Use pip install $package_name to install package

# NUMPY

## Creating an Array:

```
In [11]: import numpy as np
```

```
In [282]: np.arange(0,10)
          np.arange(0,20,2)
          np.linspace(0,10,3)
          np.ones((5,5))
          np.zeros((4,4))
          np.eye(5)
          y = np.diag(x)
          y[0:9:2,0:9:3] #slicing in matrix

Out[282]: dtype('int32')
```

```
In [12]: np.diag(x)
         np.full((3,3),5)

Out[12]: array([[5, 5, 5],
                [5, 5, 5],
                [5, 5, 5]])
```

```
In [16]: np.random.randint(0,20,5) #random numbers between the range

Out[16]: array([10, 16,  4,  9, 19])
```

```
In [277]: a=np.arange(0,20)
          a.resize(4,5)
          a

Out[277]: array([[ 0,  1,  2,  3,  4],
                 [ 5,  6,  7,  8,  9],
                 [10, 11, 12, 13, 14],
                 [15, 16, 17, 18, 19]])
```

```
In [38]: b=np.arange(30)
         b.reshape(5,6)

Out[38]: 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])
```

## Operations on Array:

In [283]:
```
len(x)
x.size
x.ndim
x.dtype
x.shape
# x.astype(str)
```

Out[283]:
```
dtype('int32')
```

In [285]:
```
p =np.random.randint(0,40,15)
p=p.reshape(5,3)
np.sum(p , axis=1)
np.mean(p , axis = 0) #np.mean(p)
np.median(p)
np.cumsum(p,axis=0)
np.std(p)
```

Out[285]:
```
12.732111634236745
```

In [289]:
```
q=np.random.randint(40,80,15)
q =q.reshape(3,5)
```

## Operation on 2 arrays (a,b)

In [59]:
```
# np.add(p,q)
# np.subtract(p,q)
# np.multiply(p,q)
# np.divide(p,q)
r =np.dot(p,q) #dimension should be aligned
r.shape
```

Out[59]:
```
(5, 5)
```

In [290]:
```
q = q.reshape(5,3)
np.vstack((p,q))
np.hstack((p,q))
```

Out[290]:
```
array([[25,  3, 29, 78, 56, 58],
       [16,  2, 28, 71, 50, 43],
       [ 0,  1,  7, 54, 75, 63],
       [16, 34, 18, 61, 48, 69],
       [16,  0, 39, 66, 66, 76]])
```

In [294]:
```
z=np.ones((4,6))
np.vsplit(z,2) #axis = 1
np.hsplit(z,2) #axis = 0
```

Out[294]:
```
[array([[1., 1., 1.],
        [1., 1., 1.],
        [1., 1., 1.],
        [1., 1., 1.]]), array([[1., 1., 1.],
        [1., 1., 1.],
        [1., 1., 1.],
        [1., 1., 1.]])]
```

# PANDAS

In [2]:
```
import pandas as pd
```

## Create DataFrame:

In [3]:
```
cols = ['preg_count','glucose','BP','skin_thick','insulin','BMI','pedigree','a
ge','class','names']
df = pd.read_excel(r"C:\Users\admin\Desktop\Pandas_sample_data - Copy.xlsx")
df.head()
```

Out[3]:

| | preg_count | glucose | BP | skin_thick | insulin | BMI | pedigree | age | class | names |
|---|---|---|---|---|---|---|---|---|---|---|
| **1** | 6.0 | 148.0 | 70.0 | 20.0 | 0.0 | 28.2 | 0.526 | 38 | 1 | NaN |
| **2** | NaN | 85.0 | 71.0 | 21.0 | 0.0 | NaN | 1.526 | 39 | 1 | NaN |
| **3** | 8.0 | 69.0 | 72.0 | 22.0 | 0.0 | 30.2 | 2.526 | 150 | 0 | NaN |
| **4** | 1.0 | 156.0 | 73.0 | 23.0 | 0.0 | 31.2 | 3.526 | 41 | 1 | NaN |
| **5** | 7.0 | 123.0 | 74.0 | 24.0 | 45.0 | 32.2 | 4.526 | 42 | 1 | NaN |

In [11]:
```
dict_={'names':['  A   ','    B','C','D','E   ','F'],'Age':[15,12,16,14,20,30
]}
dict_=pd.DataFrame(dict_)
dict_['names']
```

Out[11]:
```
0      A   
1          B
2          C
3          D
4      E   
5          F
Name: names, dtype: object
```

In [13]:
```
dict_['names'][:3] #accessing any data by slicing
dict_['names'].str.lower()
dict_['names']=dict_['names'].str.strip()
dict_.head(6)
```

Out[13]:

| | names | Age |
|---|---|---|
| **0** | A | 15 |
| **1** | B | 12 |
| **2** | C | 16 |
| **3** | D | 14 |
| **4** | E | 20 |
| **5** | F | 30 |

## Data Exploration:

In [39]:
```
dict_.describe()
dict_.count()
# dict_.info()
# dict_.max() # min
# dict_.median() # mean,mode
# dict_.shape
```

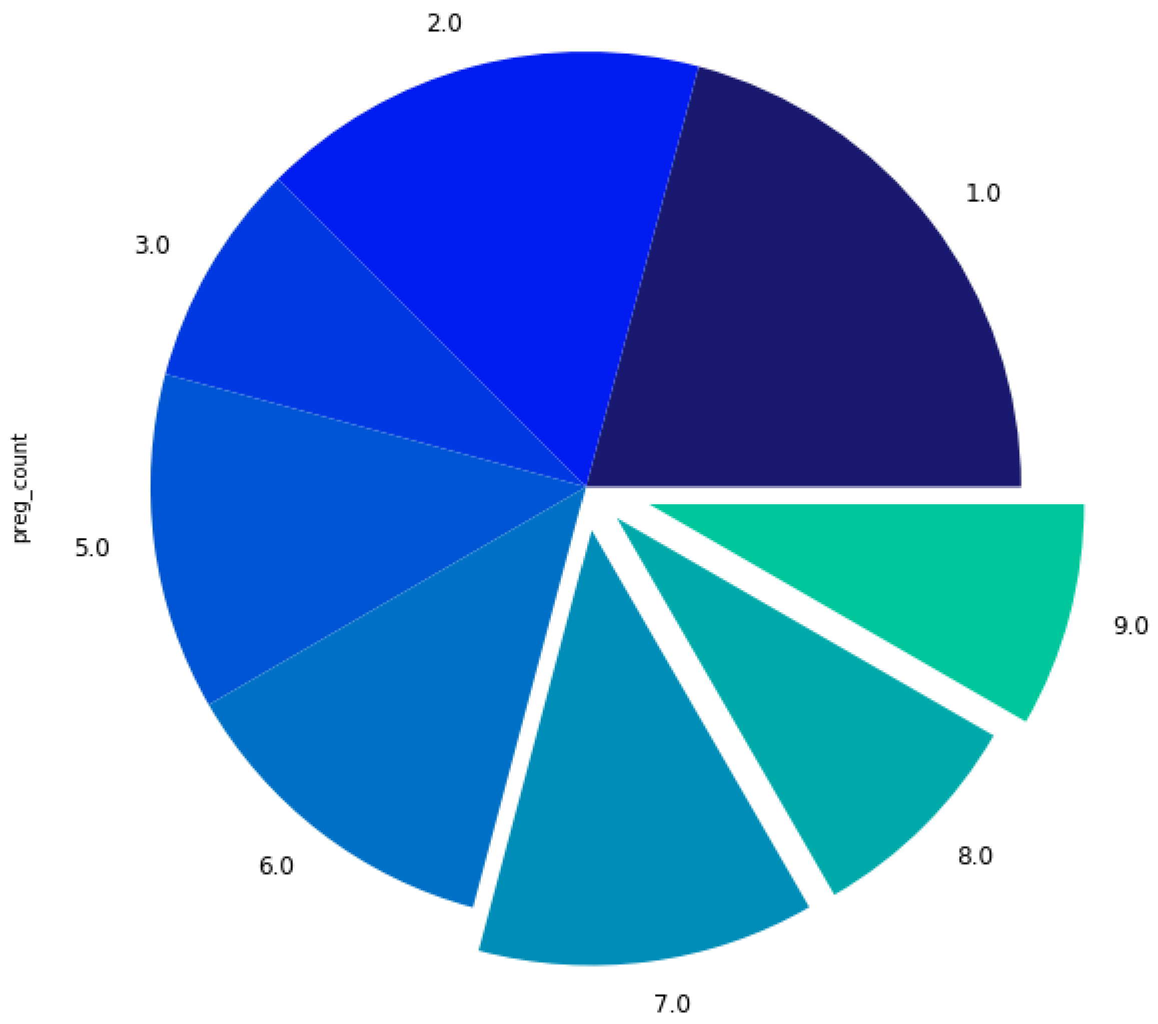
```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-39-da712089bfca> in <module>
----> 1 dict_.describe()
      2 dict_.count()
      3 # dict_.info()
      4 # dict_.max() # min
      5 # dict_.median() # mean,mode

NameError: name 'dict_' is not defined
```

In [25]:
```
df['BMI'].mean() #mean for specific column
df.describe()
```

Out[25]:
```
43.16551724137933
```

In [41]:
```
%matplotlib inline
df['preg_count'].value_counts().sort_index()
colors = ['#191970', '#001CF0', '#0038E2', '#0055D4', '#0071C6', '#008DB8', '#00AAAA','#00C69C']
explode = [0, 0, 0, 0, 0, 0.1, 0.1, 0.15]
dff = df['preg_count'].value_counts().sort_index()
dff.plot(kind='pie', colors=colors, explode=explode, fontsize=12,figsize=(10, 10))
```

Out[41]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x937b87b00>
```



## Data cleaning:

In [34]:
```
cols = ['preg_count','glucose','BP','skin_thick','insulin','BMI','pedigree','a
ge','class','names']
df = pd.read_excel(r"C:\Users\admin\Desktop\Pandas_sample_data - Copy.xlsx",co
lumns = cols)
df.head()
df.iloc[1:10] #uses index of the dataframe in both rows and columns
df.loc[1:10] #uses location label of the dataframe in both rows and column nam
es in columns
```

Out[34]:

| | preg_count | glucose | BP | skin_thick | insulin | BMI | pedigree | age | class | names |
|---|---|---|---|---|---|---|---|---|---|---|
| **1** | 6.0 | 148.0 | 70.0 | 20.0 | 0.0 | 28.2 | 0.526 | 38 | 1 | NaN |
| **2** | NaN | 85.0 | 71.0 | 21.0 | 0.0 | NaN | 1.526 | 39 | 1 | NaN |
| **3** | 8.0 | 1000.0 | 72.0 | 22.0 | 0.0 | 30.2 | 2.526 | 150 | 0 | NaN |
| **4** | 1.0 | 156.0 | 73.0 | 23.0 | 0.0 | 31.2 | 3.526 | 41 | 1 | NaN |
| **5** | 7.0 | 123.0 | 74.0 | 24.0 | 45.0 | 32.2 | 4.526 | 42 | 1 | NaN |
| **6** | 5.0 | 78.0 | 75.0 | 25.0 | 15.0 | 33.2 | 5.526 | 43 | 0 | NaN |
| **7** | NaN | 159.0 | 76.0 | 26.0 | 26.0 | 34.2 | 6.526 | 44 | 1 | NaN |
| **8** | NaN | 162.0 | 77.0 | 27.0 | 124.0 | 35.2 | 7.526 | 45 | 1 | NaN |
| **9** | 1.0 | 87.0 | 78.0 | 28.0 | 3.0 | 36.2 | 8.526 | 46 | 0 | NaN |
| **10** | 2.0 | 96.0 | 79.0 | NaN | 4.0 | 37.2 | 9.526 | 47 | 1 | NaN |

In [9]:
```
df.count()
pd.isna(df).head()
df.dropna(how='any',axis=1) #how='any'/'all' and axis=0/1, drops when at least
one value is nan
df.dropna(how='all',axis=1) #drops when all values of column are nan
df[df['preg_count'].isna()].index #returns the index of all null values

df[df['preg_count'].isna()].index
```

Out[9]:
```
Int64Index([2, 7, 8, 16, 22, 26], dtype='int64')
```

In [164]:
```
df['preg_count'].fillna(value=5)
df['preg_count'].fillna(df['preg_count'].mean())
```

In [26]:
```
df['age'].loc[df['age']>100] = np.nan
df=df.fillna(df.mean())
df.loc[:, df.columns != 'names']
df.loc[0:10,['BP','skin_thick']]
df.head()
df.tail()
df.iloc[0:10,:-1] #'all columns except names'
```

Out[26]:

| | preg_count | glucose | BP | skin_thick | insulin | BMI | pedigree | age | class |
|---|---|---|---|---|---|---|---|---|---|
| **1** | 6.000000 | 148.0 | 70.0 | 20.000000 | 0.0 | 28.200000 | 0.526 | 38.000000 | 1 |
| **2** | 4.458333 | 85.0 | 71.0 | 21.000000 | 0.0 | 43.165517 | 1.526 | 39.000000 | 1 |
| **3** | 8.000000 | 69.0 | 72.0 | 22.000000 | 0.0 | 30.200000 | 2.526 | 52.931034 | 0 |
| **4** | 1.000000 | 156.0 | 73.0 | 23.000000 | 0.0 | 31.200000 | 3.526 | 41.000000 | 1 |
| **5** | 7.000000 | 123.0 | 74.0 | 24.000000 | 45.0 | 32.200000 | 4.526 | 42.000000 | 1 |
| **6** | 5.000000 | 78.0 | 75.0 | 25.000000 | 15.0 | 33.200000 | 5.526 | 43.000000 | 0 |
| **7** | 4.458333 | 159.0 | 76.0 | 26.000000 | 26.0 | 34.200000 | 6.526 | 44.000000 | 1 |
| **8** | 4.458333 | 162.0 | 77.0 | 27.000000 | 124.0 | 35.200000 | 7.526 | 45.000000 | 1 |
| **9** | 1.000000 | 87.0 | 78.0 | 28.000000 | 3.0 | 36.200000 | 8.526 | 46.000000 | 0 |
| **10** | 2.000000 | 96.0 | 79.0 | 33.884615 | 4.0 | 37.200000 | 9.526 | 47.000000 | 1 |

In [28]:
```
df['glucose']=df['glucose'].replace(1000,80) #replacing values in columns
df.head()
```

Out[28]:

| | preg_count | glucose | BP | skin_thick | insulin | BMI | pedigree | age | class | names |
|---|---|---|---|---|---|---|---|---|---|---|
| **1** | 6.0 | 148.0 | 70.0 | 20.0 | 0.0 | 28.2 | 0.526 | 38 | 1 | NaN |
| **2** | NaN | 85.0 | 71.0 | 21.0 | 0.0 | NaN | 1.526 | 39 | 1 | NaN |
| **3** | 8.0 | 80.0 | 72.0 | 22.0 | 0.0 | 30.2 | 2.526 | 150 | 0 | NaN |
| **4** | 1.0 | 156.0 | 73.0 | 23.0 | 0.0 | 31.2 | 3.526 | 41 | 1 | NaN |
| **5** | 7.0 | 123.0 | 74.0 | 24.0 | 45.0 | 32.2 | 4.526 | 42 | 1 | NaN |

In [35]:
```
#Applying a function on a column
def div_100(x):
    return x/100
df['glucose'].apply(div_100).head()
```

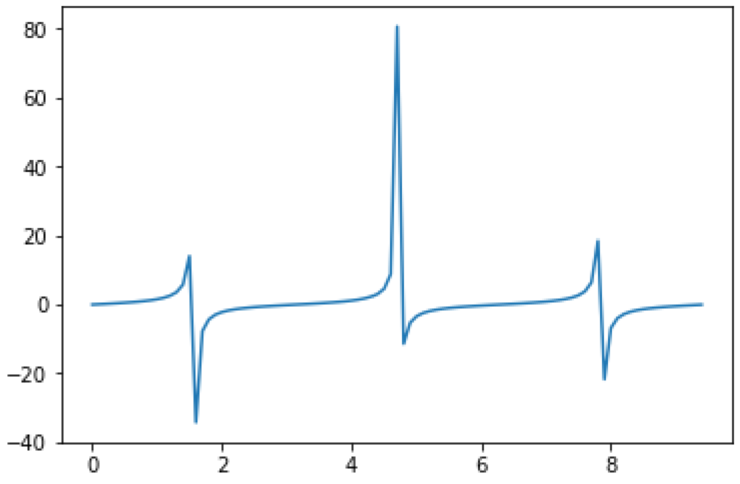
Out[35]:
```
1     1.48
2     0.85
3    10.00
4     1.56
5     1.23
Name: glucose, dtype: float64
```

## Saving DataFrame after processing:

In [237]:
```
dict_.to_excel(r"C:\Users\admin\Desktop\dict_.xlsx")
```

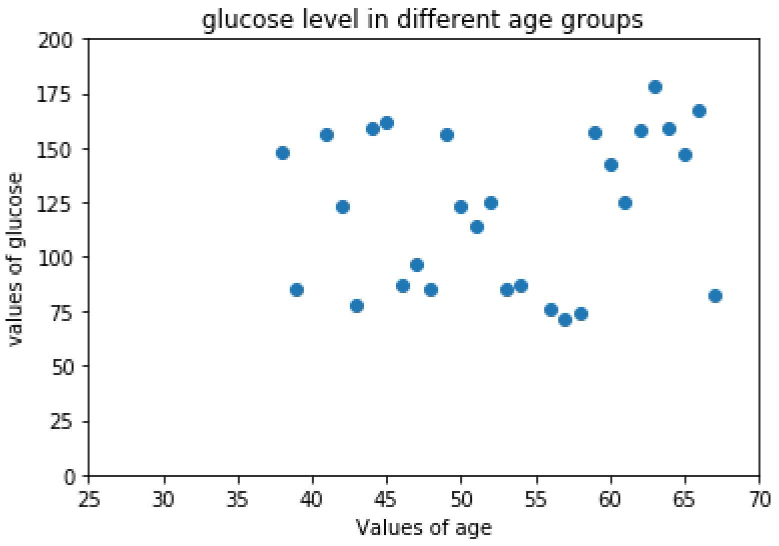
## Matplotlib

In [42]:
```
import matplotlib.pyplot as plt
x = np.arange(0,3*np.pi,0.1)
y = np.tan(x) #sin, cos, tan
plt.plot(x,y)
plt.show()
```



In [261]:
```
import matplotlib.pyplot as plt
x1 = df.iloc[:,7:8]
y1 = df.iloc[:,1:2]
plt.xlim(25,70)
plt.ylim(0,200)
plt.scatter(x1,y1)
plt.xlabel('Values of age')
plt.ylabel('values of glucose')
plt.title('glucose level in different age groups')
```

Out[261]:
```
Text(0.5, 1.0, 'glucose level in different age groups')
```



In [ ]: