メモを取れる。

変数

In [2]:

```
a = 1
```

In [3]:

```
print(a)
```

1

In [4]:

```
type(a)
```

Out[4]:

int

In [5]:

```
10 / 2
```

Out[5]:

5.0

比較演算

In [6]:

```
1 < 2
```

Out[6]:

True

In [7]:

```
a = 3
```

In [8]:

```
a < 2
```

Out[8]:

False

In [9]:

```
a ==3
```

Out[9]:

True

In [10]:

```
a == 4
```

Out[10]:

False

エスケープシーケンス

In [12]:

```
print('私はキカガクです。\n宜しくおねがいします。\n\tタブ')
```

私はキカガクです。
宜しくおねがいします。
	タブ

複数の変数をつかう

リスト

In [13]:

```
num = [4, 5, 6, 7]
```

In [14]:

```
num
```

Out[14]:

[4, 5, 6, 7]

In [15]:

```
num[0:2] # 2の手前1まで
```

Out[15]:

[4, 5]

In [16]:

```
num[1:3]
```

Out[16]:

[5, 6]

In [17]:

```
num[:2] #num[0:2]
```

Out[17]:

[4, 5]

In [18]:

```
num[2:] # num2~全部
```

Out[18]:

[6, 7]

タプル

In [22]:

```
t = (4, 5, 6, 7) #=> タプルは()書き換えができない ex,t[0] = 1
```

In [20]:

```
t
```

Out[20]:

(4, 5, 6, 7)

In [21]:

```
t[0]
```

Out[21]:

4

辞書

In [23]:

```
results = {'数学':90, '理科':80} #辞書の定義{}
```

In [24]:

```
results
```

Out[24]:

{'数学': 90, '理科': 80}

In [26]:

```
results['数学']
```

Out[26]:

90

制御構文

for文：繰り返し

In [32]:

```
for i in range(5):
    print(i)
print('キカガク')
```

```
0
1
2
3
4
キカガク
```

In [33]:

```
names = ['佐藤', '鈴木', '高橋', '吉田']
```

In [34]:

```
for i in range(len(names)):
    print(names[i] + 'さん')
```

```
佐藤さん
鈴木さん
高橋さん
吉田さん
```

In [35]:

```
for name in names:
    print(name + 'さん')
```

```
佐藤さん
鈴木さん
高橋さん
吉田さん
```

if文：条件分岐

In [40]:

```
val = -1
if val > 0:
    print('正の値です')
elif val == 0:
    print('0です')
else:
    print('負の値です')
```

```
負の値です
```

関数

入力（引数）のある関数

In [41]:

```
# name: 引数
def say_hello(name):
    print('こんにちは' + name + 'さん')
```

In [44]:

```
# 使う
say_hello('幾何学')
```

こんにちは幾何学さん

出力(返り・戻り値)のある関数

In [55]:

```
def add(a, b):
#     print(a + b)
    return a + b
```

In [56]:

```
result = add(3,5)
```

In [57]:

```
result
```

Out[57]:

8

In [58]:

```
abs(-1)
```

Out[58]:

1

In [67]:

```
# 絶対値を返すabs関数(名前はabso)を作成
def abso(n):
    if n < 0:
        return n * (-1)
    elif n == 0:
        return n
    else:
        return n
```

```
abso(-5.2)
```

Numpy：数値計算

In [69]:

```
import numpy as np
```

In [70]:

```
# ベクトルの定義
```

In [71]:

```
x = np.array([1,2,3])
```

In [72]:

```
x
```

Out[72]:

```
array([1, 2, 3])
```

In [73]:

```
y = np.array([2,3.9,6.1])
```

In [74]:

```
y
```

Out[74]:

```
array([2. , 3.9, 6.1])
```

データの平均

In [77]:

```
# 平均の計算 (1+ 2+3)/3 = 2
x.mean()
```

Out[77]:

```
2.0
```

In [78]:

```
y.mean()
```

Out[78]:

```
4.0
```

In [81]:

```
# 中心化
xc = x -x.mean()
xc
```

Out[81]:

array([-1., 0., 1.])

In [82]:

```
yc = y -y.mean()
yc
```

Out[82]:

array([-2. , -0.1, 2.1])

パラメータaの計算

In [84]:

```
# 要素ごとの掛け算(要素積)
xx = xc * xc
xx
```

Out[84]:

array([1., 0., 1.])

In [87]:

```
xy = xc * yc
```

In [88]:

```
xy
```

Out[88]:

array([2. , -0. , 2.1])

In [90]:

```
xx.sum()
```

Out[90]:

2.0

In [91]:

```
xy.sum()
```

Out[91]:

4.1

In [92]:

```
a = xy.sum()/xx.sum()
```

In [93]:

```
a
```

Out[93]:

2.05

Pandas : データベースの操作

In [96]:

```
import pandas as pd
```

In [98]:

```
# CSVファイルの読み込み
df = pd.read_csv('original.csv')
```

In [99]:

```
print(df)
```

```
       x       y
0   40.362  137500.0
1   40.686  132500.0
2   38.430   93000.0
3   36.822   96500.0
4   37.002  100500.0
..     ...      ...
95  47.250  250000.0
96  43.722  166500.0
97  42.642  151500.0
98  43.644  173000.0
99  41.850  174500.0

[100 rows x 2 columns]
```

In [101]:

```
df.head(3)
```

Out[101]:

	x	y
0	40.362	137500.0
1	40.686	132500.0
2	38.430	93000.0

In [102]:

```
# データ抽
x = df['x']
y = df['y']
```

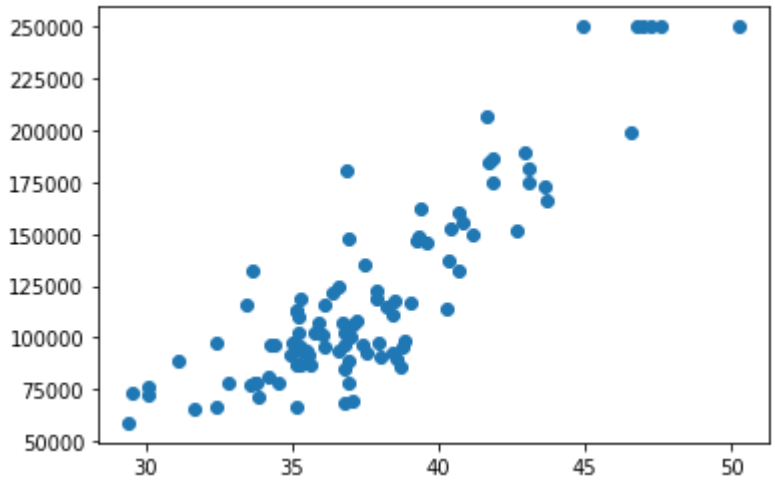
Matplotlib：グラフの描画

In [104]:

```
import matplotlib.pyplot as plt
```

In [106]:

```
# 横軸をx, 縦軸をyの散布図(scatter)をプロット
plt.scatter(x,y)
plt.show()
```



単回帰分析の実装

データの中心化

In [108]:

```
# データの概要を表示
df.describe()
```

Out[108]:

	x	y
count	100.000000	100.000000
mean	37.622220	121065.000000
std	4.087547	47174.009226
min	29.418000	59000.000000
25%	35.151000	90375.000000
50%	36.909000	104250.000000
75%	39.439500	147250.000000
max	50.250000	250000.000000

In [109]:

```
df.mean()
```

Out[109]:

```
x       37.62222
y   121065.00000
dtype: float64
```

In [112]:

```
# 中心化
df_c = df - df.mean() #=> 全てに対して平均が引かれる
```

In [114]:

```
df_c.head(3)
```

Out[114]:

	x	y
0	2.73978	16435.0
1	3.06378	11435.0
2	0.80778	-28065.0

In [115]:

```
df_c.describe() #=> 中心化されたあとのデータ
```

Out[115]:

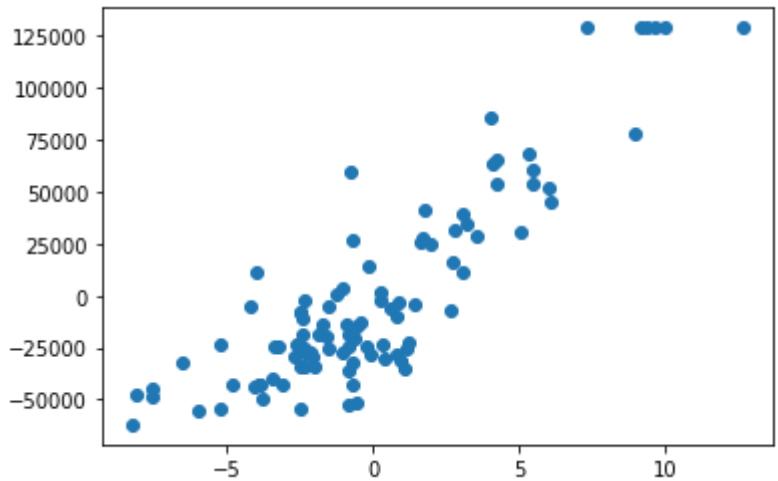
	x	y
count	1.000000e+02	100.000000
mean	1.918465e-15	0.000000
std	4.087547e+00	47174.009226
min	-8.204220e+00	-62065.000000
25%	-2.471220e+00	-30690.000000
50%	-7.132200e-01	-16815.000000
75%	1.817280e+00	26185.000000
max	1.262778e+01	128935.000000

In [116]:

```
# データの抽出
x = df_c['x']
y = df_c['y']
```

In [118]:

```
# xとyの散布図をプロット
plt.scatter(x,y)
plt.show()
```



パラメータaの計算(TEXテフ表示)

傾きaの計算式

$$a = \frac{\sum_{n=1}^{N} x_n y_n}{\sum_{n=1}^{N} x_n^2}$$

In [119]:

```
xx = x * x  # 要素積
```

In [120]:

```
xy = x * y
```

In [122]:

```
a = xy.sum() /xx.sum()
```

In [123]:

```
a
```

Out[123]:

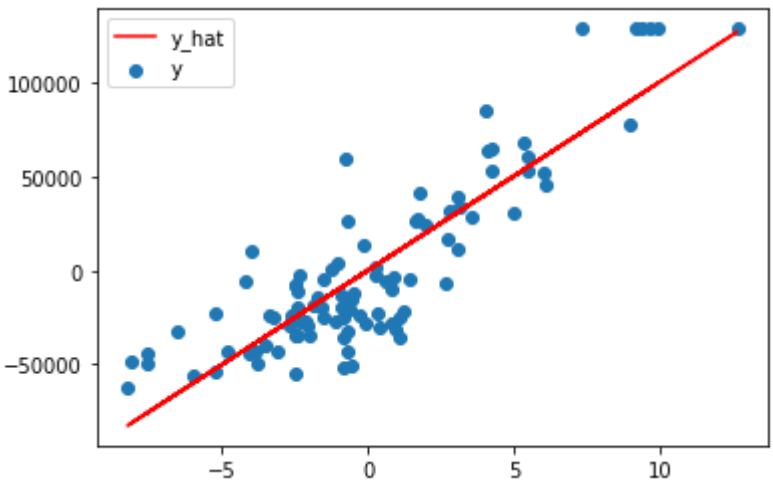
10069.022519284063

予測値をプロットして確認

$$\hat{y} = ax$$

In [128]:

```
plt.scatter(x,y,label='y') # 実測値
plt.plot(x, a*x,label='y_hat', color='red') #予測値
plt.legend() # 凡例の表示
plt.show()
```



予測値の計算

$$y - \bar{y} = a(x - \bar{x})$$
$$\rightarrow y = a(x - \bar{x}) + \bar{y}$$

In [129]:

```
x_new = 40 #40平米の部屋ならどうだろう(推論の例)
```

In [130]:

```
mean = df.mean()
```

In [131]:

```
mean['x']
```

Out[131]:

37.62222

In [132]:

```
# 中心化
xc = x_new - mean['x']
```

In [133]:

```
xc
```

Out[133]:

2.3777800000000013

In [136]:

```
# 単回帰分析による予測
yc = a * xc
yc
```

Out[136]:

23941.920365903272

In [138]:

```
# 元のスケールの予測値
y_hat = a * xc + mean['y']
y_hat
```

Out[138]:

145006.92036590326

In [141]:

```
a
```

Out[141]:

10069.022519284063

In [142]:

```
mean['x']
```

Out[142]:

37.62222

In [143]:

```
mean['y']
```

Out[143]:

121065.0

予測値を計算する関数の作成

In [147]:

```
def predict(x):
    # 定数項
    a = 10069.022519284063
    xm = 37.62222
    ym = 121065.0
    # 中心化
    xc = x - xm
    # 予測値の計算
    y_hat = a * xc + ym
    # 出力
    return y_hat
```

In [148]:

```
# 予測値
predict(40)
```

Out[148]:

145006.92036590326

In [149]:

```
predict(25) #=> もともと30~のデータ(内挿)なので外装のためマイナス表記
```

Out[149]:

-6028.417423357663

In [150]:

```
predict(30)
```

Out[150]:

44316.695173062646

In [ ]:

```
```