ファイルの実行

In [1]:
```
%run -i test.py
```

```
15
```

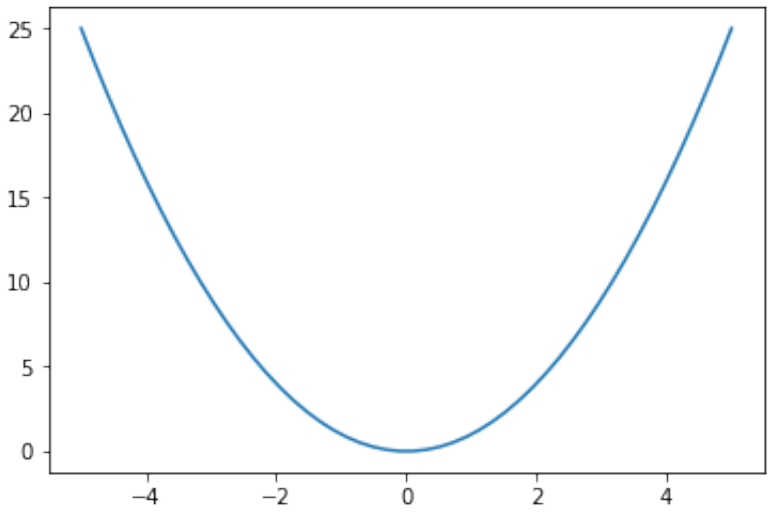
In [2]:
```
# %run はマジックコマンドで、このノートブックでのみ意味を持つ
```

グラフの表示

In [9]:
```
%matplotlib inline
import matplotlib.pyplot as plt
import numpy as np
```

In [10]:
```
# %matplotlib inline もマジックコマンドで、グラフをノートブック内で表示
```

In [13]:
```
x = np.linspace(-5,5)  # 描画範囲
y = x**2
plt.plot(x,y)
plt.show()
```



基本構文

In [14]:
```
for i in range(3):
    print("hello")
```

```
hello
hello
hello
```

数学的な記述

In [16]:
```
import math
```

In [18]:
```
math.sqrt(2)
```

Out[18]: 1.4142135623730951

In [20]:
```
math.exp(1)
```

Out[20]: 2.718281828459045

In [21]:
```
math.log(10)
```

Out[21]: 2.302585092994046

In [25]:
```
math.cos(2*math.pi)
```

Out[25]: 1.0

In [26]:
```
math.pi
```

Out[26]: 3.141592653589793

In [27]:
```
math.sin(math.pi/2)
```

Out[27]: 1.0

In [28]:
```
math.tan(math.pi/4)
```

Out[28]: 0.9999999999999999

演算子

In [34]:
```
a = 1
a += 1
a
```

Out[34]: 2

In [35]:
```
a *= 2
a
```

Out[35]: 4

In [36]:
```
c = "hello"
c
```

Out[36]: 'hello'

In [40]:
```
d = """rugby
tennis
soccer"""
print(d)    #  """で複数行をコメントアウト
```

```
rugby
tennis
soccer
```

スライシング

In [41]:
```
a = "afivajfvo"
a
```

Out[41]: 'afivajfvo'

In [42]:
```
len(a)
```

Out[42]: 9

In [43]:
```
a[:3]
```

Out[43]: 'afi'

In [44]:
```
a[-1]
```

Out[44]: 'o'

format

In [49]:
```
a = 1
b = 3
c = 3
```

In [47]:
```
s = "{}と{}の和は、{}です。".format(a,b,c)
s
```

Out[47]: '1と2の和は、3です。'

In [54]:
```
t = "{}を{}で割ると、{:5.3f}です。".format(a,b,a/b)
t
```

Out[54]: '1を3で割ると、0.333です。'

In [55]:
```
print(a
      +
      b
     )
```

```
4
```

制御構文

In [56]:
```
x = 100
if x >100:
    print("big")

else:
    print("small")
```

```
small
```

In [72]:
```
a = 0
for i in range(1,5):
    a = a + i
print(a)
```

```
10
```

In [73]:
```
for i in range(1,10,2):
    print(i)
```

```
1
3
5
7
9
```

In [75]:
```
i = 0
while i < 5:
    print(i)
    i += 1
```

```
0
1
2
3
4
```

In [79]:
```
for i in range(1,21):
    if i%15==0:
        print("fizzbuzz")
    elif i%3==0:
        print("fizz")
    elif i%5==0:
        print("buzz")
    else:
        print(i)
```

```
1
2
fizz
4
buzz
fizz
7
8
fizz
buzz
11
fizz
13
14
fizzbuzz
16
17
fizz
19
buzz
```

リスト

In [86]:
```
l = [1,2,3,4]
l
```

Out[86]: [1, 2, 3, 4]

In [87]:
```
l.append(5)
l
```

Out[87]: [1, 2, 3, 4, 5]

In [88]:
```
l.insert(1,100)
l
```

Out[88]: [1, 100, 2, 3, 4, 5]

In [89]:
```
s = [6,7,8,9]
s
```

Out[89]: [6, 7, 8, 9]

In [90]:
```
l.extend(s)
```

In [92]:
```
l
```

Out[92]: [1, 100, 2, 3, 4, 5, 6, 7, 8, 9]

In [93]:
```
l+s
```

Out[93]: [1, 100, 2, 3, 4, 5, 6, 7, 8, 9, 6, 7, 8, 9]

append もinsert もextendも破壊的な操作

In [95]:
```
l = [1,2,3]
l
```

Out[95]: [1, 2, 3]

In [96]:
```
m = l
```

In [98]:
```
m.append(4)
```

In [99]:
```
m
```

Out[99]: [1, 2, 3, 4]

In [101]:
```
l # lも変わってしまう
```

Out[101]: [1, 2, 3, 4]

リストのコピー

In [102]:
```
m = l[:]
m
```

Out[102]: [1, 2, 3, 4]

In [103]:
```
m.append(5)
```

In [104]:
```
m
```

Out[104]: [1, 2, 3, 4, 5]

In [105]:
```
l
```

Out[105]: [1, 2, 3, 4]

リストのリスト

In [106]:
```
l = [[],[],[]]
l
```

Out[106]:
```
[[], [], []]
```

In [107]:
```
l[1].append(1)
l
```

Out[107]:
```
[[], [1], []]
```

In [109]:
```
l = [i**2 for i in range(1,5)]
l
```

Out[109]:
```
[1, 4, 9, 16]
```

In [110]:
```
m = [[ i*10+j for j in range(5)] for i in range(5)]
m
```

Out[110]:
```
[[0, 1, 2, 3, 4],
 [10, 11, 12, 13, 14],
 [20, 21, 22, 23, 24],
 [30, 31, 32, 33, 34],
 [40, 41, 42, 43, 44]]
```

タプル

In [113]:
```
u =1,
u
```

Out[113]:
```
(1,)
```

In [112]:
```
a= 1,2,3
a
```

Out[112]:
```
(1, 2, 3)
```

シーケンス型（文字列、リスト、タプル）

In [116]:
```
l = [2,4,6]
```

In [117]:
```
for i in l:
    print(i)
```

```
2
4
6
```

In [118]:
```
m = [ x*2 for x in l]
m
```

Out[118]:
```
[4, 8, 12]
```

In [119]:
```
s = "abcd"
```

In [121]:
```
for x in s:
    print(x)
```

```
a
b
c
d
```

In [122]:
```
["*"+x+"*" for x in s]
```

Out[122]:
```
['*a*', '*b*', '*c*', '*d*']
```

In [123]:
```
list(s)
```

Out[123]:
```
['a', 'b', 'c', 'd']
```

In [125]:
```
s = list(s)
s[1]
```

Out[125]:
```
'b'
```

集合（重複を許さず、順番に意味を持たない）

In [126]:
```
a = set() # 空集合
```

In [129]:
```
a.add(1)
a.add(2)
```

In [130]:
```
a
```

Out[130]:
```
{1, 2}
```

In [131]:
```
2 in a
```

Out[131]:
```
True
```

In [132]:
```
5 in a
```

Out[132]:
```
False
```

In [133]:
```
b = {2,3,4}
b
```

Out[133]: {2, 3, 4}

In [137]:
```
a&b #共有集合
```

Out[137]: {2}

In [139]:
```
a-b #差集合
```

Out[139]: {1}

In [140]:
```
a|b #和集合
```

Out[140]: {1, 2, 3, 4}

In [141]:
```
a.add(2)
```

In [142]:
```
a
```

Out[142]: {1, 2}

関数

In [144]:
```
def sum(a,b):
    return a+b
```

In [146]:
```
sum(5,4)
```

Out[146]: 9

In [147]:
```
def add(a,b=100):
    return a+b
```

In [148]:
```
add(50)
```

Out[148]: 150

In [149]:
```
add(50,10)
```

Out[149]: 60

モジュール：関数やクラスを集めたもの

datetimeモジュール

In [150]:
```
import datetime
```

In [153]:
```
d = datetime.date(2019,8,8)
d.year
```

Out[153]: 2019

In [154]:
```
d.month
```

Out[154]: 8

In [155]:
```
d.day
```

Out[155]: 8

In [158]:
```
d.weekday()
```

Out[158]: 3

クラス定義

In [159]:
```
class Person:
    def __init__(self,first_name,last_name):    # インスタンス化時に呼ばれる
        self.first_name = first_name
        self.last_name = last_name
```

In [163]:
```
person1 = Person("kawamoto","tatsunori")
person1.last_name
```

Out[163]: 'tatsunori'

In [164]:
```
person1.first_name
```

Out[164]: 'kawamoto'

In [1]:
```
class Person:
    def __init__(self,first_name,last_name):    # インスタンス化時に呼ばれる
        self.first_name = first_name
        self.last_name = last_name

    def get_name(self):
        return self.first_name + " " + self.last_name

    def __str__(self):
        return self.last_name + " "+ self.first_name

    def __call__(self):
        return self.first_name + " call " + self.last_name
```

```
In [2]: person2 = Person("kawamoto","tatsunori")

In [3]: person2.get_name()
Out[3]: 'kawamoto tatsunori'

In [4]: print(person2)
        tatsunori kawamoto

In [5]: person2()
Out[5]: 'kawamoto call tatsunori'
```

xxは特殊メソッド

- **call** :　関数みたいに返される
- **init** : インスタンス化時に呼び出される
- **str** : print文などに入れられた時に返される

モジュール

```
In [6]: import datetime

In [7]: d = datetime.date(2019,8,9)

In [8]: d.year
Out[8]: 2019
```

・・・のようにしても良いが
dateクラスのみをしたい場合は、モジュールからクラスを指定してインポートする

```
In [9]: from datetime import date

In [10]: d = date(2019,8,3)

In [11]: d.day
Out[11]: 3
```

モジュールの作成

module.pyのファイルを事前に作成

In [191]:
```
mod.py
```

```
---------------------------------------------------------------------
---------
NameError                                 Traceback (most recent c
all last)
~/Desktop/python/MLE/test.py in <module>
----> 1 mod.py

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

ファイル操作（読み込み）

In [23]:
```
f = open("sample.txt")
for line in f:         #1行ずつ処理
 #   line = line.rstrip()    #rstrip:改行文字を削除
    print(line)
f.close()
```

```
rugby

music

tennis
```

In [24]:
```
f = open("sample.txt")
for line in f:         #1行ずつ処理
    line = line.rstrip()    #rstrip:改行文字を削除
    print(line)
f.close()
```

```
rugby
music
tennis
```

- closeメソッドで、リソースを解放
- 例外処理などで、closeメソッドが呼ばれない可能性がある場合は、with構文を使う

In [25]:
```
with open ("sample.txt") as f:
    for line in f:
        line = line.rstrip()
        print(line)
```

```
rugby
music
tennis
```

通常はwith構文を使う

ファイル操作（書き込み）

In [28]:
```
with open ("output.txt","w") as fw:
    with open ("sample.txt") as fr:
        for line in fr:
            print(line, end="",file=fw)
```

print文のendのデフォルトは改行！
なので、改行をなくしている

pickleモジュール

In [29]:
```
from datetime import date
import pickle

x = date(2019,8,10)
x
```

Out[29]:
```
datetime.date(2019, 8, 10)
```

In [31]:
```
with open ("today.pkl","wb") as f:
    pickle.dump(x,f,-1)
```

- バイナリの方が処理が早い為、バイナリとする
- dump関数：書き込みたいオブジェクト、書き込み先のファイルオブジェクト、書き込みフォーマットのバージョンを指定する（-1で最新となる）

In [35]:
```
with open ("today.pkl","rb") as f:
    y = pickle.load(f)
```

In [36]:
```
y
```

Out[36]:
```
datetime.date(2019, 8, 10)
```

その他のファイル形式

csvファイルの取り扱い

In [37]:
```
import csv
```

２列目を足し合わせる

In [42]:
```
s = 0
with open ("sample.csv") as f:
    reader = csv.reader(f)
    next(reader)       # abcd（見出し行）をスキップ：１行進める
    for row in reader:    # 1行ずつ
        s += int(row[1])
print(s)
```

```
18
```

In [43]:
```
import csv

data = [[1,"a",1.1],
            [2,"b",1.2],
            [3,"c",1.3]]

with open ("output.csv","w") as f:
    wr = csv.writer(f)
    for row in data:
        wr.writerow(row)
```

jsonファイル

In [44]:
```
import json
data = {"a":1,"b":"x","c":[1,2,3],"d":{"a":1,"b":2}}

s = json.dumps(data) # dumpsで文字列として書き出す
s
```

Out[44]:
```
'{"a": 1, "b": "x", "c": [1, 2, 3], "d": {"a": 1, "b": 2}}'
```

In [47]:
```
data2 = json.loads(s) # loadsで文字列を読み込む
data2
```

Out[47]:
```
{'a': 1, 'b': 'x', 'c': [1, 2, 3], 'd': {'a': 1, 'b': 2}}
```

例外処理

In [52]:
```
d = {"a":1,"b":2,"c":3}
print(d["d"])
```

```
---------------------------------------------------------------------
---------
KeyError                                  Traceback (most recent c
all last)
<ipython-input-52-0f4213320bad> in <module>
      1 d = {"a":1,"b":2,"c":3}
----> 2 print(d["d"])

KeyError: 'd'
```

In [54]:
```
d = {"a":1,"b":2,"c":3}

try:
    print(d["d"])
except KeyError:
    print("KeyError")
```

```
KeyError
```

In [55]:
```
d = {"a":1,"b":2,"c":3}

try:
    print(d["d"])
except KeyError as err:    #発生したエラーを変数errに格納
    print("KeyError: {}".format(err))
```

```
KeyError: 'd'
```

In [56]:
```
d = {"a":1,"b":2,"c":3}

try:
    print(d["d"])
except:
    print("something wrong")
```

```
something wrong
```

なにも指定しないと全ての例外を受け取る（あまりよくない）

すべての例外を受け取り、かつ変数で受け取る

In [60]:
```
d = {"a":1,"b":2,"c":3}

try:
    print(d["d"])
except Exception as err:
    print(type(err))
    print(err)
```

```
<class 'KeyError'>
'd'
```

In [ ]:

In [ ]:

In [ ]: