○［AI 自習帳 3］　「手書き数字」の、Python プログラムによる処理例

amenbo the 3rd

2018.10.07

◆本稿は［自習 2］の補足版であり、
　掲載する目的は、「手書き数字（文字）データ」の準備方法（例）を解説することです。
　アメンボの手書き数字（文字）の読込みで試した「コードと結果」、および、
　scikit-learn 付属の手書き数字（文字）の表示で試した「コードと結果」を紹介しました。
　（再現性にやや問題のあった例も、そのまま掲載しています）
※本稿では、「svm_03.py」を除いては「Python コード（.py)」としては添付していません、
　他の「コード」記述部では、実行するために必要な最低限の部分のみを掲載しました。
※画像表示例は、Spyder の IPython コンソールに出力されたものです、

目次

１．本稿の目的

・本稿の目的は、任意の手書き数字（文字）をどの様に整形すれば、scikit-learn に付属の手書き数字(文字)セットを教師付き学習用データとして利用できるかを、解説することです。

※文字認識がアメンボの本来の目的ではなく、ＡＩ手法を理解するために調べたため、やや荒っぽい解説になりました事をご容赦ください。

２．Python 用モジュール追加

・本稿で使うライブラリは、

「numpy」		； 数値計算ライブラリ
「scikit-learn」		； AI 用「機械学習ライブラリ」
「Pillow」	(PIL)	； 画像処理ライブラリ
「matplotlib」		； グラフ描画ライブラリ

です。

・ただし、py コード上で呼出すときは、

numpy	⇒	import numpy
scikit-learn	⇒	from sklearn import **
Pillow	⇒	from PIL import **
matplotlib	⇒	from matplotlib import **

と記述します。

・インストールには、下記の「pip（又は pip3)」コマンドを使います。

```
pip install ライブラリ名
```

インストールした結果は、「pip list」で確認出来ます。

３．データの準備

・アメンボは、紙に書いた文字をスキャナーで取り込んで、「tegaki_01.png」として保存しました。
このデータは、読込み処理を行う「py プログラム」と同じフォルダー内に置きます。

４．「手書き数字」を色々な「コード」で読込み表示してみた

※Python コードの詳細は解説しません、専門書や Web 上の情報を参照してください。

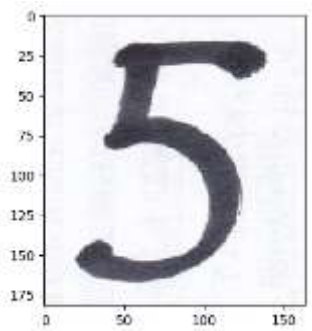
例１； 「手書き文字」の読み込みと描画（その１）

（１）コード；[tegaki_suuzi_yomikomi_01.py]・・このコードのみ、「(.py)」として本稿に添付。

```
# -*- coding: utf-8 -*-
"""
Created on Sun Jul 15 00:05:40 2018
[tegaki_suuzi_yomikomi_01.py]
@author: kenken
"""
import numpy as np
from PIL import Image
from matplotlib import pylab as plt

#i_image=Image.open("tegaki_01.png")
_img=np.array(Image.open("tegaki_01.png"))
#_img=np.asarray(Image.open("tegaki_01.png")) でも同じ結果
plt.imshow(_img)
```

（２）結果

Spyder の IPython コンソールに出力された内容（例）を示します；



③手書き文字の読み込み「手順」記録.docx

――― 以下のコード例では、動作に必要な部分のみを示します ―――

※本稿に添付した「p_code_01_uni.txt」にコード例を記載しました。

コードの記載順はランダムで、実行順ではありません。

（また、一部「再現性」に問題があるものが含まれますが、実行結果をそのまま掲載）

例４； 「手書き文字」の読み込みと描画（その２）

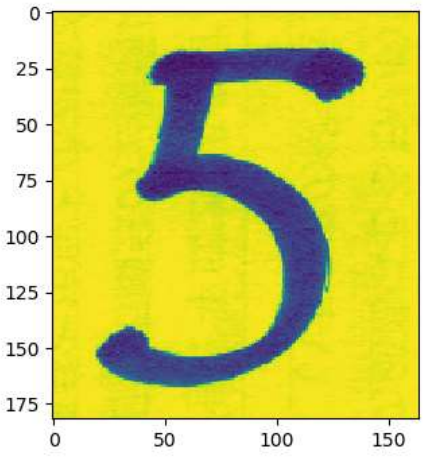
（１）コード；

```
import numpy as np
from PIL import Image
from matplotlib import pylab as plt
print("----------------------------")
_image=Image.open("tegaki_01.png").convert('L')
img=np.asarray(_image)
plt.imshow(img)
plt.show()
```

（２）結果

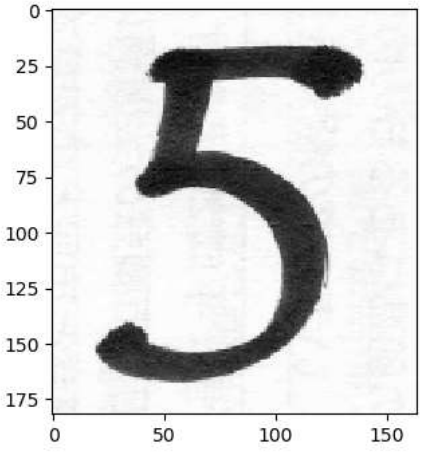
コンソールに出力された内容（例）を示します；

①１回目の実行



①２回目の実行（別のコードの実行後）

？？同じコードで実行したのだが？？



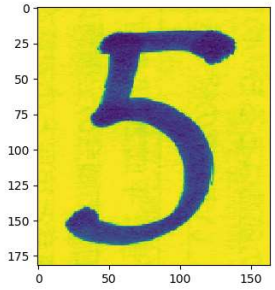
例２； 「手書き文字」の読み込みと描画（その３）

（１）コード；

```
import numpy as np
from PIL import Image
from matplotlib import pylab as plt
print("---------------------------")
_image=Image.open("tegaki_01.png").convert('L')
img=np.asarray(_image)
plt.imshow(img)
```

（２）結果

Spyder の IPython コンソールに出力された内容（例）を示します；



？？？！！！何これ！！！、
「グレースケール」のはずが

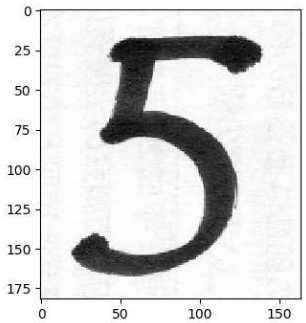
例３； 「手書き文字」の読み込みと描画（その４）

（１）コード；

```
import numpy as np
from PIL import Image
from matplotlib import pylab as plt
#
print("---------------------------")
_image=Image.open("tegaki_01.png").convert('L')
img=np.asarray(_image)
plt.imshow(img)
#
plt.gray()
plt.show()
```

（２）結果

Spyder の IPython コンソールに出力された内容（例）を示します；



５.「scikit-learn」に付属の「digits」データを理解する

※「scikit-learn」に付属の手書き数字（文字）である「digits」データを、色々な「コード」で読込んでから表示してみました。

例１；　「digits」データ（0～9）を適当に表示してみる

（１）コード；

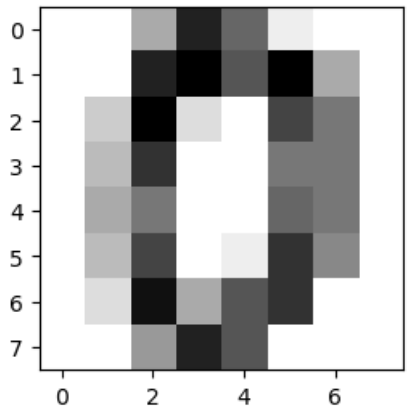
```
#-----------------------------------
import numpy as np
from PIL import Image
from matplotlib import pylab as plt
#-----------------------------------
from sklearn import datasets
digits=datasets.load_digits()
data=digits.images[0]
#data=digits.images[5]
#data=digits.images[7]
#data=digits.images[15]
#data=digits.images[25]
#data=digits.images[30]
plt.figure(figsize=(3,3))
plt.imshow(data,cmap=plt.cm.gray_r)
plt.show()
```

（２）結果

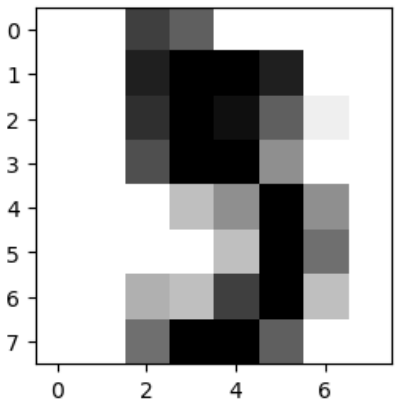
Spyder の IPython コンソールに出力された内容（例）を示します；

「#data=digits.images[＊]」の「#」を、「0」から順番に外して実行してみる

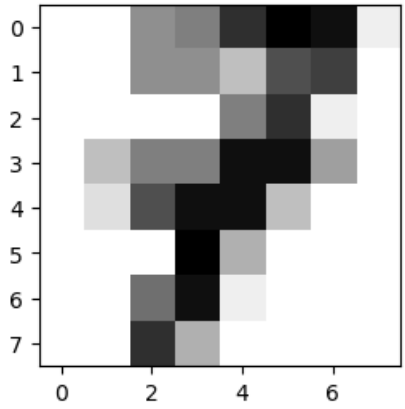
data=digits.images[0]の場合；



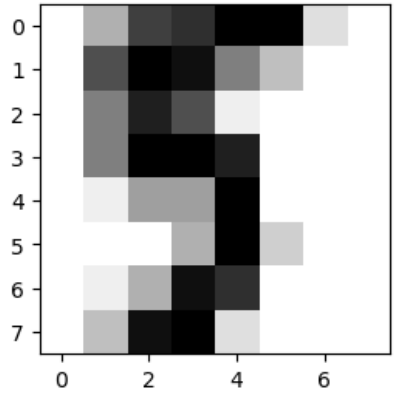
data=digits.images[5]の場合；



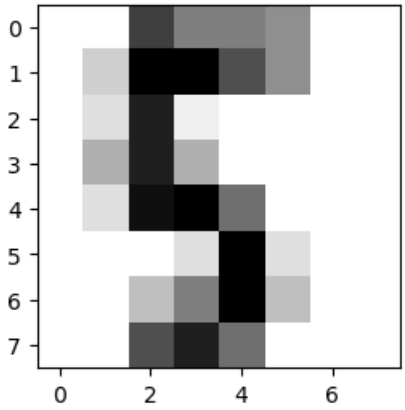
data=digits.images[7]の場合；



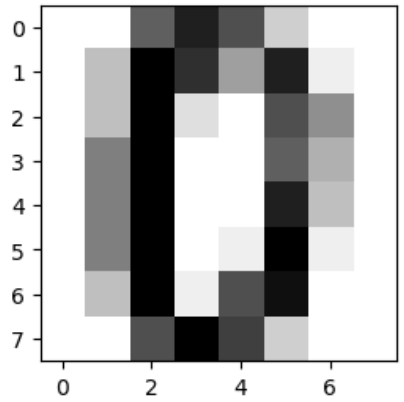
data=digits.images[15]の場合；



data=digits.images[25]の場合；



data=digits.images[30]の場合；

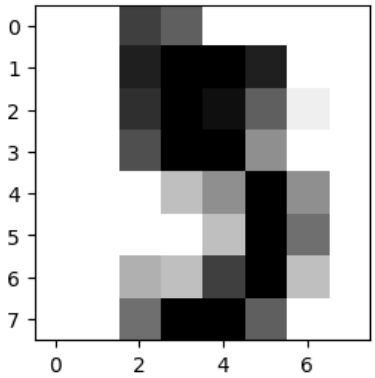


例２； 例３と「digits」データの表示方法を比較する

（１）コード； （再掲）

```
#-----------------------------------
import numpy as np
from PIL import Image
from matplotlib import pylab as plt
#-----------------------------------
from sklearn import datasets
digits=datasets.load_digits()
data=digits.images[5]
plt.figure(figsize=(3,3))
plt.imshow(data,cmap=plt.cm.gray_r)
plt.show()
```

（２）結果



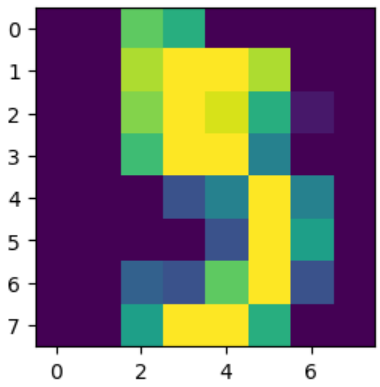
例３；　例２と「digits」データの表示方法を比較する

（１）コード

```
import numpy as np
from PIL import Image
from matplotlib import pylab as plt
#i_image=Image.open("tegaki_01.png")
#-----------------------------------
from sklearn import datasets
digits=datasets.load_digits()
data=digits.images[5]
plt.figure(figsize=(3,3))
#plt.imshow(data,cmap=plt.cm.gray_r)
plt.imshow(data)
plt.show()
```

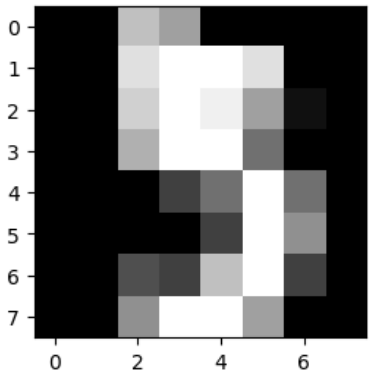
（２）結果

実行１回目



！！わーお、
なんだこれは

実行２回目 以降



これが
正解？？

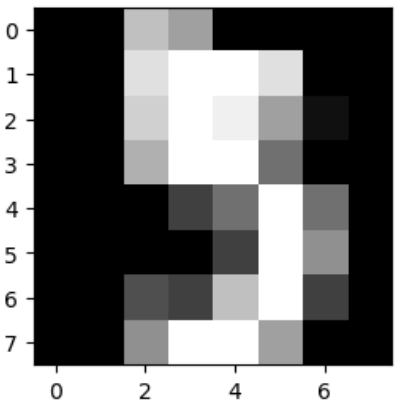
例４；　コードにチョット追加

（１）コード

```
import numpy as np
from PIL import Image
from matplotlib import pylab as plt
#i_image=Image.open("tegaki_01.png")
#-----------------------------------
from sklearn import datasets
digits=datasets.load_digits()
data=digits.images[5]
plt.figure(figsize=(3,3))
#plt.imshow(data,cmap=plt.cm.gray_r)
plt.imshow(data)
plt.gray()
plt.show()
```

（２）結果

再現性あり（１回目、２回目も同じ）



③手書き文字の読み込み「手順」記録.docx

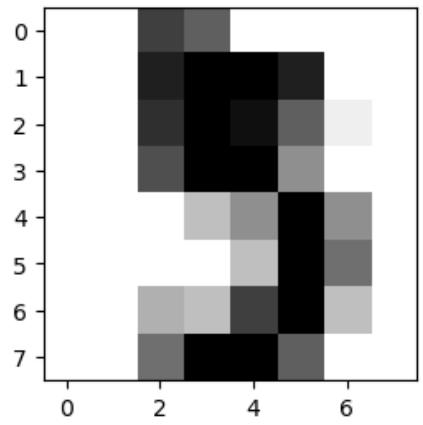
例５； 「digits」データの記録形式を調べる

（１）コード

```
import numpy as np
from PIL import Image
from matplotlib import pylab as plt
#---------------------------------
from sklearn import datasets
digits=datasets.load_digits()
data=digits.images[5]
plt.figure(figsize=(3,3))
plt.imshow(data,cmap=plt.cm.gray_r)
plt.show()
print("---------------------------")
print(data)
#data.tofile("data.csv",sep="",format="%.0f")#NG
np.savetxt("data.csv",data,delimiter=',',  fmt='%d')
```

（２）－１　結果１

IPython コンソールへの出力

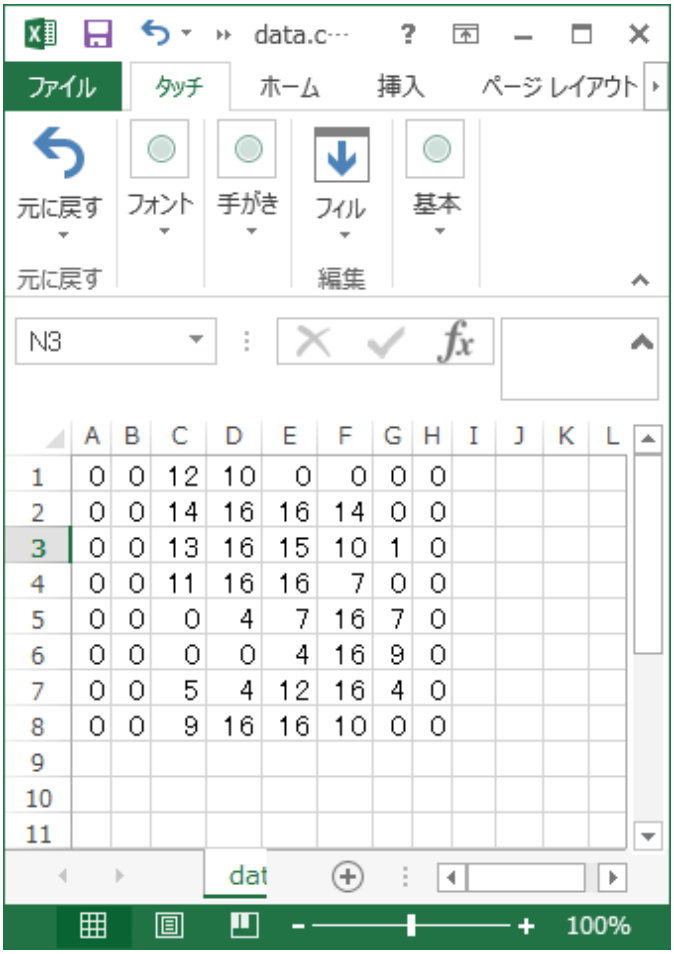


```
---------------------------
[[ 0.  0. 12. ...  0.  0.  0.]
 [ 0.  0. 14. ... 14.  0.  0.]
 [ 0.  0. 13. ... 10.  1.  0.]
 ...
 [ 0.  0.  0. ... 16.  9.  0.]
 [ 0.  0.  5. ... 16.  4.  0.]
 [ 0.  0.  9. ... 10.  0.  0.]]
```

※［８×８］ピクセル、
　色深度４ビット（０～１６）

（２）－２　結果２

（エクセル）CSV ファイル「data.csv」への出力



	A	B	C	D	E	F	G	H
1	0	0	12	10	0	0	0	0
2	0	0	14	16	16	14	0	0
3	0	0	13	16	15	10	1	0
4	0	0	11	16	16	7	0	0
5	0	0	0	4	7	16	7	0
6	0	0	0	0	4	16	9	0
7	0	0	5	4	12	16	4	0
8	0	0	9	16	16	10	0	0
9								
10								
11								

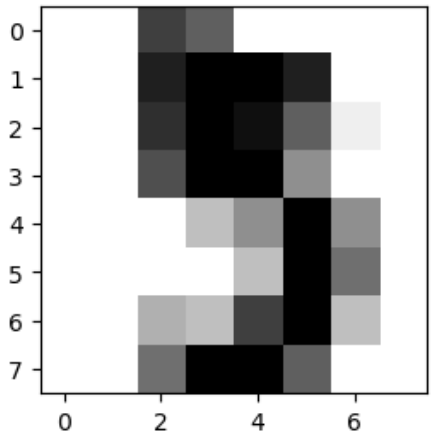
例６； 「digits」の「５」データと、アメンボの手書き数字「５」データを比較する

（１）コード

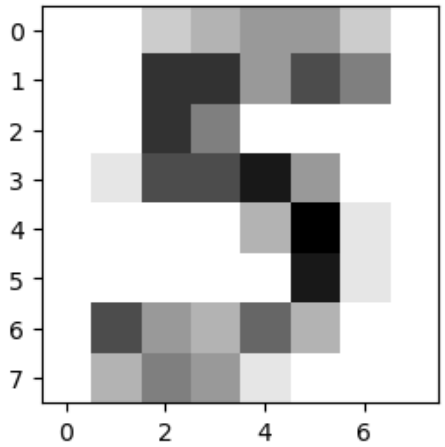
```
import numpy as np
from PIL import Image
from matplotlib import pylab as plt

#----------------------------------

from sklearn import datasets
digits=datasets.load_digits()
data=digits.images[5]
plt.figure(figsize=(3,3))
plt.imshow(data,cmap=plt.cm.gray_r)
#plt.imshow(data)
#plt.gray()
plt.show()
print("---------------------------")
print(data)
#data.tofile("data.csv",sep="",format="%.0f")#NG
np.savetxt("data.csv",data,delimiter=',', fmt='%d')

print("++++++++++++++++++++++++++++++++++")

_image=Image.open("tegaki_01.png").convert('L')
_image=_image.resize((8,8),Image.LANCZOS)
img=np.asarray(_image,dtype=float)
img=np.floor(16-16*(img/256))
np.savetxt("data_tegaki.csv",img,delimiter=',', fmt='%d')
#画像の表示
plt.figure(figsize=(3,3))
#plt.imshow(img)
plt.imshow(img,cmap=plt.cm.gray_r)
plt.show()
print("---------------------------")
print(img)
```

（2）-1　結果１

IPython コンソールへの出力



```
---------------------------

[[ 0.   0.  12. ...  0.   0.   0. ]
 [ 0.   0.  14. ... 14.   0.   0. ]
 [ 0.   0.  13. ... 10.   1.   0. ]
 ...
 [ 0.   0.   0. ... 16.   9.   0. ]
 [ 0.   0.   5. ... 16.   4.   0. ]
 [ 0.   0.   9. ... 10.   0.   0. ]]
++++++++++++++++++++++++++++++++++
```

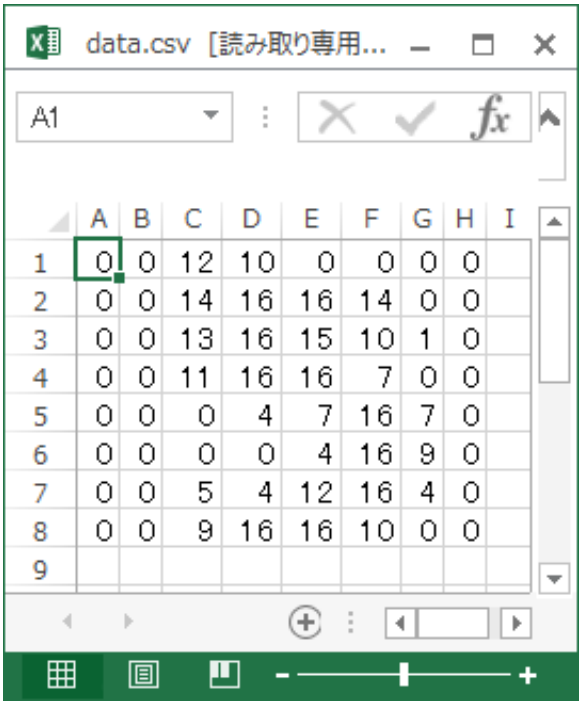


```
---------------------------

[[0.  0.  2. ... 4.  2.  0. ]
 [0.  0.  8. ... 7.  5.  0. ]
 [0.  0.  8. ... 0.  0.  0. ]
 ...
 [0.  0.  0. ... 9.  1.  0. ]
 [0.  7.  4. ... 3.  0.  0. ]
 [0.  3.  5. ... 0.  0.  0. ]]
```

（2）-2　結果２

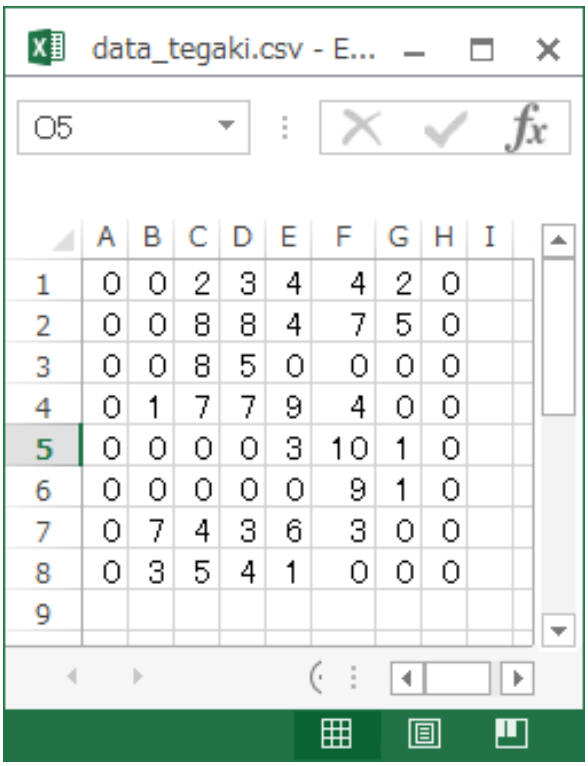
（エクセル）CSV ファイルへの出力

「data.csv」



	A	B	C	D	E	F	G	H	I
1	0	0	12	10	0	0	0	0	
2	0	0	14	16	16	14	0	0	
3	0	0	13	16	15	10	1	0	
4	0	0	11	16	16	7	0	0	
5	0	0	0	4	7	16	7	0	
6	0	0	0	0	4	16	9	0	
7	0	0	5	4	12	16	4	0	
8	0	0	9	16	16	10	0	0	
9									

「data_tegaki.csv」



	A	B	C	D	E	F	G	H	I
1	0	0	2	3	4	4	2	0	
2	0	0	8	8	4	7	5	0	
3	0	0	8	5	0	0	0	0	
4	0	1	7	7	9	4	0	0	
5	0	0	0	0	3	10	1	0	
6	0	0	0	0	0	9	1	0	
7	0	7	4	3	6	3	0	0	
8	0	3	5	4	1	0	0	0	
9									

例７； 「digits」の「５」データが格納されている形式を確認する

（１）コード

```
import numpy as np
from PIL import Image
from matplotlib import pylab as plt
#---------------------------------
from sklearn import datasets
digits=datasets.load_digits()
data=digits.images[5]
print(data)
np.savetxt("data.csv",data,delimiter=',', fmt='%d')
print("---------------------------")
print(digits.data[5])
np.savetxt("data_5.csv",digits.data[5],delimiter=',', fmt='%d')
```

（２）-１　結果１

IPython コンソールへの出力

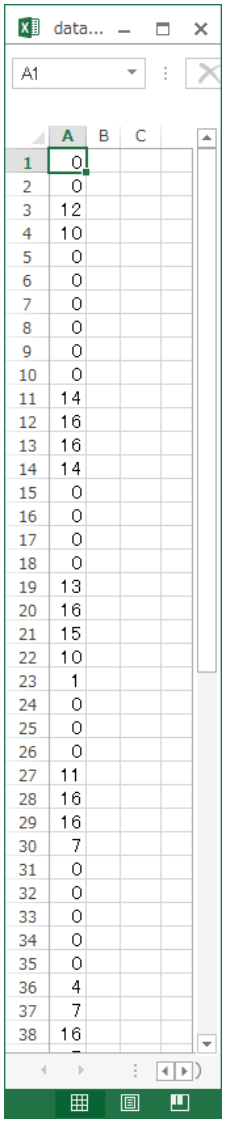
```
[[ 0.   0.  12. ...  0.   0.   0.]
 [ 0.   0.  14. ... 14.   0.   0.]
 [ 0.   0.  13. ... 10.   1.   0.]
 ...
 [ 0.   0.   0. ... 16.   9.   0.]
 [ 0.   0.   5. ... 16.   4.   0.]
 [ 0.   0.   9. ... 10.   0.   0.]]
---------------------------
[ 0.   0.  12. ... 10.   0.   0.]
```

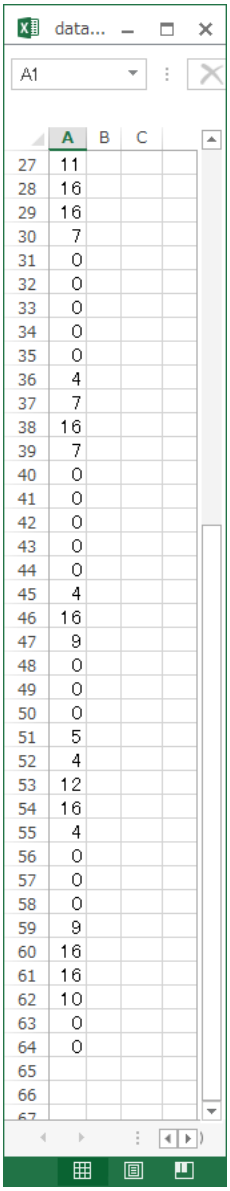
[64 行、1 列]の配列形式で
データが格納されている。

※[1画像]＝8×8＝64ピクセル

（２）-２　結果２

CSV ファイル「data_5.csv」への出力





例８； 「digits」の「５」データと、加工後のアメンボの手書き数字「５」データが、
同じデータ形式で格納されていることを確認する

（1）コード

```
import numpy as np
from PIL import Image
from matplotlib import pylab as plt
#i_image=Image.open("tegaki_01.png")
#-----------------------------------

from sklearn import datasets
digits=datasets.load_digits()
data=digits.images[5]
print(data)
np.savetxt("data.csv",data,delimiter=',', fmt='%d')

print("---------------------------")

print(digits.data[5])
np.savetxt("data_5.csv",digits.data[5],delimiter=',', fmt='%d')

print("+++++++++++++++++++++++++++++++++")

#  画像加工後に表示する
_image=Image.open("tegaki_01.png").convert('L')
_image=_image.resize((8,8),Image.LANCZOS)
img=np.asarray(_image,dtype=float)
img=np.floor(16-16*(img/256))
np.savetxt("data_tegaki.csv",img,delimiter=',', fmt='%d')
print(img)
print("---------------------------")
img=img.flatten()
print(img)
np.savetxt("data_tegaki_5.csv",img,delimiter=',', fmt='%d')
```

（2）-1　結果１

IPython コンソールへの出力

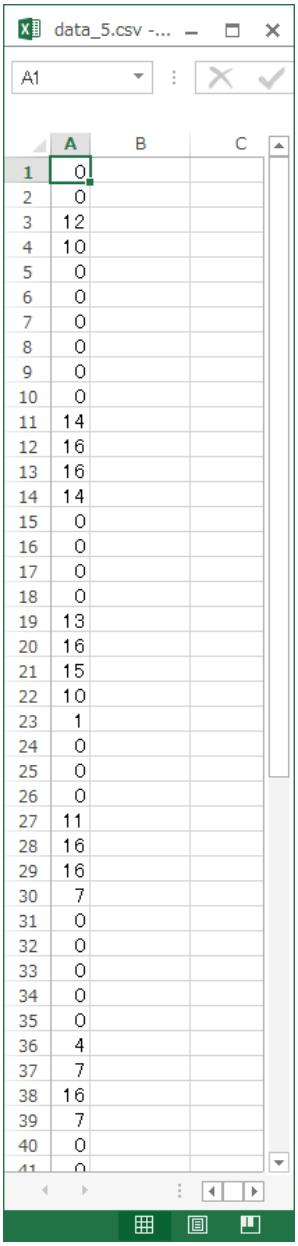
```
[[ 0.   0.  12. ...  0.   0.   0. ]
 [ 0.   0.  14. ... 14.   0.   0. ]
 [ 0.   0.  13. ... 10.   1.   0. ]
 ...                -------digita データ------
 [ 0.   0.   0. ... 16.   9.   0. ]
 [ 0.   0.   5. ... 16.   4.   0. ]
 [ 0.   0.   9. ... 10.   0.   0. ]]
--------------------------
[ 0.   0.  12. ... 10.   0.   0. ]
++++++++++++++++++++++++++++++++
[[0.  0.  2. ... 4.  2.  0. ]
 [0.  0.  8. ... 7.  5.  0. ]
 [0.  0.  8. ... 0.  0.  0. ]
 ...               -------アメンボのデータ------
 [0.  0.  0. ... 9.  1.  0. ]
 [0.  7.  4. ... 3.  0.  0. ]
 [0.  3.  5. ... 0.  0.  0. ]]
--------------------------
[0.  0.  2. ... 0.  0.  0. ]
```

アメンボの手書き数字も、
「digits」と同じ様に
[64 行、1 列]の配列形式で
格納できたことを確認。

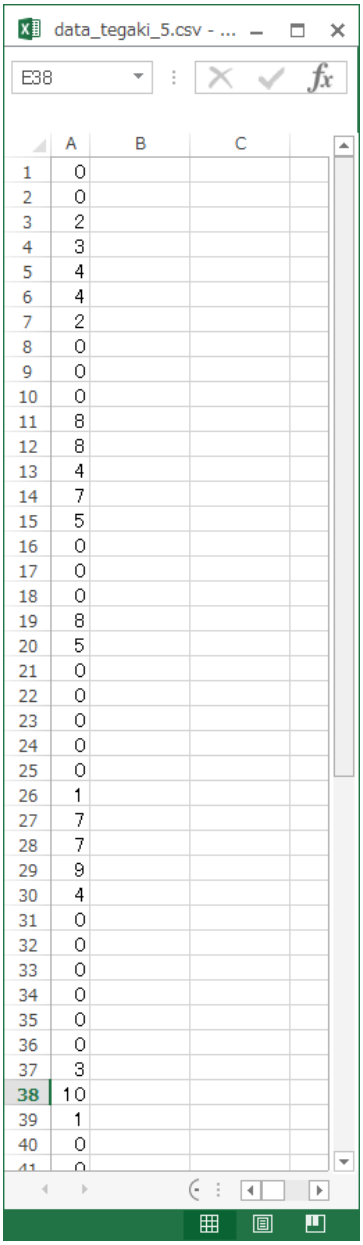
（2）-2　結果２

（エクセル）CSV ファイルへの出力

「data_5.csv」
digits の手書き数字



「data_tegaki_5.csv」
アメンボの手書き数字



例９； 「digits」の「５」「２５」番目には、手書き数字「５」が保存されていることを再確認

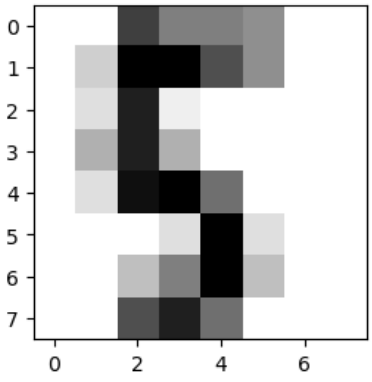
（１）コード

```
import numpy as np
from PIL import Image
from matplotlib import pylab as plt

#---------------------------------

from sklearn import datasets
digits=datasets.load_digits()
#data=digits.images[5]
data=digits.images[25]
print(data)
np.savetxt("data.csv",data,delimiter=',',  fmt='%d')

plt.figure(figsize=(3,3))
plt.imshow(data,cmap=plt.cm.gray_r)
plt.show()
print("----------------------------")
```

（２）-１　結果１

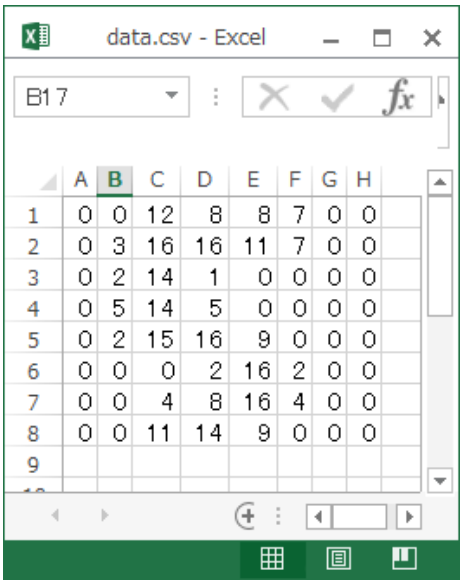
IPython コンソールへの出力

```
[[ 0.  0. 12. ...  7.  0.  0.]
 [ 0.  3. 16. ...  7.  0.  0.]
 [ 0.  2. 14. ...  0.  0.  0.]
 ...
 [ 0.  0.  0. ...  2.  0.  0.]
 [ 0.  0.  4. ...  4.  0.  0.]
 [ 0.  0. 11. ...  0.  0.  0.]]
```



（２）-２　結果２

（エクセル）CSV ファイル「data.csv」への出力



※再確認;
この結果からも、
「5」(「15」)「25」・・・・・に
「5 のデータ」が入っていることが判る。

６．纏めと補足

（１）「scikit-learn」に付属の「digits」データの構造

※「１つ」の手書き数字データは、［64 行、１列］の配列形式で収納されています。

digit.data[*]	0	1	2			9	10	11	12			19	20	21			30	31			100	
収納手書き数字１数字は、［64 行、１列］で表現	0	1	2			9	10	1	2			9	0	1			0	1			0	

※「0～9」の手書き数字が、繰り返し収納されています、
従って、手書き数字データは［64 行、＊列］にまとめられています。

※アメンボの手書き数字のデータも、識別を可能にするためには、プログラム実行時に同一の形式に変換しておく必要があります。

（２）モノクロ（グレースケール）画像の表現

※色深度（ピクセルごとのビット数；**bits per pixel**（bpp））がどの様な場合でも、
「0」＝黒、「最大値」＝白
で、表現されます。

色深度	黒	白
16 ビット	0	65, 535
8 ビット	0	255
4 ビット	0	16

※「scikit-learn」に付属の「digits」データは「色深度＝4 ビット」で、
「アメンボの手書き数字」をスキャニングしたデータでは「色深度＝8 ビット」でした。

◆そこで、数字を識別するプログラムでは、
まず、「アメンボの手書き数字」を「digits」データと同じ
・「色深度＝4 ビット」に
・８×８ピクセル表示に
変換してから実行することになります。

以　上