. Something like this, I can look at someone skillset from 6 months ago and compare to now.

Step 1: Collect their new skills after six months.

We can use google doc, excel form, or any text form to collect their new skill evaluation after six months. I won't go deeper into this section except for your need.

In this task, I assume that the employee number 3 will be the performance of employee number 2 after six months. Hence, I will try to compare employee number 2 and employee number 3.

In [8]:
```
before = df.iloc[2-1, ]
after = df.iloc[3-1, ]
```

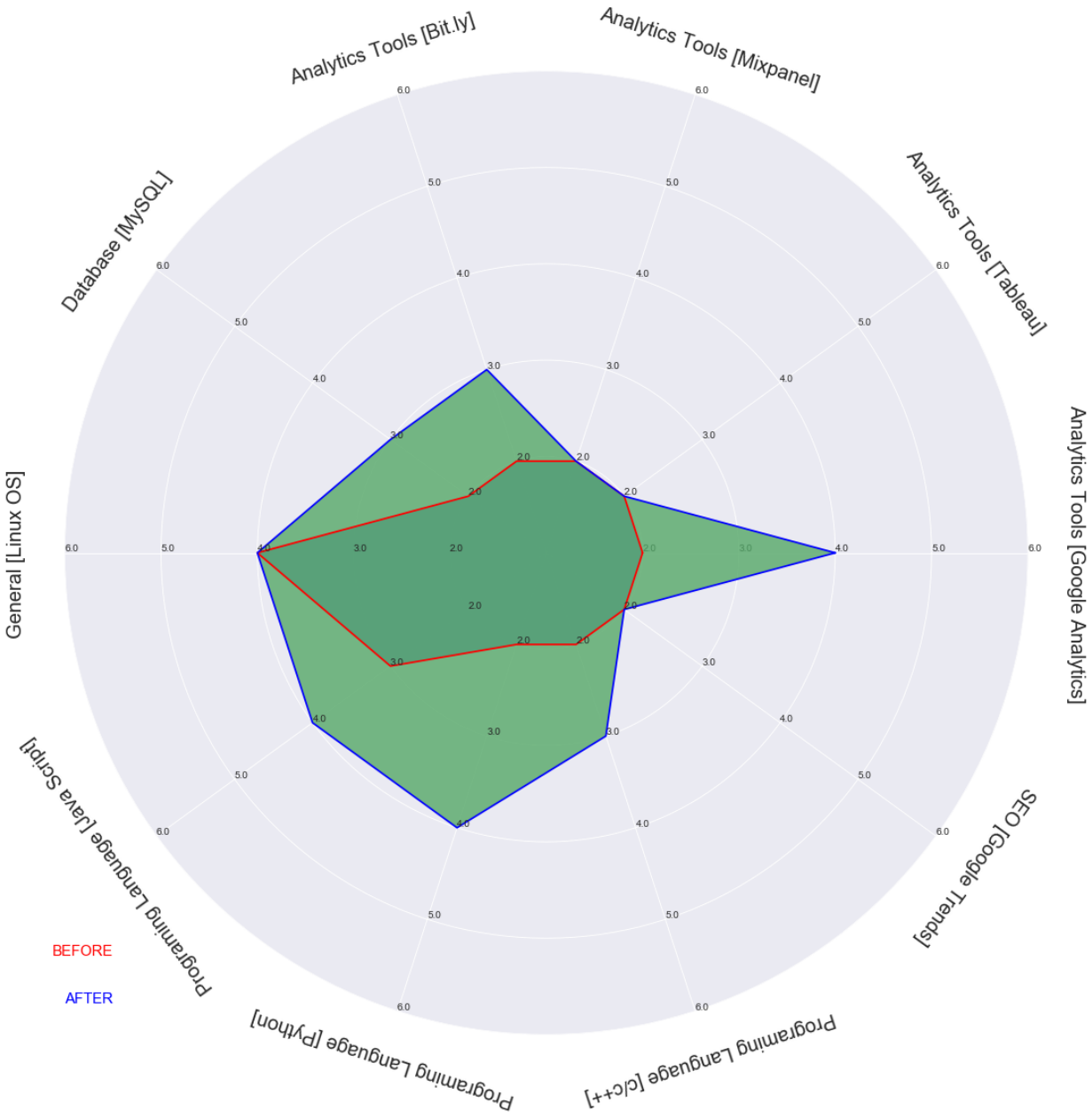
Step 2: Choose some set of skills you want to track.

These are more than 70 skills on the skill set, of course we don't want to track every skill. For example, I only want to check 10 skills: ['Analytics Tools [Google Analytics]', 'Analytics Tools [Tableau]', 'Analytics Tools [Mixpanel]', 'Analytics Tools [Bit.ly]', 'Database [MySQL]', 'General [Linux OS]', 'Programing Language [Java Script]', 'Programing Language [Python]', 'Programing Language [c/c++]', 'SEO [Google Trends]']

In [9]:
```
check_skills = ['Analytics Tools [Google Analytics]',
 'Analytics Tools [Tableau]',
 'Analytics Tools [Mixpanel]',
 'Analytics Tools [Bit.ly]',
 'Database [MySQL]',
 'General [Linux OS]',
 'Programing Language [Java Script]',
 'Programing Language [Python]',
 'Programing Language [c/c++]',
 'SEO [Google Trends]']
```

Step 3: Plot to compare and conclusion.

In [10]:
```
rada_plot(before, after, variables = check_skills)
```




```
Improvement in each individual skills:
Analytics Tools [Google Analytics] : 100.0 % -> 2 step.
Analytics Tools [Tableau] : 0.0 % -> 0 step.
Analytics Tools [Mixpanel] : 0.0 % -> 0 step.
Analytics Tools [Bit.ly] : 50.0 % -> 1 step.
Database [MySQL] : 50.0 % -> 1 step.
General [Linux OS] : 0.0 % -> 0 step.
Programing Language [Java Script] : 33.33 % -> 1 step.
Programing Language [Python] : 100.0 % -> 2 step.
Programing Language [c/c++] : 50.0 % -> 1 step.
SEO [Google Trends] : 0.0 % -> 0 step.

Improvement in total:  34.78 %
```

III. Based on this dataset, what kind of machine

learning algorithim would you use to pick the best team of 3-5 people for a certain project if you know the technology stack requirement (Project A needs Python, Java, Data Framework and Tableau for example)?

=> Based on this dataset, I will choose the k-nearest neighbor algorithm (k = 3:5) to pick the best team for some technology stack requirement. Its main idea is picking up 3-5 people which their skill match the technology stack requirement most by applying the similarity function.

Step 1: Match the technology stack requirement to the skill dataset.

Assume the the technology stack requirements are: ['Python', 'Java Script', 'Linux OS', 'Tableau', 'MySQL', 'Google Trends','Mixpanel', 'Bit.ly', 'Google Analytics', 'c/c++']. We need to find some skills which are matched to these requirement in the skill dataseet.

In [11]:
```
dataset_skills = list(df1.columns)
project_skills = ['Python', 'Java Script', 'Linux OS', 'Tableau', 'MySQL',
                  'Google Trends','Mixpanel', 'Bit.ly', 'Google Analytics', 'c/c+

matched_skills = matching_skills(project_skills, dataset_skills)
```

Besides that, I suggest a capacity requirement set for each skill. For example, with above technology stack requirement I require some details:

'Python': Skillful (as 5.0)

'Java Script': Willing to learn (as 2.0)

'Linux OS': Willing to learn (as 2.0)

...

In [12]:
```
capacity_requirements = np.array([2.0, 2.0, 2.0, 5.0, 3.0, 2.0, 2.0, 2.0, 5.0, 3.(
```

Step 2: Calculate the similarity function between capacity_requirements and all the employees.

In [13]:
```
employees_skills = [[i + 1, np.array(df[matched_skills].iloc[i, :])] for i in df.
evaluated_result = similarity_func(capacity_requirements, employees_skills)
```

Step 3: Choose the best of k employees for the project

In [14]:
```
k = 5
result_list = best_of_k_list(evaluated_result, k)
print('capacity_requirements: ', capacity_requirements)
print('\nList of available employees: ')
for i in result_list:
    print('Employee number ', i[0], 'Capacity skills:', i[1], 'Matched point:', i
```

```
capacity_requirements:  [ 2.  2.  2.  5.  3.  2.  2.  2.  5.  3.]

List of available employees:
Employee number  17 Capacity skills: [ 3.  4.  1.  2.  5.  5.  5.  5.  3.  3.]
 Matched point: 31.0
Employee number  22 Capacity skills: [ 4.  4.  1.  1.  5.  5.  5.  5.  4.  2.]
 Matched point: 30.0
Employee number  3 Capacity skills: [ 4.  2.  2.  3.  3.  4.  4.  4.  3.  2.] M
atched point: 21.0
Employee number  28 Capacity skills: [ 3.  4.  2.  2.  4.  4.  2.  4.  4.  1.]
 Matched point: 19.0
Employee number  27 Capacity skills: [ 2.  3.  1.  2.  3.  3.  3.  3.  6.  1.]
 Matched point: 18.0
```

Some suggestions and conclusion:

1. We can improve the algorithm performance by adding the features: 'Ready for new project'. It will help us to modify who is now available for a new project, who is busy right now.
2. Also, we need to add the features: 'Job position'. Without it, we can add 5 project managers into a team :)
3. If you have to choose 5 people for the project, let use this algorithm for choosing 10 best people. After that, we can apply a more complicated ML algorithm or personal decision to choose the best suitable 5 people. Sometimes, the best is not actually the best!