In [8]:

```
import pandas as pd
import numpy as np
from sklearn import linear_model
import matplotlib.pyplot as plt
from word2number import w2n
```

In [86]:

```
df = pd.read_csv("https://raw.githubusercontent.com/codebasics/py/master/ML/1_linear_reg/Exercise/c
df.head()
```

Out[86]:

	year	per capita income (US$)
0	1970	3399.299037
1	1971	3768.297935
2	1972	4251.175484
3	1973	4804.463248
4	1974	5576.514583

In [48]:

```
df.rename({'per capita income (US$)':'income'},axis=1,inplace=True)
df.head(1)
```

Out[48]:

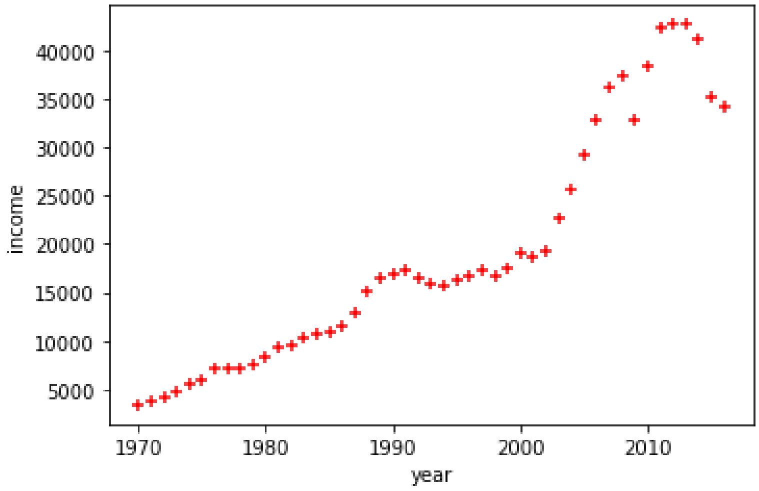
	year	income
0	1970	3399.299037

In [49]:

```
%matplotlib inline
plt.xlabel('year')
plt.ylabel('income')
plt.scatter(df.year,df.income ,color='red',marker='+')
```

Out[49]:

```
<matplotlib.collections.PathCollection at 0x15c3a83e190>
```

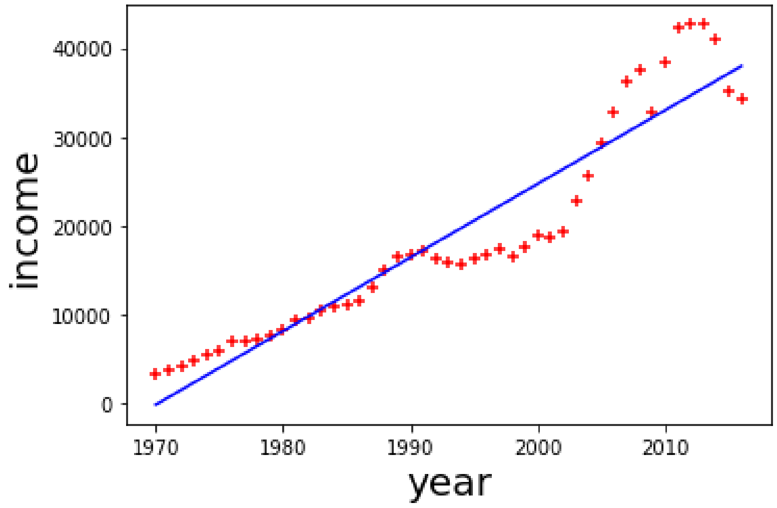


In [82]:

```
%matplotlib inline
plt.xlabel('year',fontsize = 20)
plt.ylabel('income',fontsize = 20)
plt.scatter(df.year,df.income ,color='red',marker='+')
plt.plot(df.year,reg.predict(df[['year']]),color='blue')
```

Out[82]:

```
[<matplotlib.lines.Line2D at 0x15c3a9e47c0>]
```



In [72]:

```python
new_df = df.drop('income',axis='columns')
new_df.head()
```

Out[72]:

	year
0	1970
1	1971
2	1972
3	1973
4	1974

In [71]:

```python
income = df.income
income.head()
```

Out[71]:

```
0    3399.299037
1    3768.297935
2    4251.175484
3    4804.463248
4    5576.514583
Name: income, dtype: float64
```

In [73]:

```python
# Create linear regression object
reg = linear_model.LinearRegression()
reg.fit(new_df,income)
```

Out[73]:

```
LinearRegression()
```

In [74]:

```python
reg.predict([[2020]])
```

Out[74]:

```
array([41288.69409442])
```

2nd question

In [21]:

```python
import pandas as pd
import numpy as np
from sklearn import linear_model
from word2number import w2n
```

In [22]:

```python
df = pd.read_csv("https://raw.githubusercontent.com/codebasics/py/master/ML/2_linear_reg_multivaria
df
```

Out[22]:

	experience	test_score(out of 10)	interview_score(out of 10)	salary($)
0	NaN	8.0	9	50000
1	NaN	8.0	6	45000

	experience	test_score(out of 10)	interview_score(out of 10)	salary($)
2	five	6.0	7	60000
3	two	10.0	10	65000
4	seven	9.0	6	70000
5	three	7.0	10	62000
6	ten	NaN	7	72000
7	eleven	7.0	8	80000

In [23]:

```
df.experience = df.experience.fillna("zero")
df
```

Out[23]:

	experience	test_score(out of 10)	interview_score(out of 10)	salary($)
0	zero	8.0	9	50000
1	zero	8.0	6	45000
2	five	6.0	7	60000
3	two	10.0	10	65000
4	seven	9.0	6	70000
5	three	7.0	10	62000
6	ten	NaN	7	72000
7	eleven	7.0	8	80000

In [24]:

```
df.experience = df.experience.apply(w2n.word_to_num)
df
```

Out[24]:

	experience	test_score(out of 10)	interview_score(out of 10)	salary($)
0	0	8.0	9	50000
1	0	8.0	6	45000
2	5	6.0	7	60000
3	2	10.0	10	65000
4	7	9.0	6	70000
5	3	7.0	10	62000
6	10	NaN	7	72000
7	11	7.0	8	80000

In [25]:

```
import math
median_test_score = math.floor(df['test_score(out of 10)'].mean())
median_test_score
```

Out[25]:

```
7
```

In [26]:

```
df['test_score(out of 10)'] = df['test_score(out of 10)'].fillna(median_test_score)
df
```

Out[26]:

	experience	test_score(out of 10)	interview_score(out of 10)	salary($)
0	0	8.0	9	50000
1	0	8.0	6	45000

	experience	test_score(out of 10)	interview_score(out of 10)	salary($)
2	5	6.0	7	60000
3	2	10.0	10	65000
4	7	9.0	6	70000
5	3	7.0	10	62000
6	10	7.0	7	72000
7	11	7.0	8	80000

In [27]:
```
reg = linear_model.LinearRegression()
reg.fit(df[['experience','test_score(out of 10)','interview_score(out of 10)']],df['salary($)'])
```

Out[27]:
```
LinearRegression()
```

In [28]:
```
reg.predict([[2,9,6]])
```

Out[28]:
```
array([53713.86677124])
```

In [29]:
```
reg.predict([[12,10,10]])
```

Out[29]:
```
array([93747.79628651])
```

cost function

In [30]:
```
import pandas as pd
```

In [31]:
```
df = pd.read_csv("https://raw.githubusercontent.com/codebasics/py/master/ML/3_gradient_descent/Exer
df
```

Out[31]:

	name	math	cs
0	david	92	98
1	laura	56	68
2	sanjay	88	81
3	wei	70	80
4	jeff	80	83
5	aamir	49	52
6	venkat	65	66
7	virat	35	30
8	arthur	66	68
9	paul	67	73

In [73]:
```
import numpy as np

def gradient_descent(x,y):
    m_curr = b_curr = 0
    iterations = 10
    n = len(x)
    learning_rate = 0.00001

    for i in range(iterations):
        y_predicted = m_curr * x + b_curr
        cost = (1/n) * sum([val**2 for val in (y-y_predicted)])
        md = -(2/n)*sum(x*(y-y_predicted))
        bd = -(2/n)*sum(y-y_predicted)
        m_curr = m_curr - learning_rate * md
        b_curr = b_curr - learning_rate * bd
        print ("m {}, b {}, cost {} iteration {}".format(m_curr,b_curr,cost, i))

x = np.array(df.cs)
y = np.array(df.math)

gradient_descent(x,y)
```

```
m 0.09891800000000002, b 0.001336, cost 4734.0 iteration 0
m 0.18754844079600005, b 0.0025336859159999997, cost 3806.234916447875 iteration 1
m 0.2669612367322398, b 0.0036074425220488718, cost 3061.411585493742 iteration 2
m 0.33811503020970224, b 0.00457015856424676, cost 2463.456559699491 iteration 3
m 0.4018687640567642, b 0.005433382348842311, cost 1983.409549731108 iteration 4
m 0.4589920503640901, b 0.006207461149043979, cost 1598.02081324698 iteration 5
m 0.5101744609524449, b 0.006901666113412, cost 1288.6251182559167 iteration 6
m 0.5560338516244612, b 0.007524304183678214, cost 1040.2377444902274 iteration 7
m 0.5971238206875977, b 0.008082818373023545, cost 840.8287401332442 iteration 8
m 0.6339403917847745, b 0.008583877615334823, cost 680.7402845432684 iteration 9
```

one hot coding by pandas

In [75]:

```
import pandas as pd
```

In [76]:

```
df = pd.read_csv("https://raw.githubusercontent.com/codebasics/py/master/ML/5_one_hot_encoding/Exer
df
```

Out[76]:

	Car Model	Mileage	Sell Price($)	Age(yrs)
0	BMW X5	69000	18000	6
1	BMW X5	35000	34000	3
2	BMW X5	57000	26100	5
3	BMW X5	22500	40000	2
4	BMW X5	46000	31500	4
5	Audi A5	59000	29400	5
6	Audi A5	52000	32000	5
7	Audi A5	72000	19300	6
8	Audi A5	91000	12000	8
9	Mercedez Benz C class	67000	22000	6
10	Mercedez Benz C class	83000	20000	7
11	Mercedez Benz C class	79000	21000	7
12	Mercedez Benz C class	59000	33000	5

In [91]:

```
dummies=pd.get_dummies(df['Car Model'])
dummies
```

Out[91]:

	Audi A5	BMW X5	Mercedez Benz C class
0	0	1	0
1	0	1	0
2	0	1	0
3	0	1	0
4	0	1	0
5	1	0	0

	Audi A5	BMW X5	Mercedez Benz C class
6	1	0	0
7	1	0	0
8	1	0	0
9	0	0	1
10	0	0	1
11	0	0	1
12	0	0	1

In [98]:

```
merge = pd.concat([df,dummies],axis = 'columns')
merge
```

Out[98]:

	Car Model	Mileage	Sell Price($)	Age(yrs)	Audi A5	BMW X5	Mercedez Benz C class
0	BMW X5	69000	18000	6	0	1	0
1	BMW X5	35000	34000	3	0	1	0
2	BMW X5	57000	26100	5	0	1	0
3	BMW X5	22500	40000	2	0	1	0
4	BMW X5	46000	31500	4	0	1	0
5	Audi A5	59000	29400	5	1	0	0
6	Audi A5	52000	32000	5	1	0	0
7	Audi A5	72000	19300	6	1	0	0
8	Audi A5	91000	12000	8	1	0	0
9	Mercedez Benz C class	67000	22000	6	0	0	1
10	Mercedez Benz C class	83000	20000	7	0	0	1
11	Mercedez Benz C class	79000	21000	7	0	0	1
12	Mercedez Benz C class	59000	33000	5	0	0	1

In [102…

```
final = merge.drop(['Car Model','Mercedez Benz C class'],axis = 'columns')
final
```

Out[102…

	Mileage	Sell Price($)	Age(yrs)	Audi A5	BMW X5
0	69000	18000	6	0	1
1	35000	34000	3	0	1
2	57000	26100	5	0	1
3	22500	40000	2	0	1
4	46000	31500	4	0	1
5	59000	29400	5	1	0
6	52000	32000	5	1	0
7	72000	19300	6	1	0
8	91000	12000	8	1	0
9	67000	22000	6	0	0
10	83000	20000	7	0	0
11	79000	21000	7	0	0

	Mileage	Sell Price($)	Age(yrs)	Audi A5	BMW X5
12	59000	33000	5	0	0

In [104…

```
x = final.drop(['Sell Price($)'],axis= 'columns')
x
```

Out[104…

	Mileage	Age(yrs)	Audi A5	BMW X5
0	69000	6	0	1
1	35000	3	0	1
2	57000	5	0	1
3	22500	2	0	1
4	46000	4	0	1
5	59000	5	1	0
6	52000	5	1	0
7	72000	6	1	0
8	91000	8	1	0
9	67000	6	0	0
10	83000	7	0	0
11	79000	7	0	0
12	59000	5	0	0

In [111…

```
y = final['Sell Price($)']
y
```

Out[111…

```
0     18000
1     34000
2     26100
3     40000
4     31500
5     29400
6     32000
7     19300
8     12000
9     22000
10    20000
11    21000
12    33000
Name: Sell Price($), dtype: int64
```

In [112…

```
from sklearn.linear_model import LinearRegression
model = LinearRegression()
```

In [115…

```
model.fit(x,y)
```

Out[115…

```
LinearRegression()
```

In [116…

```
model.score(x,y)
```

Out[116…

```
0.9417050937281083
```

In [118…

```
model.predict([[45000,4,0,0]])
```

Out[118…

```
array([36991.31721061])
```

In [117…

```
model.predict([[86000,7,0,1]])
```

Out[117… array([11080.74313219])

one hot coding by sklearn

In [147…

```
df
```

Out[147…

	Car Model	Mileage	Sell Price($)	Age(yrs)
0	1	69000	18000	6
1	1	35000	34000	3
2	1	57000	26100	5
3	1	22500	40000	2
4	1	46000	31500	4
5	0	59000	29400	5
6	0	52000	32000	5
7	0	72000	19300	6
8	0	91000	12000	8
9	2	67000	22000	6
10	2	83000	20000	7
11	2	79000	21000	7
12	2	59000	33000	5

In [148…

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
```

In [149…

```
dfle = df
dfle['Car Model'] = le.fit_transform(dfle['Car Model'])
dfle
```

Out[149…

	Car Model	Mileage	Sell Price($)	Age(yrs)
0	1	69000	18000	6
1	1	35000	34000	3
2	1	57000	26100	5
3	1	22500	40000	2
4	1	46000	31500	4
5	0	59000	29400	5
6	0	52000	32000	5
7	0	72000	19300	6
8	0	91000	12000	8
9	2	67000	22000	6
10	2	83000	20000	7
11	2	79000	21000	7
12	2	59000	33000	5

In [150…

```
x = dfle[['Mileage','Age(yrs)']].values
x
```

Out[150…

```
array([[69000,     6],
       [35000,     3],
       [57000,     5],
       [22500,     2],
       [46000,     4],
       [59000,     5],
       [52000,     5],
       [72000,     6],
       [91000,     8],
       [67000,     6],
       [83000,     7],
       [79000,     7],
       [59000,     5]], dtype=int64)
```

In [151…

```
y = dfle['Sell Price($)'].values
y
```

Out[151…

```
array([18000, 34000, 26100, 40000, 31500, 29400, 32000, 19300, 12000,
       22000, 20000, 21000, 33000], dtype=int64)
```

In [152…

```
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer
ct = ColumnTransformer([('Car Model', OneHotEncoder(), [0])], remainder = 'passthrough')
```

In [153…

```
x = ct.fit_transform(x)
x
```

Out[153…

```
<13x13 sparse matrix of type '<class 'numpy.float64'>'
        with 26 stored elements in Compressed Sparse Row format>
```

In [154…

```
x = x[:,1:]
x
```

Out[154…

```
<13x12 sparse matrix of type '<class 'numpy.float64'>'
        with 25 stored elements in Compressed Sparse Row format>
```

In [155…

```
model.fit(x,y)
```

Out[155…

```
LinearRegression()
```

In [ ]: