デジタルメディア処理1

担当: 井尻 敬

提出方法:『**dm1学籍番号**』というフォルダを作成し，その中にソースコードを書いたファイルを入れてzip圧縮してscombより提出．フォルダ名は全て半角．

フォルダ名の例: dm2AL150999

zipファイル名の例: dm2AL150999.zip

課題雛形: http://takashiijiri.com/classes/dm2019_1/dm1exer_firsthalf.zip

入出力 ： 課題では入力画像を受け取り，画像またはファイルを保存するプログラムを作る．入出力ファイル名は，以下の例のようにコマンドライン引数より与えるものとする．（※各課題の指定に従うこと）

```
$python  exer*.py  fname_in.png  fname_out.png
```

注意 ：

採点は自動化されています．フォルダ名・ファイル名やプログラムの仕様は指示に厳密に従ってください．入出力の仕様を満たさないコードは評価できず0点扱いとなることがあります．

今回は計算速度を重視しませんが，60秒以上の計算時間がかかるものは，自動採点の都合上0点とします．

各課題について，入出力例を用意するので，作成したプログラムのテストに利用してください．

締め切り

課題番号	締め切り
前半：課題01~11	**11/29 23:59**
後半：課題12~19	**12/20 23:59**

※ 発展問題 (課題18, 19)を除いて全問正解で合計 約50点とする予定

※ 配点は1問3点程度にする予定

注意

- **この課題は，知人同士で相談しながら取り組んでよいです**
  - 教える立場の方は理解が補強でき，教わる方は難しい課題も解けるようになり，メリットは大きいです
  - ただし，教わる人は他人のコードをコピーするのではなく，なぜそのように記載すべきかを学び，自身でコードを書いてください．
  - **明らかなコピーが発見された場合は，プログラミング課題とミニッツペーパーの得点を0点とするなどの対応をします．**
- **この課題の解答となるコードを，『この課題の解答と分かる形』でWeb上（GitHub，SNS，個人web page)に公開することは避けてください**
- **知恵袋やteratailなどのナリッジコミュニティサイトにて，問題文をそのまま掲載し，解答を得ることは行なわないでください**
  - 上記のような活動を井尻が発見した場合は，しかるべき処理をとります
  - 分からない部分がある場合は『どこがどう分からないかを自分の言葉で明確に説明し』他者から知識を受け取ってください
- **プログラミングが不得意な方：**課題を読むと難しく感じるかもしれませんが，各課題のひな形にヒントを多く記載してあります．まずはそこを読んでみてください．
- **プログラミング得意な方:** それなりに歯応えのあるものを用意しようと思ってはいるのですが，まだ簡単なものも多いです．簡単すぎたらすみません。。

演習1 Python入門

- exer1 コマンドライン引数とprint文	very easy
- exer2 FileIO, for文とif文	very easy
- exer3 FizzBuzz	very easy
- exer4 硬貨カウント	easy
- exer5 画像データの2値化	easy
- exer6 モザイク画像	normal
演習2 線形・非線形フィルタ	
- exer7 色変換	very easy
- exer8 ガウシアンフィルタ	normal
- exer9 ソーベルフィルタ	normal
- exer10 メディアンフィルタ	normal
- exer11 勾配強度画像作成	normal
演習3 ハーフトーン処理	
- exer12 ティザ法	normal
- exer13 濃度パターン法	normal
- exer14 誤差拡散法	hard
演習4 フーリエ変換	
- exer15 フーリエ変換1D	normal
- exer16 逆フーリエ変換1D	normal
- exer17 フーリエ変換2D	hard?
演習5 発展課題	
- exer18 迷路	hard
- exer19 名簿管理	normal

1回 pythonの基本と画像の入出力 課題1~6

課題1. 標準出力 - 雛形 exer1.py

コマンドライン引数から2つの整数を受け取り，その積の回数だけ『hello, world』と標準出力に表示するプログラムを作成せよ

雛形(exer1.py)に途中まで作製したコードがあるので参考にすること

- 実行コマンドと実行例は以下の通り

```
$ python exer1.py 1 4
hello, world
hello, world
hello, world
hello, world
```

```
$ python exer1.py 2 3
hello, world
hello, world
hello, world
hello, world
hello, world
hello, world
```

・自動採点の都合上，関係ない文字列が標準出力に出力されている場合不正解になるので注意すること

課題2. 標準出力 - 雛形 exer2.py

***1*からN*までの整数を順番に標準出力に表示するプログラムを作成せよ．ただし，以下の仕様を満たすこと．**

- 表示する数字が3の倍数の時には数字ではなく ”hoge” と表示する
- 表示する数字が4の倍数の時には 数字ではなく”fuga” と表示する
- 表示する数字が，3と4，両方の倍数の時には ”hogefuga”と表示する
- 最大の数 *N* は，コマンドライン引数より与える

実行コマンドは以下の通り．

```
$ python exer2.py N
```

課題3. ファイル入力と配列 - 雛形 exer3.py

数値データをファイルから読み込み，その最大値・最小値をファイルに出力するプログラムを作成せよ

- 入力ファイル名，出力ファイル名は，コマンドライン引数より与えること
- 入力ファイルには１行に１つの実数値が記載されているとする
- 出力ファイルの一行目に，最大値と最小値を記載する
- 最大値と最小値の間には，半角スペースを一つだけ書くこととする

実行コマンドは以下の通り (file_in.txt, file_out.txt は，適当なファイル名)

```
$ python   exer3.py   file_in.txt  file_out.txt
```

ファイル入出力については雛形を参照

課題4. 画像のグレースケール化 - 雛形 exer4.py

画像データを読み込み，グレースケール化して保存せよ

- 入力ファイル名，出力ファイル名は，コマンドライン引数より与えられることとする
- グレースケール値は，赤・緑・青チャンネルの平均を利用することとする

  $I = (r+g+b)/3$

実行コマンドは以下の通り．(file_in.txt, file_out.txt は，適当なファイル名)

```
$ python   exer4.py   file_in.png  file_out.png
```

画像の入出力・画素値へのアクセス方法は雛形を参照

課題5. 画像の2値化 - 雛形 exer5.py

画像データを読み込み，グレースケール化した後，与えられた閾値により二値化し，その画像を保存せよ

- 入力ファイル名，出力ファイル名，閾値は，コマンドライン引数により与えられるとする
- 画素値が閾値**以上**なら，その画素値を255にする
- 画素値が閾値より小さいなら，その画素値を 0 とする

実行コマンドは以下の通り．(file_in.txt/file_out.txt は適当なファイル名，threshは閾値)

```
$ python   exer5.py   file_in.png  thresh  file_out.png
```

画像の入出力・画素値へのアクセス方法は雛形を参照

課題6. モザイク画像の作成（exer6.py）

カラー画像を読み込み，モザイク画像を作成するプログラムを作成せよ

- ファイル名はexer6.pyとし，ファイル名・モザイクサイズ*R*を以下の通りコマンドライン引数として取得せよ

```
$python   exer6.py   fname_in.png  R  fname_out.png
```

- モザイク画像生成の際，画像を*R* x *R*のブロックに分割し，各ブロックをその平均画素値でべた塗りすることとする
- 画像の右端・下端に割り切れないブロック（サイズが*R*に満たない領域）が発生する場合も，その領域を平均画素値で塗ること

※ヒント：スライスを活用してください

※ヒント：スライスにより配列の範囲外を指定すると，エラーにはならず無視されます

課題1～6の実行例

- 課題どおりのプログラムを作成し実行した結果得られる出力ファイルを雛形フォルダに入れておきます.
- このファイルは上書きせず，自身で作成したプログラムの出力と見比べることでデバッグに利用してください.

実行例

課題2

```
$ python  exer2.py  20
1
2
hoge
fuga
5
hoge
7
fuga
hoge
10
11
hogefuga
13
14
hoge
fuga
17
hoge
19
fuga
```

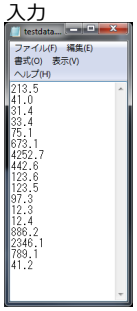
課題3

雛形フォルダ内のtestdata.txtに対して以下のコマンドを実行すると，out_exer3.txtファイルが出力されます

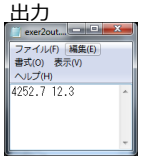
実行コマンド

```
$ python  exer3.py  testdata.txt  out_exer3.txt
```

入力



出力



実行例

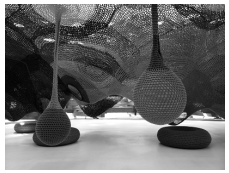
課題4

雛形フォルダ内のimg1.pngに対して以下のコマンドを実行すると，img_gray.pngが出力される

実行コマンド

```
$ python  exer4.py  img1.png img1_gray.png
```

img1.png
入力



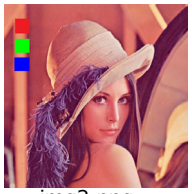
img1_gray.png
出力

課題5

雛形フォルダ内のimg2.pngに対して閾値を150として実行すると，img_bin.pngが出力される

実行コマンド

```
$ python  exer5.py  img2.png  150  img2_bin.png
```



img2.png
入力

img2_bin2.png
出力

実行例

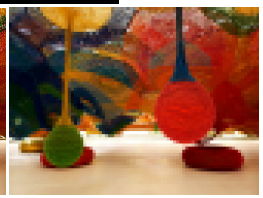
課題6

雛形フォルダ内のimg1.pngに対して以下のコマンドを実行すると，img_mo*.pngが出力される

実行コマンド

```
$ python  exer6.py  img1.png  5  img_mo5.png
```

img.png
入力



img_mo5.png
出力

実行コマンド

```
$ python  exer6.py  img1.png  15  img_mo15.png
```

img.png
入力

img_mo15.png
出力

フィルタ処理
課題7~11

課題7. 色の変換（exer7.py）

カラー画像を読み込み，画像の赤と緑チャンネルを入れ替えた画像を保存するプログラムを作成せよ

ファイル名はexer7.pyとし，実行コマンドは以下の通りとする

```
$python   exer7.py   file_in.png   fname_out.png
```

- file_in.png, file_out.txt は，入出力ファイル名

課題8. ガウシアンフィルタ（exer8.py）

画像を読み込み，グレースケール画像に変換後，ガウシアンフィルタを掛けた画像を保存するプログラムを作成せよ

ファイル名は exer8.py とし，実行コマンドは以下の通り

```
$python   exer8.py   fname_in.png   fname_out.png
```

- グレースケール化の方法についてはひな形を参照
- 右図のガウシアンフィルタを利用すること
- 画像の周囲1pixelは計算せず0を入れること
- 今回はプログラミング練習が目的なので，次の関数『cv2.filter2D() / cv2.GaussianBlur() / cv2.Sobel() / np.convolve() 』は利用しないこと

1/16	2/16	1/16
2/16	4/16	2/16
1/16	2/16	1/16

フィルタの係数

課題9. ソーベルフィルタ（exer9.py）

画像を読み込み，グレースケール画像に変換後，縦ソーベルフィルタを掛けた画像を保存するプログラムを作成せよ

- ファイル名は exer9.py とし，実行コマンドは以下の通り

```
$python   exer9.py   fname_in.png   fname_out.png
```

1	2	1
0	0	0
-1	-2	-1

- 画像の周囲1pixelは計算せず0を入れること
- フィルタ適用後，負値となる画素は －1 倍して正値に変換すること
- フィルタ適用後，値が255を超える画素には255を代入すること
- 今回はプログラミング練習が目的なので，次の関数『cv2.filter2D() / cv2.GaussianBlur() / cv2.Sobel() / np.convolve()』は利用しないこと

課題10. 勾配強度画像の作成（exer10.py）

画像を読み込み，グレースケール画像に変換後，勾配強度画像を計算し保存するプログラムを作成せよ

ファイル名は exer10.py とし，実行コマンドは以下の通りとする

```
$python  exer10.py  fname_in.png  fname_out.png
```

-1	0	1
-2	0	2
-1	0	1

横方向
ソーベルフィルタ

1	2	1
0	0	0
-1	-2	-1

縦方向
ソーベルフィルタ

- 画像の周囲1pixelは計算せず0を入れること
- ある画素の勾配強度は $I = \sqrt{f_x^2 + f_y^2}$ とする．ただし，f_xとf_yはそれぞれ横方向・縦方向のソーベルフィルタの応答である
- 勾配強度を計算後、画素値が255を越えていたら255を代入すること

※今回はプログラミング練習が目的なので，次の関数『cv2.filter2D() / cv2.GaussianBlur() / cv2.Sobel() / np.convolve()』は利用しないこと

課題11. メディアンフィルタ（exer11.py）

画像を読み込み，グレースケール画像に変換後，メディアンフィルタを掛けた画像を保存するプログラムを作成せよ

- ファイル名は exer11.py とし，実行コマンドは以下の通り

```
$python  exer11.py  fname_in.png  fname_out.png
```

- np.arrayには，配列の平均・分散・中央値を求める関数があるので利用方法を調べて活用するとよい（自分で計算してもよい）
- フィルタサイズは**5x5**とする
- 画像の周囲2pixelは計算せず0を入れること
- 今回はプログラミング練習が目的なので，cv2の関数『cv2.medianBlur』は利用しないこと

課題7～11の実行例

- 課題どおりのプログラムを作成し実行した結果得られる出力ファイルを雛形フォルダに入れておきます．
- このファイルは上書きせず，自身で作成したプログラムの出力と見比べることでデバッグに利用してください．

実行例 課題7

```
$python  exer7.py  img1.png  img1_flipRG.png
```

入力画像
img1.png

出力画像
img1_flipRG.png

```
$python  exer7.py  img2.png  img2_flipRG.png
```

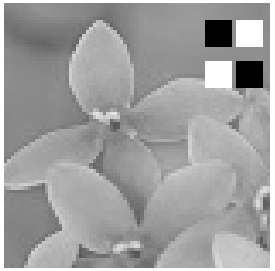
入力画像
img2.png

出力画像
img2_flipRG.png

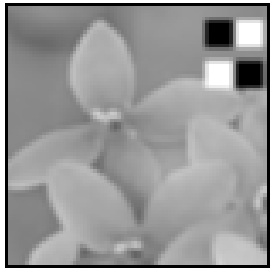
参考 (出力結果の例)

課題8

```
$python  exer8.py  img3.png  img3_gauss.png
```



入力画像
img3.png



出力画像
img3_gauss.png

参考 (出力結果の例)

課題9

```
$python  exer9.py  img3.png  img3_sobel.png
```

入力画像
img3.png

出力画像
img3_sobel.png

課題10

```
$python  exer10.py  img3.png  img3_gm.png
```

入力画像
img3.png

出力画像
img3_gm.png

参考 (出力結果の例)

課題11

```
$python  exer10.py  img4.png  img4_median.png
```

入力画像
img4.png

出力画像
img4_median.png

Salt and pepper noiseのようなノイズに対して
Median filterは堅固に動きます。
この例では画像解像度が低いので、
5x5の窓は大きすぎるかもしれないですね。。

※画像は学生がタイの動物園（gPBL)で撮影したものです