機器學習

Lecture 1 Python

Python

- 成為python數據分析達人的第一課
  http://moocs.nccu.edu.tw/course/123/intro

- 常用的套件
  - Numpy: n-dimensional array (矩陣)運算
  - Pandas:資料處理
  - Matplotlib: 繪圖
  - SciKit-Learn: 包含機器學習常用的演算法

關鍵字

- 具有語法功能的保留字
- Python 有 33個關鍵字

True	break	else	if	not	while
False	class	except	import	or	with
None	contiune	finally	in	pass	yield
and	def	for	is	raise	
as	del	from	lambda	return	
assert	elif	global	nonlocal	try	

https://www.itread01.com/articles/1506142629.html

識別字 (identifier)

- 寫程式時依據需求自行定義的名稱
- 包括變數(Variable)、函數(function)、類別(class)等
- 除了底線、英文字母和數字外，亦可用中文，但不可與關鍵字相同

```
In [1]: 變數 = 100

In [2]: 變數
Out[2]: 100
```

數學運算

- + - * / %

```
In [3]: 2+3
Out[3]: 5

In [4]: 5/2
Out[4]: 2.5

In [5]: 5%2
Out[5]: 1

In [6]: 2**3
Out[6]: 8

In [7]: 5//2
Out[7]: 2
```

字串 (String)

- 字串通常用單引號或雙引號包起來 （引號必須兩兩成對出現）
- 連續使用三個雙引號可以建立多行字串

```
In [11]: a = 'single quoted string'
         b = "double quoted string"
         c = 'Alice said "Thank you" '
         d = """ line 1
         line 2
         line 3"""
```

列表 (List)

- 最常用的資料結構
- 儲存一堆按照順序排列的東西

```
In [12]: a = []
         b = [1, 2, 3]
         c = [1, "string", [1,2,3]]
```

```
In [14]: len(a)
Out[14]: 0

In [15]: sum(b)
Out[15]: 6
```

列表 (List)

```
In [23]: x = [0, 1, 2, 3, 4]

In [24]: x[0]
Out[24]: 0

In [25]: x[1]
Out[25]: 1

In [26]: x[-1]
Out[26]: 4

In [27]: x[0] = -1

In [28]: x
Out[28]: [-1, 1, 2, 3, 4]
```

```
In [29]: x[:2]
Out[29]: [-1, 1]

In [30]: x[2:]
Out[30]: [2, 3, 4]

In [31]: x[1:3]
Out[31]: [1, 2]

In [32]: x[-2:]
Out[32]: [3, 4]
```

列表 (List)

```
In [34]: x = [1, 2, 3]
         x.extend([4, 5, 6])

In [35]: x
Out[35]: [1, 2, 3, 4, 5, 6]

In [36]: x = [1, 2, 3]
         y = x + [4, 5, 6]

In [37]: y
Out[37]: [1, 2, 3, 4, 5, 6]

In [38]: x = [1, 2, 3]
         x.append(4)

In [39]: x
Out[39]: [1, 2, 3, 4]
```

列表 (List)

```
In [46]: a = [1, 2]
         x, y = a

In [47]: x
Out[47]: 1

In [48]: y
Out[48]: 2
```

```
In [51]: _, z = a
         z
Out[51]: 2
```

如果等號兩邊的數量不一致？

序對 (Tuple)

- 類似 list，最大的不同為 tuple 是一種唯讀且不可變更的資料結構，也就是不可取代tuple中的任意一個元素

```
In [52]: list = [1, 2]
         tuple = (1, 2)
         list[1] = 3
         list

Out[52]: [1, 3]
```

```
In [55]: tuple[1] = 3

---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-55-36364d40dca7> in <module>()
----> 1 tuple[1] = 3

TypeError: 'tuple' object does not support item assignment
```

字典 (Dictionary)

- 包含 key：value 配對的資料結構
- 字典中的元素是無序的
- key 不能是 list，但可以是 tuple

```
In [57]: dict = {1: 45, "a": 2}
         dict["a"]

Out[57]: 2
```

```
In [59]: dict['b'] = 35

In [60]: dict

Out[60]: {1: 45, 'a': 2, 'b': 35}
```

字典 (Dictionary)

```
In [61]: dict[[1, 2]] = 5

---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-61-9e8f8a0aa6c2> in <module>()
----> 1 dict[[1, 2]] = 5

TypeError: unhashable type: 'list'

In [62]: dict[(1, 2)] = 5

In [63]: dict
Out[63]: {1: 45, 'a': 2, 'b': 35, (1, 2): 5}
```

集合 (Set)

- 類似數學中的集合
- 包含一堆各不相同的元素

```
In [65]: a = [1, 2, 3, 3]
         setA = set(a)
         setA
Out[65]: {1, 2, 3}

In [68]: setB = {2, 3, 4}
         setA - setB
Out[68]: {1}

In [69]: setA | setB   # 聯集
Out[69]: {1, 2, 3, 4}

In [70]: setA & setB   # 交集
Out[70]: {2, 3}
```

集合 (Set)

- 使用集合的兩個時機
  1. 檢查某個元素是否存在時，速度會比 list 快

```
In [71]: listA = [1, 2, 3, 4, 5]
         setA = set(listA)
         1 in listA
Out[71]: True

In [72]: 1 in setA
Out[72]: True
```

集合 (Set)

- 使用集合的兩個時機
  2. 只是想從一大堆資料中，找出不同的項目

```
In [73]: listB = [1, 2, 3, 1, 2, 3]
         setB = set(listB)
         l = len(listB), len(setB)
         l

Out[73]: (6, 3)
```

控制流程

- if 條件判斷

```
In [82]: if i < 3:
             i += 1
         elif i == 3:
             i = 100
         else: i = 0
```

Python 使用縮排的方式來切分程式碼的區塊

- 也可以寫在一行

```
In [83]: parity = "even" if x % 2 == 0 else "odd"
```

控制流程

- 迴圈
  - for loop (指定執行次數)
  - while loop (指定條件)

控制流程

- for 迴圈

range(1, count+1, 1) 產生 1, 2, ..., count
range(1, 10+1, 2) 產生 1, 3, 5, 7, 9

```
In [86]: sum = 0

         for x in range(1, 10, 1):
             sum+=x

In [87]: sum
Out[87]: 45
```

控制流程

In [91]:
```
help(range)
```

```
Help on class range in module builtins:

class range(object)
 |  range(stop) -> range object
 |  range(start, stop[, step]) -> range object
 |
 |  Return an object that produces a sequence of integers from start (inclusive)
 |  to stop (exclusive) by step.  range(i, j) produces i, i+1, i+2, ..., j-1.
 |  start defaults to 0, and stop is omitted!  range(4) produces 0, 1, 2, 3.
 |  These are exactly the valid indices for a list of 4 elements.
 |  When step is given, it specifies the increment (or decrement).
 |
```

控制流程

- while 迴圈

In [89]:
```
x = 0
while x < 10:
    print(x, "is less than 10")
    x +=2
```

```
0 is less than 10
2 is less than 10
4 is less than 10
6 is less than 10
8 is less than 10
```

解析式列表 (list comprehensions)

- 將某個列表轉換成另一個列表，比如挑選其中幾個元素，或對某些元素進行轉換

```
In [27]: even_numbers = [x for x in range(0, 5, 1) if x % 2 == 0]
         squares = [x * x for x in range(0, 5, 1)]
         even_squares = [x * x for x in even_numbers]

In [28]: even_numbers
Out[28]: [0, 2, 4]

In [29]: squares
Out[29]: [0, 1, 4, 9, 16]

In [30]: even_squares
Out[30]: [0, 4, 16]
```

解析式字典或解析式集合

```
In [31]: square_dict = {x: x * x for x in range(0, 5, 1)}
         square_set = {x * x for x in [1, -1]}

In [32]: square_dict
Out[32]: {0: 0, 1: 1, 2: 4, 3: 9, 4: 16}

In [33]: square_set
Out[33]: {1}
```

函數 (function)

```
In [35]: def double(x):
             return x * 2

         double(3)
Out[35]: 6
```

函數 (function)

```
In [35]: def double(x):
             return x * 2

         double(3)
Out[35]: 6
```

```
In [38]: def apply_to_one(f):
             return f(1)

         apply_to_one(double)
```

函數 (function)

```
In [35]: def double(x):
             return x * 2

         double(3)
Out[35]: 6
```


```
In [38]: def apply_to_one(f):
             return f(1)

         apply_to_one(double)
Out[38]: 2
```

函數 (function)

```
In [35]: def double(x):
             return x * 2

         double(3)
Out[35]: 6

In [38]: def apply_to_one(f):
             return f(1)

         apply_to_one(double)
Out[38]: 2

In [39]: apply_to_one(lambda x: x + 4)
Out[39]: 5
```

lambda: 類似函數，卻又不像函數需要額外命名函數的識別字

numpy

In [1]:
```
import numpy as np

a = np.array( [1, 2, 3] )
b = np.array( [4, 5, 6] )
```

In [2]:
```
a[0]
```

Out[2]: 1

In [3]:
```
a*b
```

Out[3]: array([4, 10, 18])

In [4]:
```
np.dot(a, b)
```

Out[4]: 32

numpy

```
a = np.array( [1, 2, 3] )
b = np.array( [4, 5, 6] )
```

```
In [5]: np.sum(a)
Out[5]: 6

In [6]: np.max(a)
Out[6]: 3

In [7]: np.min(a)
Out[7]: 1

In [8]: a.size
Out[8]: 3
```

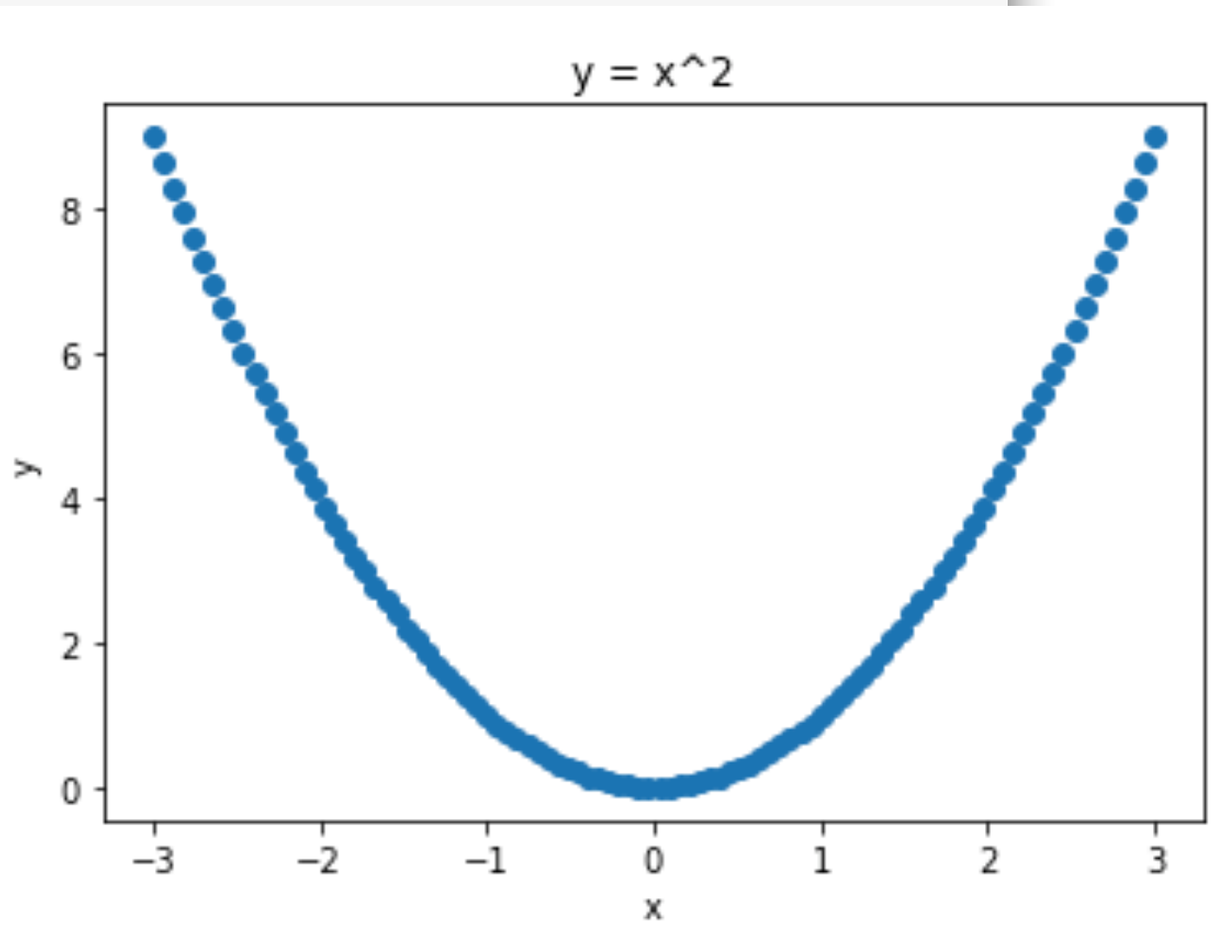
numpy

```
a = np.array( [1, 2, 3] )
b = np.array( [4, 5, 6] )
```

```
In [9]:  c = np.append(a, 5)

In [10]: c
Out[10]: array([1, 2, 3, 5])

In [11]: d = np.append(a, b)

In [12]: d
Out[12]: array([1, 2, 3, 4, 5, 6])
```

matplotlib

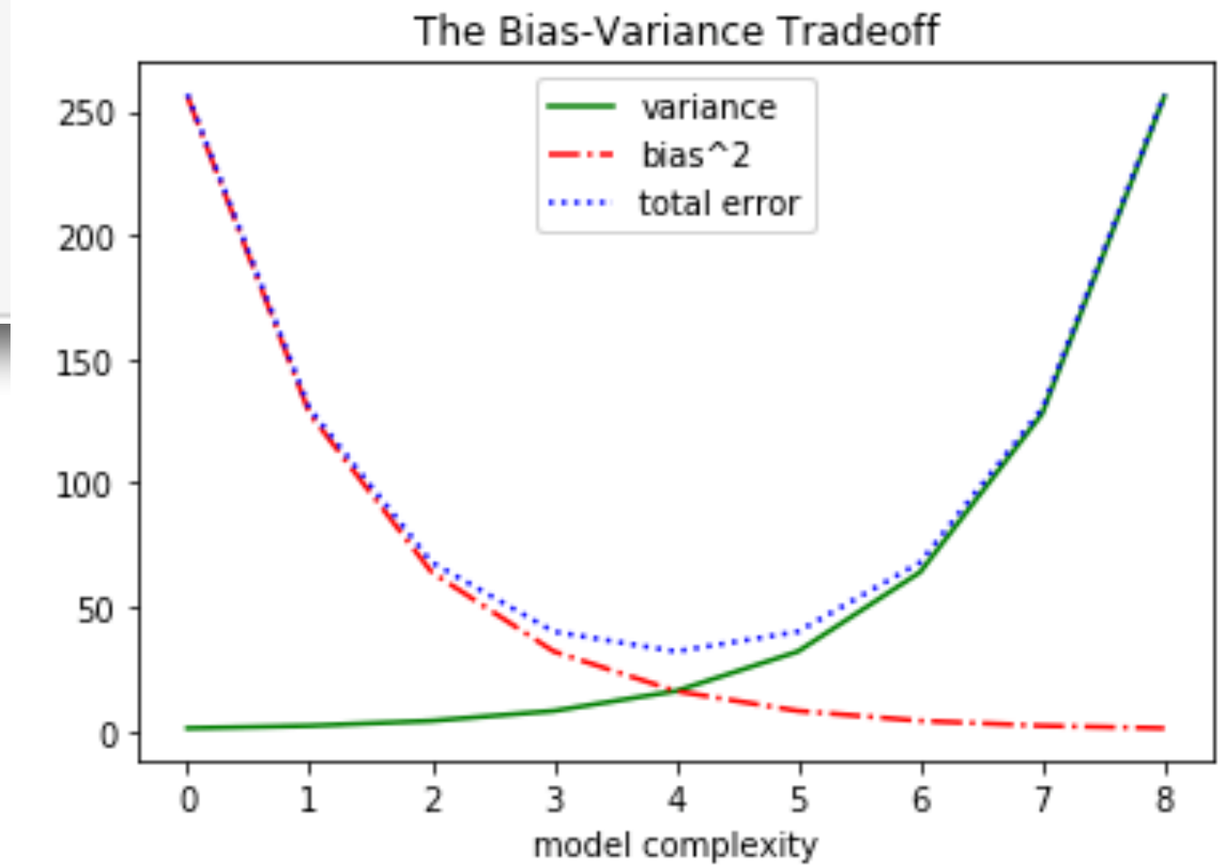
```
import numpy as np
import matplotlib.pyplot as plt

%matplotlib inline

x = np.linspace(-3, 3, 100)  # 生成一個 -3 到 3，個數為100 的等差級數
y = x**2
plt.plot(x, y, "o")

plt.title("y = x^2")

plt.xlabel("x")
plt.ylabel("y")
```



matplotlib

```
variance     = [1,2,4,8,16,32,64,128,256]
bias_squared = [256,128,64,32,16,8,4,2,1]
total_error  = [x + y for x, y in zip(variance, bias_squared)]

xs = range(len(variance))

plt.plot(xs, variance,     'g-',  label='variance')
plt.plot(xs, bias_squared, 'r-.', label='bias^2')
plt.plot(xs, total_error,  'b:',  label='total error')

plt.legend(loc=9)      # loc=9 中間偏上
plt.xlabel("model complexity")
plt.title("The Bias-Variance Tradeoff")
```

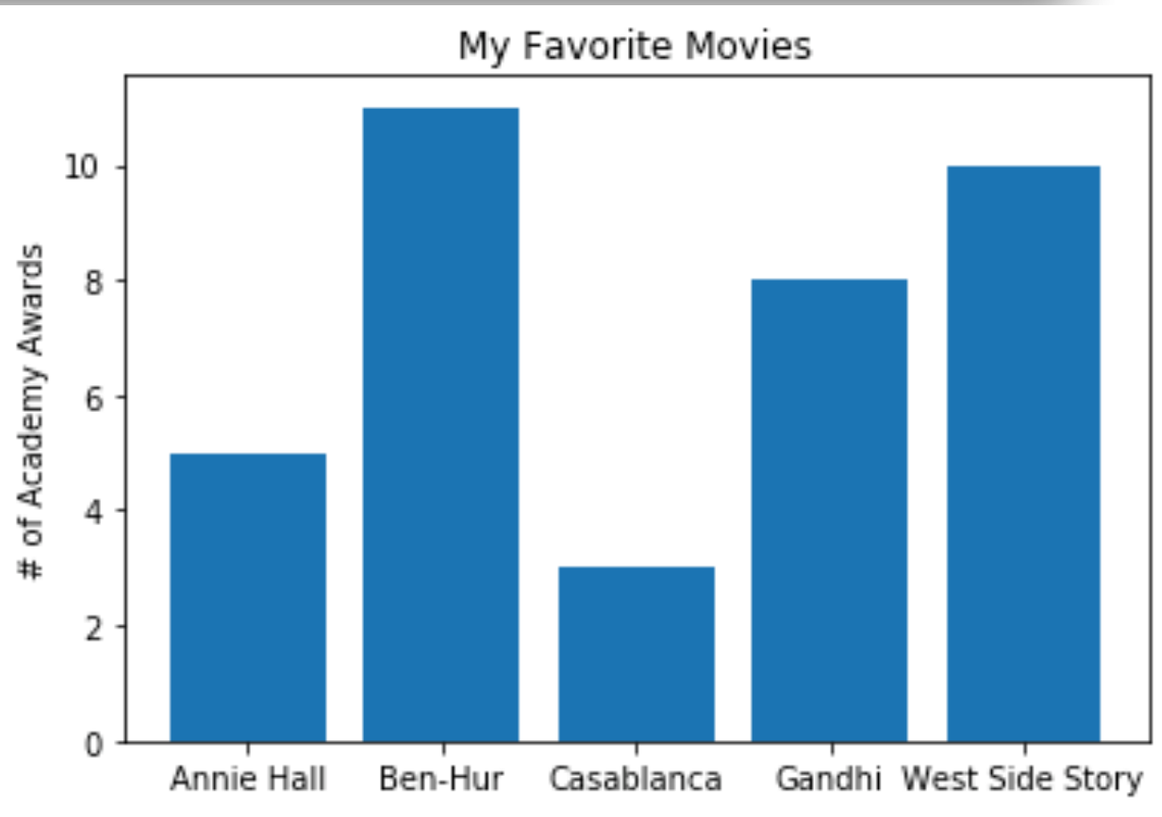
zip: 將多個列表轉換成元組列表



matplotlib

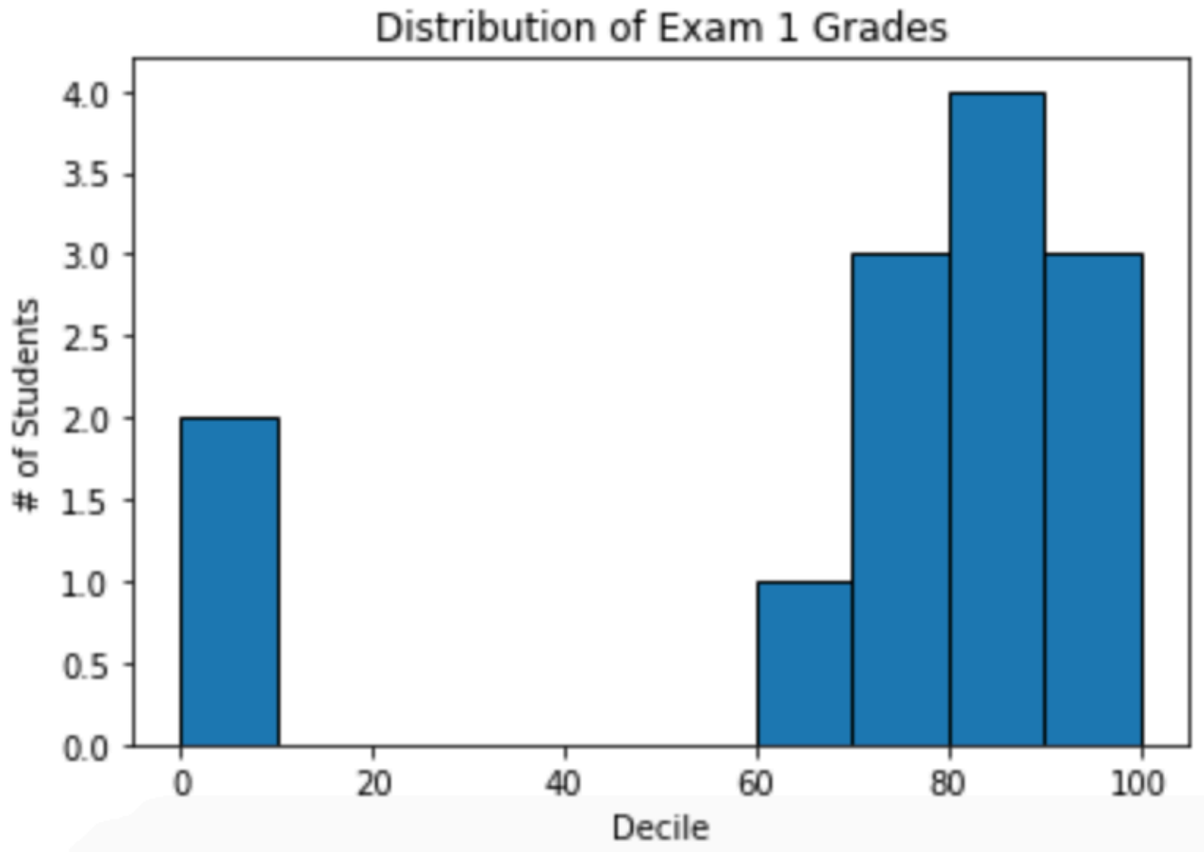
```
movies = ["Annie Hall", "Ben-Hur", "Casablanca", "Gandhi", "West Side Story"]
num_oscars = [5, 11, 3, 8, 10]

xs = [i  for i, _ in enumerate(movies)]

plt.bar(xs, num_oscars)
plt.ylabel("# of Academy Awards")
plt.title("My Favorite Movies")

plt.xticks([i  for i, _ in enumerate(movies)], movies)
```

enumerate: 產生許多（key, value）的 tuple



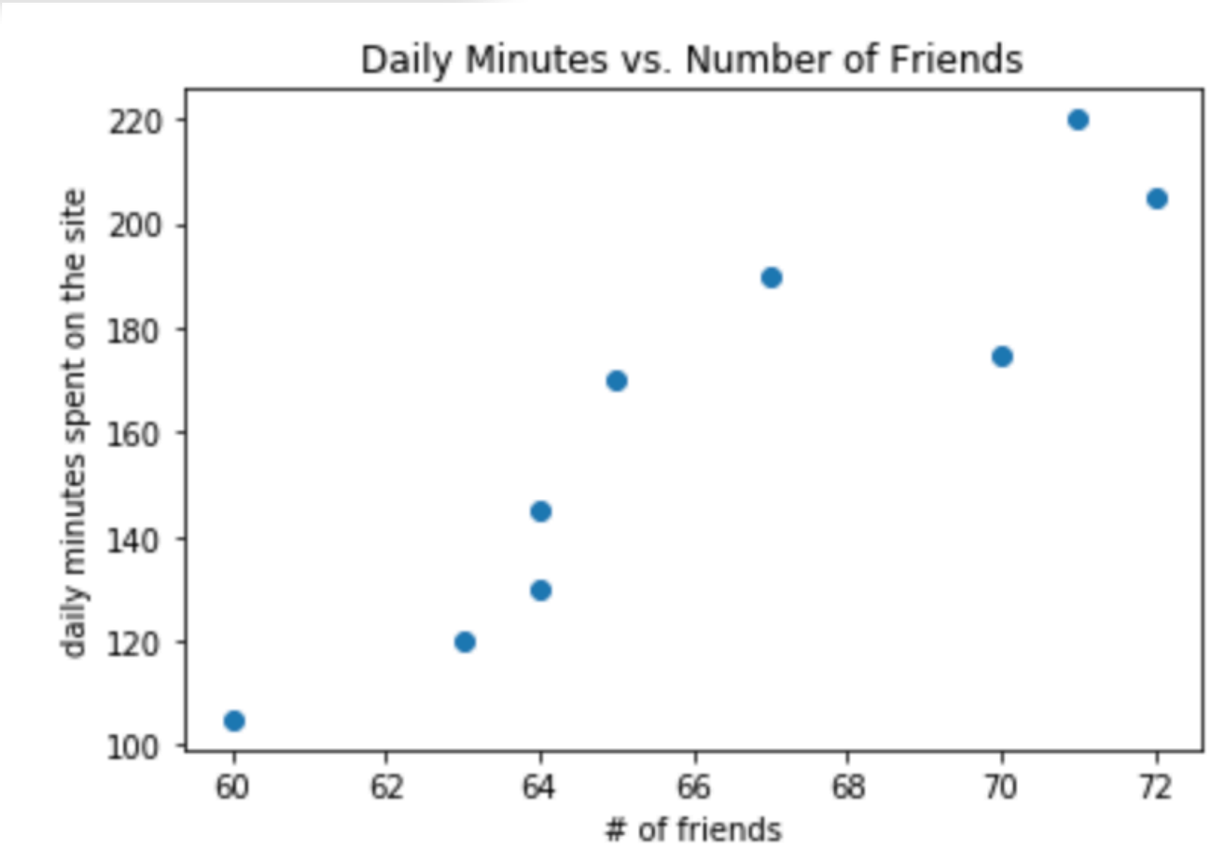
matplotlib

```
grades = np.array([83,95,91,87,70,0,85,82,100,67,73,77,0])

plt.hist(grades, bins = [0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100],
         edgecolor = 'black')

plt.xlabel("Decile")
plt.ylabel("# of Students")
plt.title("Distribution of Exam 1 Grades")
```



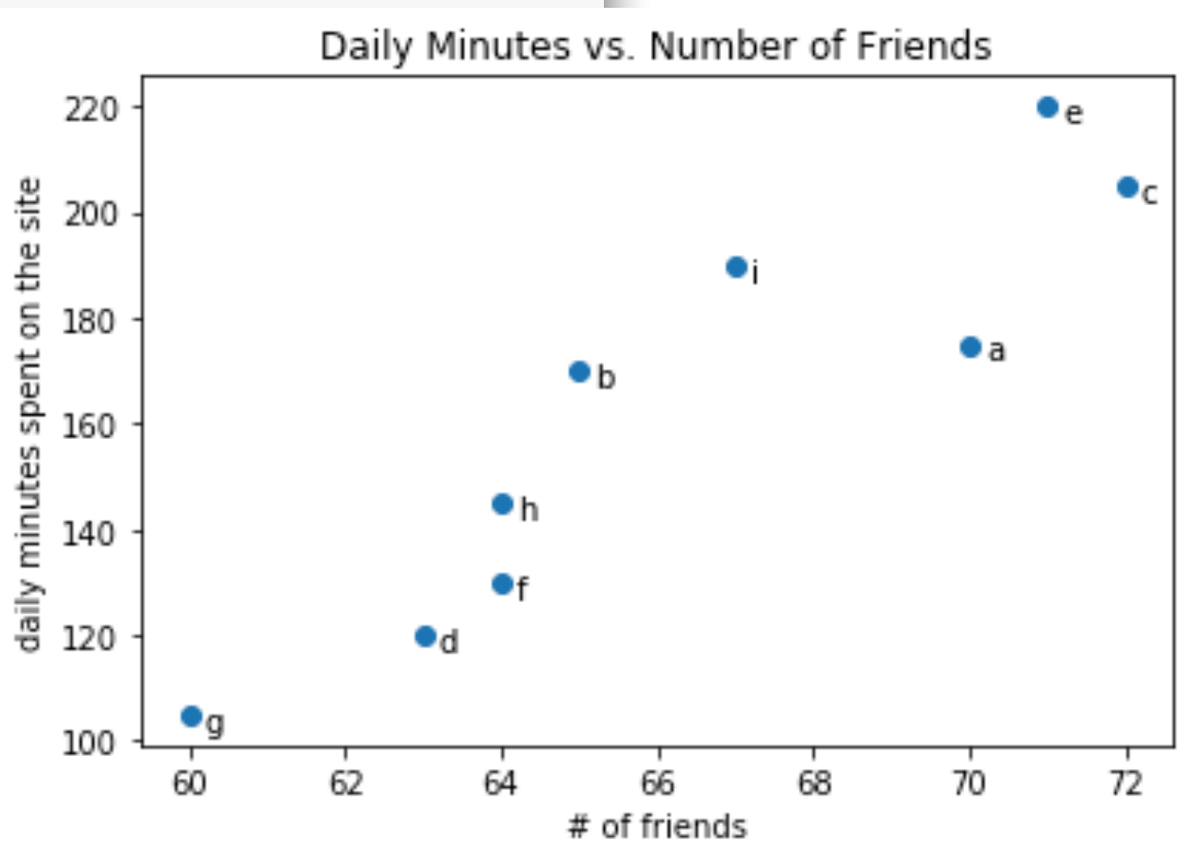
matplotlib

```
friends = [ 70, 65, 72, 63, 71, 64, 60, 64, 67]
minutes = [175, 170, 205, 120, 220, 130, 105, 145, 190]
labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i']

plt.scatter(friends, minutes)

plt.title("Daily Minutes vs. Number of Friends")
plt.xlabel("# of friends")
plt.ylabel("daily minutes spent on the site")
```



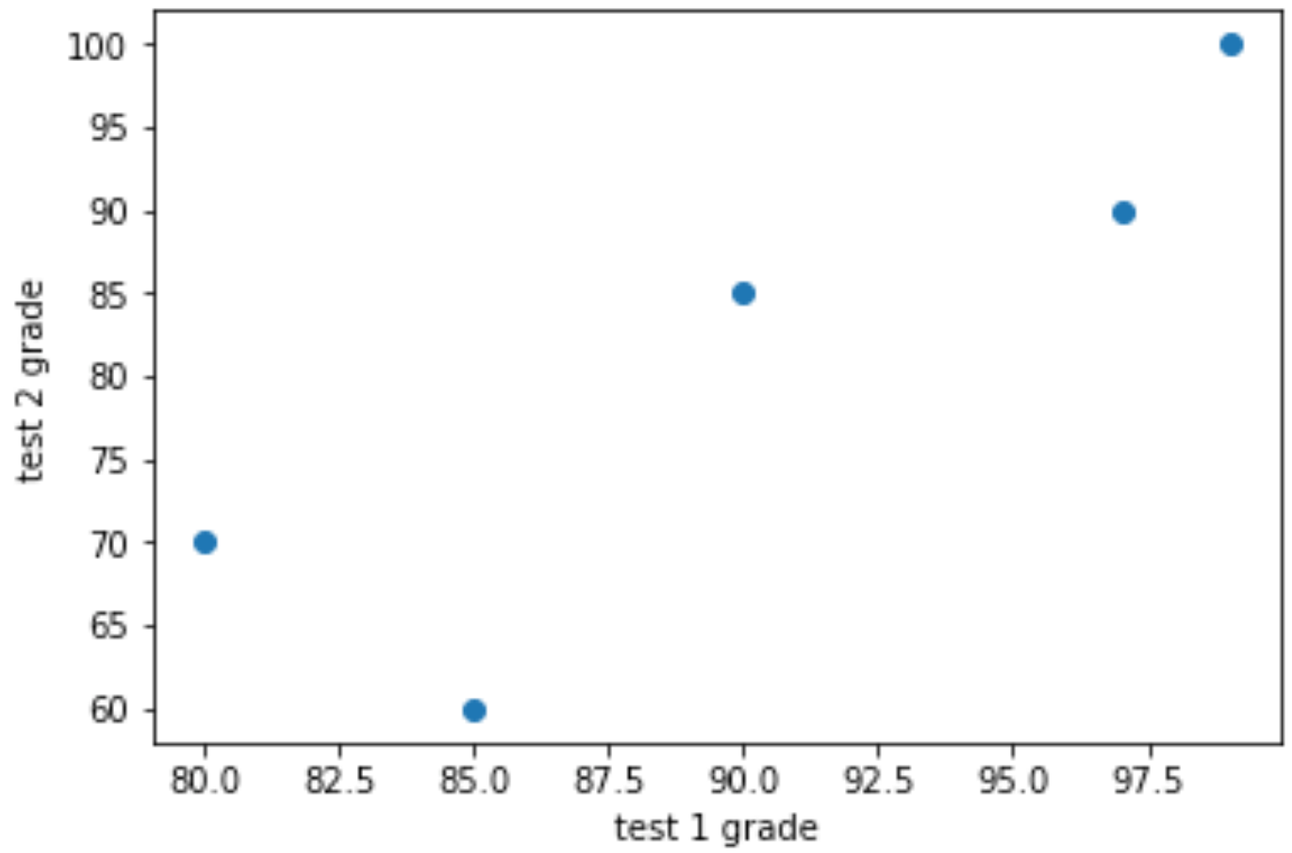
matplotlib

```
friends = [ 70, 65, 72, 63, 71, 64, 60, 64, 67]
minutes = [175, 170, 205, 120, 220, 130, 105, 145, 190]
labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i']

plt.scatter(friends, minutes)

for label, friend_count, minute_count in zip(labels, friends, minutes):
    plt.annotate(label,
    xy=(friend_count, minute_count),    # 把標籤放到對應的點上
    xytext=(5, -5),
    textcoords='offset points')

plt.title("Daily Minutes vs. Number of Friends")
plt.xlabel("# of friends")
plt.ylabel("daily minutes spent on the site")
```



matplotlib

```
test_1_grades = [ 99, 90, 85, 97, 80]
test_2_grades = [100, 85, 60, 90, 70]

plt.scatter(test_1_grades, test_2_grades)
plt.xlabel("test 1 grade")
plt.ylabel("test 2 grade")
```



matplotlib

```
test_1_grades = [ 99, 90, 85, 97, 80]
test_2_grades = [100, 85, 60, 90, 70]

plt.scatter(test_1_grades, test_2_grades)
plt.xlabel("test 1 grade")
plt.ylabel("test 2 grade")
plt.axis("equal")
```