デジタルメディア処理

担当: 井尻 敬

提出方法:

1. 『**dm_学籍番号**』というフォルダを作成
2. その中にソースコードを書いたファイルを入れてzip圧縮する
3. Scombの課題より提出．フォルダ名は全て半角．

   フォルダ名の例: dm_AL200999

   zipファイル名の例: dm_AL200999.zip

課題雛形: http://takashiijiri.com/classes/dm2021/
入出力　: 各課題について詳細な仕様が定義されるので正しく従うこと

注意 :

- 採点自動化のため，フォルダ名・ファイル名が間違っているもの，入出力の仕様を満たさないコードは，評価できず0点扱いとなります．
- この課題では計算速度を重視しませんが，評価用入力データに対して20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01~12	**12/20 23:59**
後半：	課題13~20	**1/21 23:59**

※ 発展問題を除いて全問正解で合計 約50点とする予定

※ 配点は未定

課題の実施方法

…魚を与えるのではなく、魚の釣り方を教えよ…

- この課題は，知人同士で相談しながら行ってよいです
- 教える立場の方は教えることで理解が補強でき，教えられる方は比較的難しい課題も解けるようになり，メリットは大きいです

- 教わる人は他人のコードをコピーするのではなく，どのような処理を行なうかを議論し，ソースコードは自身で作成してください
- 画面共有を行う場合は，正解を表示する行為は避けてください.
  - 教わる側の画面を共有しコメントをするようにしてください
  - 両者が正解を出している場合にコードを確認することは問題ありません

注意

- 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上に公開する事は避けてください（GitHub，SNS，個人web page）
  - これを許してしまうと，発見し次第課題を差し替える事になり，数年後には難解な課題のみが残ってしまうので。。。
- 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，解答を得る事は行なわないでください
  - 上記のような活動を発見した場合は，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
- ソースコードのコピーが発見された場合には，コピー元・コピー先の両名ともカンニングと同様の処置をとります※Pythonはコードがどうしても似てしまうことは理解しています．していないカンニングの疑いをかけられないかと不安になったり、あえて”へんな書き方”をする必要はないです

演習の実施方法

- 演習時間中はzoom利用とslackとPC室を併用
  - 50個のブレークアウトルームを自由に利用してよい
  - 知人同士で教えあいながら演習を進めてよい
- 質問がある場合
  - ブレークアウトルームへ移動し，SlackでTAを呼ぶ（部屋番号伝える）
    →TAは順番に訪問する
  - Slackへ直接質問を書き込んでもOK
  - PC室で教員に質問してもOK
- 約束
  - 正解コードを共有しないこと（mail/zoom画面共有など）
  - できていないコードを共有するのはOK
  - できている人同士なら画面共有してもOK

pythonの基本

- exer1 Hello world *
- exer2 FizzBuzz *
- exer3 ファイル *
- exer4 ファイル **

画像入出力

- exer5 grayscale化 **
- exer6 二値化 ***
- exer7 モザイク ***

フィルタ処理

- exer8 色変換 *
- exer9 ガウシアンフィルタ ***
- exer10 ソーベルフィルタ ***
- exer11 勾配強度画像作成 ***
- exer12 メディアンフィルタ ***

ハーフトーン処理

- exer13 ディザ法 ***
- exer14 濃度パターン法 ***
- exer15 誤差拡散法 ****

フーリエ変換処理

- exer16 フーリエ変換 ***
- exer17 逆フーリエ変換 ***
- exer18 2Dフーリエ変換 ****

発展課題

- exer19 迷路 *****
- exer20 Deconvolution *****

課題1~4
標準入出力，ファイル入出力

課題1 標準出力 (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

実行コマンドは以下の通り（aとbは非負整数値）

```
$python  exer1.py  a  b
```

すべての課題にひな形 exerX.pyが用意してあります.

参考にしてください.

課題2. 標準出力 – (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』かつ『5の倍数でない』時には，数字ではなく ”hoge” と表示する
- 表示する数字が『5の倍数』かつ『3の倍数でない』時には，数字ではなく”fuga” と表示する
- 表示する数字が『3の倍数』かつ『5の倍数』の時には，数字ではなく”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
```

実行例は右の通り

```
$ python   exer2.py  16
1
2
hoge
4
fuga
hoge
7
8
hoge
fuga
11
hoge
13
14
hogefuga
16
```

課題3. ファイル入力と配列 – (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・平均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，最小値，平均値 を記載する
- 最大値，最小値，平均値の間には，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. ファイル入力と配列 - 雛形 exer4.py

数値データをファイルから読み込み，大きい順に3個の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
- 入力ファイルには必ず3こ以上の実数値がきさいされているとする
- 出力ファイルの一行目に，最大値，2番目に大きい値，3番目に大きい値 を記載する
- 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   exer4.py   file_in.txt  file_out.txt
```

ファイル入出力については雛形を参照

実行例

課題1. 標準出力 - 雛形 exer1.py

```
$ 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
```

ヒント：
カンマは半角，その後半角スペースが一つ．helloやworldのスペルミスをする方が毎年ごく少数います．
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

- すべての課題に対して，正解プログラムが出力する例を提供します
- 正解画像ファイルなどは雛形のフォルダ内に入れておきます
- 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
- 正解ファイルを上書きするとデバッグが難しくなるのでご注意を

実行例

課題2. 標準出力 - 雛形 exer2.py 出力例

N=16の実行例

```
$ python  exer2.py  16
1
2
hoge
4
fuga
hoge
7
8
hoge
fuga
11
hoge
13
14
hogefuga
16
```

この課題もhogeとfugaのスペルが違う方が毎年います．
そもそも意味のない文字列ではあるのですが，
仕様は仕様なので間違えないようにご注意を．

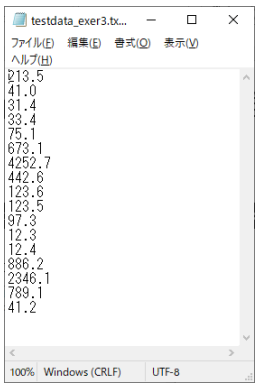
実行例

課題3. ファイル入力と配列 - 雛形 exer3.py 実行例

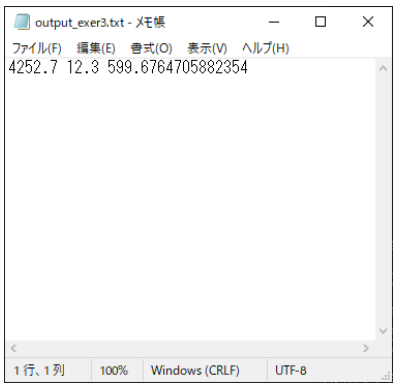
実行コマンド

```
$ python   exer3.py   basic/testdata.txt   basic/output_exer3.txt
```

入力



出力



testdata.txtとoutput_exer3.txt
ファイルは雛形のbasicフォルダ内にあります

このコマンドをそのまま実行するとoutput_exer3.txt
が上書きされてしまうので，適宜出力ファイル名を
変更してください

※実行環境によっては，平均値に誤差が生じる場合
もあると思います
※再展示の動作テストは異なるファイルで行います

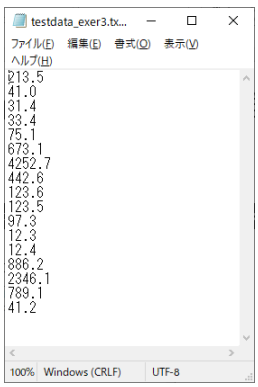
実行例

課題4. ファイル入力と配列 - 雛形 exer4.py 実行例

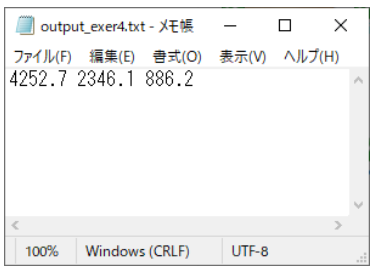
実行コマンド

```
$ python   exer4.py   basic/testdata.txt   basic/output_exer4.txt
```

入力



出力



testdata.txtとoutput_exer4.txt
ファイルは雛形のbasicフォルダ内にあります

このコマンドをそのまま実行するとoutput_exer4.txt
が上書きされてしまうので，適宜出力ファイル名を
変更してください

※実行環境によっては，平均値に誤差が生じる場合
もあると思います
※再展示の動作テストは異なるファイルで行います

課題5~7
画像処理

課題5. 画像のグレースケール化 - 雛形 exer5.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整数にキャストしたものとすること

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

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   exer5.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6. 画像の2値化 - 雛形 exer6.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

- グレースケール化は前課題のものを利用すること
- 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   exer6.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7. モザイク画像の作成（exer7.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exer7.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   exer7.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

スライスによる指定が，配列の範囲外にはみ出していても，エラーにはならず無視されます

実行例

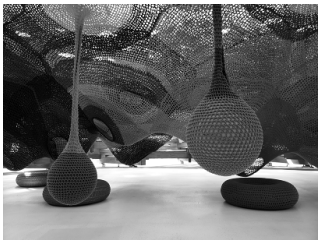
課題5. 画像のグレースケール化

雛形フォルダ内のbasic/img.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   exer5.py   basic/img.png   basic/img_gray.png
```

img.png
入力



img_gray.png
出力

入力・出力画像はbasicフォルダ内にあります

このコマンドをそのまま実行すると正解画像が上書きされるので，出力ファイル名を変えるなどして対応してください

実行例

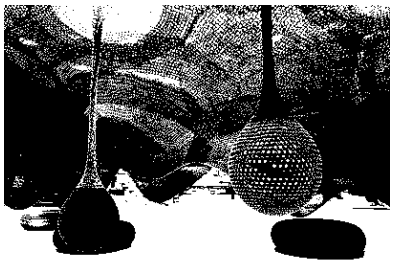
課題6. 画像の2値化 - 雛形 exer6.py

雛形フォルダの basic/img.pngに対して閾値を95として実行すると，basic/img_bin.pngが出力される

実行コマンド

```
$ python   exer6.py   basic/img.png  95  basic/img_bin.png
```

入力 img.png



出力 img_bin.png

入力・出力画像はbasicフォルダ内にあります

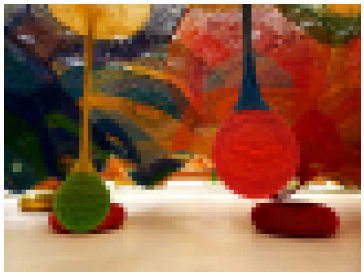
実行例

課題7. モザイク画像の作成（exer7.py）

実行コマンド

```
$ python  exer7.py  basic/img.png  5  basic/img_mo5.png
```

img.png
入力



img_mo5.png
出力

実行コマンド

```
$ python  exer7.py  basic/img.png  15  basic/img_mo15.png
```

img.png
入力

img_mo15.png
出力

入力・出力画像はbasicフォルダ内にあります

課題8~12
フィルタ処理

課題8. 色の変換（exer8.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exer8.pyとし，実行コマンドは以下の通りとする

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

file_in.pngは入力ファイル名

file_out.txtは出力ファイル名

課題9. ガウシアンフィルタ（exer9.py）

画像を読み込み，グレースケール画像に変換後，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 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
```

- file_in.pngは入力ファイル名, file_out.txtは出力ファイル名
- 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() 』は利用せず、for文を用いて自作すること。

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10. ソーベルフィルタ（exer10.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

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

- ファイル名は exer10.py とし，実行コマンドは以下の通り

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

- file_in.pngは入力ファイル名, file_out.txtは出力ファイル名
- グレースケール化の方法についてはひな形を参照
- 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() 』は利用せず、for文を用いて自作すること。

課題11. 勾配強度画像の作成（exer11.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 exer11.py とし，実行コマンドは以下の通りとする

```
$python  exer11.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

- file_in.pngは入力ファイル名, file_out.txtは出力ファイル名
- グレースケール化の方法についてはひな形を参照
- 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

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

縦方向
ソーベルフィルタ

課題12. メディアンフィルタ（exer12.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 exer12.py とし，実行コマンドは以下の通り

```
$python　exer12.py　fname_in.png　fname_out.png
```

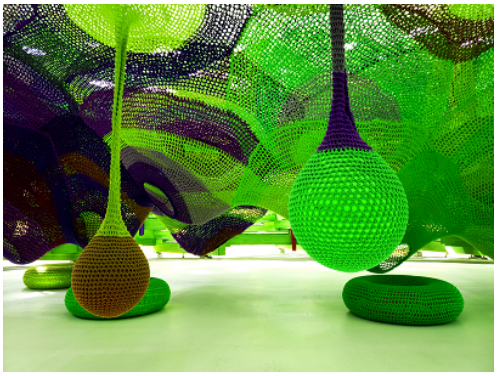
- file_in.pngは入力ファイル名, file_out.txtは出力ファイル名
- グレースケール化の方法についてはひな形を参照
- numpyには，配列の平均・分散・中央値を求める関数があるので利用方法を調べて活用するとよい（自分で計算してもよい）
- フィルタサイズは**5x5**とする
- 画像の周囲2pixelは計算せず0を入れること
- 今回はプログラミング練習が目的なので，cv2の関数『cv2.medianBlur』は利用しないこと

実行例

課題8. 色の変換（exer8.py）

```
$python　exer8.py　filter/img.png　filter/out_exer8.png
```

img.png
入力



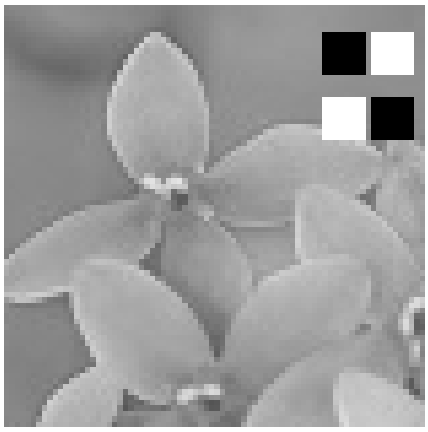
out_exer8.png
出力

ファイルはfilterフォルダ内にあります

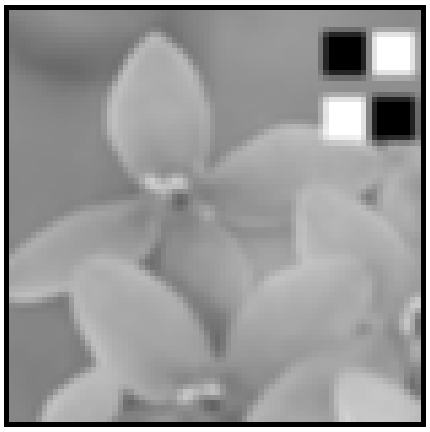
実行例

課題9. ガウシアンフィルタ（exer9.py）

```
$python  exer9.py  filter/img_small.png  filter/img_small_gauss.png
```



img_small.png
入力



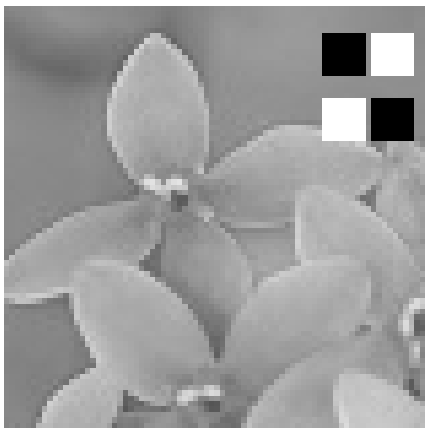
img_small_gauss.png
出力

ファイルはfilterフォルダ内にあります

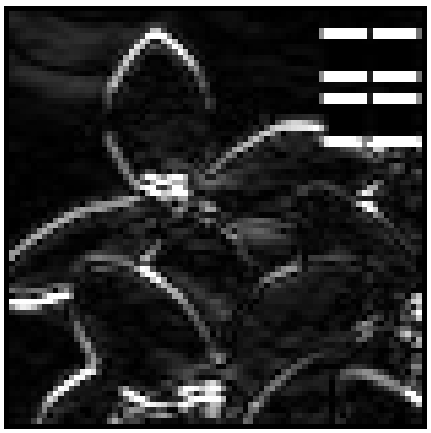
実行例

課題10. ソーベルフィルタ（exer10.py）

```
$python  exer10.py  filter/img_small.png  filter/img_small_sobel.png
```



img_small.png
入力



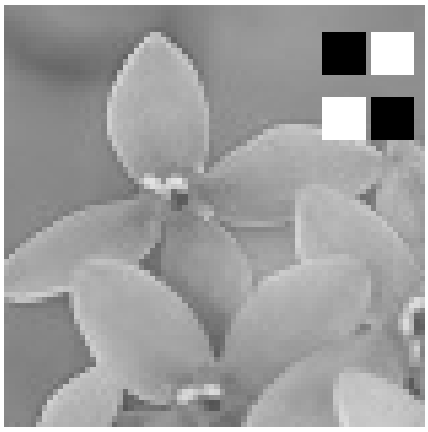
img_small_sobel.png
出力

ファイルはfilterフォルダ内にあります

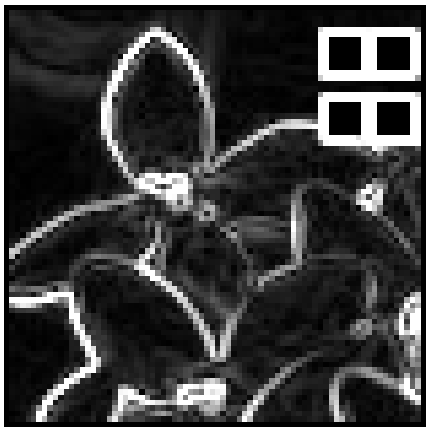
実行例

課題11. 勾配強度画像の作成（exer11.py）

```
$python  exer11.py  filter/img_small.png  filter/img_small_gm.png
```



img_small.png
入力



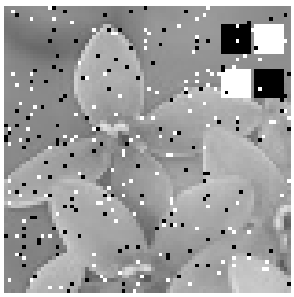
img_small_gm.png
出力

ファイルはfilterフォルダ内にあります

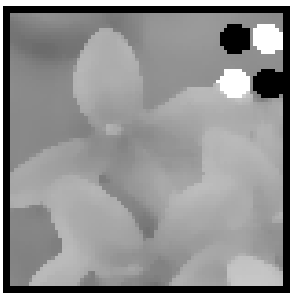
実行例

課題12. メディアンフィルタ（exer12.py）

```
$python  exer12.py  filter/img_noise1.png  filter/img_noise1_med.png
```



img_noise1.png
入力



img_noise1_med.png
出力

```
$python  exer12.py  filter/img_noise2.png  filter/img_noise2_med.png
```

img_noise2.png
入力

img_noise2_med.png
出力

ファイルはfilterフォルダ内にあります

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。
これら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。。

課題13~15
ハーフトーン処理

課題13. ハーフトーン – ディザ法（exer13.py）

画像を読み込み，グレースケール画像に変換後，ティザ法により濃淡を維持した二値画像を作成・保存せよ

- ファイル名は exer13.py とし，実行コマンドは以下の通り

```
$python  exer13.py  fname_in.png  fname_out.png
```

- 出力画像は2値画像（0か255）とする
- ブロックサイズは 4 とし，右図のディザパターンを利用すること
- 画素値xは y=x*16/255 と値を変換してからディザパターンと比較し，画素値がディザパターン値以上のとき時，255を代入すること
- 画像の幅・高さが4の倍数でない場合，画像の右端・下端に余る領域が発生する．この領域は計算せず0を代入するか，そのままディザパターンを重ね合わせて計算すること

15	7	13	1
4	11	5	9
12	3	14	6
0	8	2	10

ディザパターン

講義資料とは少し違うので注意

課題14. ハーフトーン – 濃度パターン法（exer14.py）

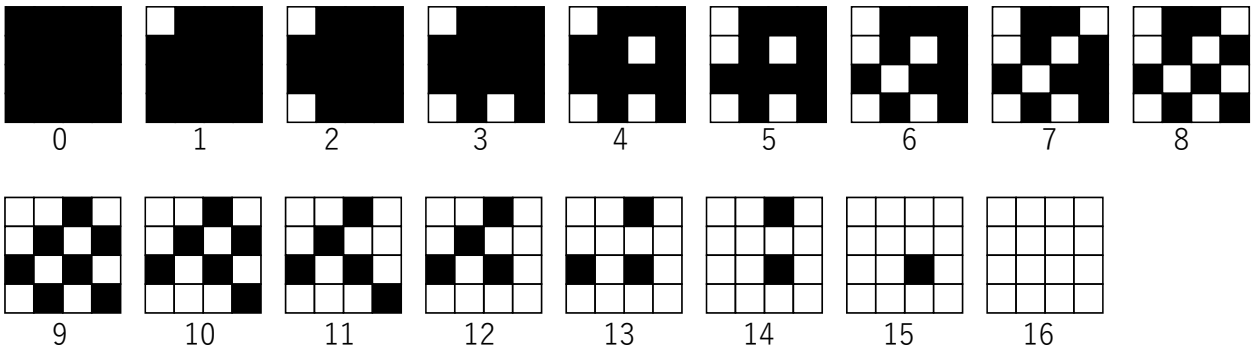
画像を読み込み，グレースケール画像に変換後，濃度パターン法により濃淡を維持した二値画像を作成・保存せよ

- ファイル名は exer14.py とし，実行コマンドは以下の通り

```
$python  exer14.py  fname_in.png  fname_out.png
```

- 出力画像は2値画像（0か255）とする
- ブロックサイズは４x4とし，ブロックの平均輝度値に応じて適切な濃度パターンを配置すること　（次ページ参照）
- 画像の幅・高さが4の倍数でない場合，画像の右端・下端に余る領域が発生する．この領域は計算せず0を代入すること

課題14. ハーフトーン – 濃度パターン法（exer14.py）



4x4ブロックの平均画素値xが，$\frac{255}{17}i \leq x < \frac{255}{17}(i+1)$の範囲に入っていれば，i番目のパターンを適用する

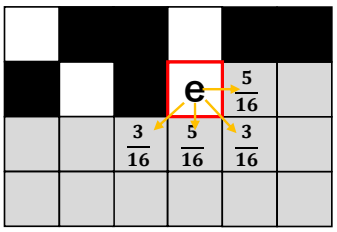
例)
あるブロックの平均画素値が 73.0の時，これは
4/17*255 ~5/17*255
の範囲なのでパターン4を採用する

課題15. ハーフトーン – 誤差拡散法（exer15.py）

画像を読み込み，グレースケール画像に変換後，誤差拡散法により濃淡を維持した二値画像を作成・保存せよ

- ファイル名は exer15.py とし，実行コマンドは以下の通り

```
$python  exer15.py  fname_in.png  fname_out.png
```



誤差拡散する隣接画素

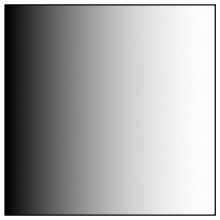
- 出力画像は2値画像(0か255)とする
- 画素値（すでに誤差値が足されたもの）が **127より**大きい時にその画素を白（255）に，そうでない画素を黒(0)にすること
- 拡散する誤差の割合は右図の通りとする
- 右端の画素を計算する際，その右隣の画素が存在しないため右へ誤差を拡散できない．**この場合は，単純に右隣への誤差拡散を省略せよ．右へ行くべき誤差を下方向に拡散させることは行わない．**
- 左端・下端の画素の誤差拡散についても同様に拡散を省略すること。

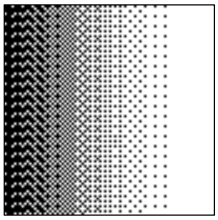
実行例

課題13. ハーフトーン – ディザ法（exer13.py）

課題13, 例1

```
$python  exer13.py  ht/grd.png ht/grd_dither.png
```


入力画像
grd.png


出力画像
grd_dither.png

課題13, 例2

```
$python  exer13.py  ht/cat.png  ht/cat_dither.png
```


入力画像
cat.png


出力画像
cat_dither.png

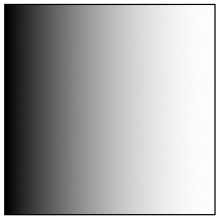
画像データはhtフォルダにあります

実行例

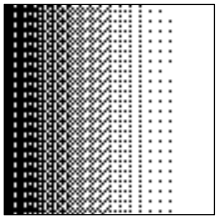
課題14. ハーフトーン – 濃度パターン法（exer14.py）

課題14, 例1

```
$python  exer14.py  ht/grd.png  ht/grd_noudo.png
```



入力画像
grd.png



出力画像
grd_noudo.png

課題14, 例2

```
$python  exer14.py  ht/cat.png  ht/cat_noudo.png
```

入力画像
cat.png

出力画像
cat_noudo.png

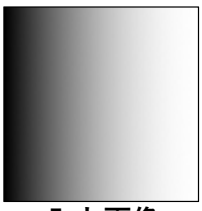
画像データはhtフォルダにあります

実行例

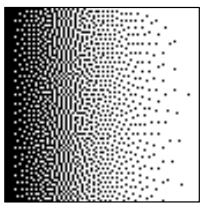
課題15. ハーフトーン – 誤差拡散法（exer15.py）

課題15, 例1

```
$python  exer15.py  ht/grd.png  ht/grd_err.png
```



入力画像
grd.png



出力画像
grd_err.png

課題15, 例2

```
$python  exer15.py  ht/cat.png  ht/cat_err.png
```

入力画像
cat.png

出力画像
cat_err.png

画像データはhtフォルダにあります

課題16~18
フーリエ変換

課題16. 1次元離散フーリエ変換（exer16.py）

実数列が書き込まれたテキストファイルを読み込み，その実数列をフーリエ変換した結果をテキストファイルとして出力せよ

ファイル名はexer16.pyとし、実行コマンドは以下の通り

入出力の詳細は雛形と出力例を参照せよ

```
$python   exer16.py   sample_fl.txt    fname_out.txt
```

- 得られる周波数係数 F_k は複素数となる．Pythonには複素数型が存在するがこれは利用せず，$F_k = R_k + i\,I_k$ と実部R_kと虚部I_kに分けて保持し，それぞれをテキスト形式で出力せよ
- 離散フーリエ変換には複数の定義が存在するが以下のものを利用すること

フーリエ変換　$$F_k = \frac{1}{N}\sum_{l=0}^{N-1} f_l \left(\cos\frac{2\pi kl}{N} - i\sin\frac{2\pi kl}{N}\right)$$

※今回はプログラミング練習が目的なので，フーリエ変換自体を行うライブラリ関数（np.fftなど）は利用しないこと．math.cos()関数などは利用してよい．

※上式の π は、math.piを利用すると良い．今回はnp.radians()は利用しないこと．

課題17. 1次元逆離散フーリエ変換（exer17.py）

複素数列が書き込まれたテキストファイルを読み込み，その配列を逆フーリエ変換した結果をテキストファイルとして出力せよ

ファイル名はexer16.pyとし，実行コマンドは以下の通り．

入出力ファイル形式は雛形と入出力例を参照のこと

```
$python  exer17.py  Fk.txt  fname_out.txt
```

- フーリエ変換には複数の定義が存在するが以下のものを利用すること

逆フーリエ変換　$f_l = \sum_{k=0}^{N-1} F_k \left(\cos\frac{2\pi kl}{N} + i\sin\frac{2\pi kl}{N} \right)$

- 入力される配列F_kと，得られる配列 f_lは複素数となる．Pythonには複素数型が存在するがこれは利用せず，$f_l = r_l + i_l$と実部r_lと虚部i_lに分けて保持し，それぞれをテキスト形式で出力せよ
- 前の課題で作成したデータを逆フーリエ変換し，ほぼ元に戻ることを確認せよ（虚部はほぼ 0 になる）

※今回はプログラミング練習が目的なので，フーリエ変換自体を行うライブラリ関数（np.fftなど）は利用しないこと（math.cos()関数などはOK）

※上式の π は、math.piを利用すると良い．今回はnp.radians()は利用しないこと．

課題18. 2次元フーリエ変換（exer18.py）

画像を読み込み，グレースケール変換後，画像 f_{ij}をフーリエ変換し，フーリエ係数F_{kl}を画像として出力せよ

ファイル名はexer17.pyとし，実行コマンドは以下の通りとする

入出力ファイルの詳細は雛形を参照せよ

```
$python  exer18.py  fname_in.png  Ruv.png  Iuv.png
```

- フーリエ変換には以下の式を利用すること

フーリエ変換：　$F_{uv} = \frac{1}{WH} \sum_{y=0}^{H-1} \sum_{x=0}^{W-1} f_{xy} e^{-\frac{2\pi xu}{W}i} e^{-\frac{2\pi yv}{H}i}$

- 結果は$F_{uv} = R_{uv} + i\, I_{uv}$ と実部と虚部に分け，それぞれを画像2枚として出力せよ
- 実部R_{uv}と虚部I_{kl}は，*値域*[0,255]の範囲に収まらないため，最小値と最大値を用いて，値を[0,255]に正規化すること
  - 具体的には，(値 – 最小値)/(最大値-最小値) * 255 という変換をせよ（※ R_{uv}とI_{uv}は個別に正規化すること）
  - 素朴な実装は処理時間が長くなるので，小さな画像でテストするとよい

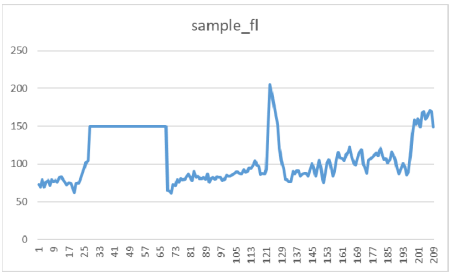
※今回はプログラミング練習が目的なので，フーリエ変換自体を行うライブラリ関数（np.fftなど）は利用しないこと（math.cos()関数などはOK）

※上式の π は、math.piを利用すると良い．今回はnp.radians()は利用しないこと．

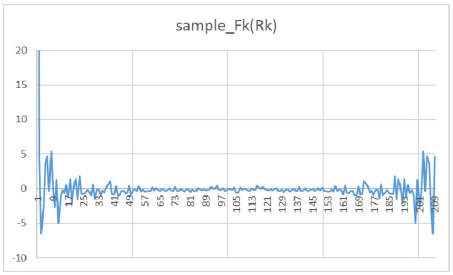
実行例

課題16. 1次元離散フーリエ変換（exer16.py）

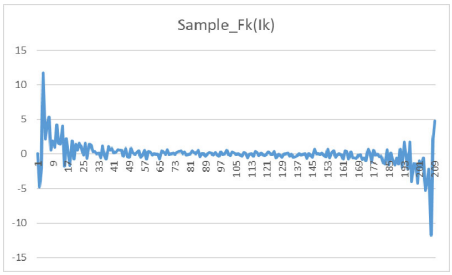
```
$python　exer16.py　fourie/sample_fl.txt　fourie/output_Fk.txt
```



入力実数列 sample_fl.txt



出力の実部 output_Fk.txt
（見やすさのため値域を[-20,20]にした）



出力の虚部 output_Fk.txt
（見やすさのため値域を[-15,15]にした）

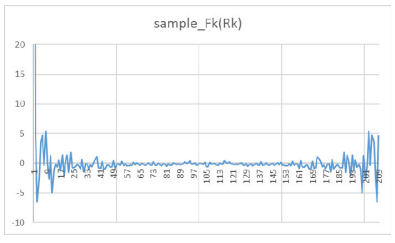
画像データはfourieフォルダにあります

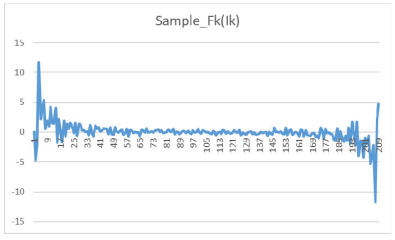
実行例

課題17. 1次元逆離散フーリエ変換（exer17.py）

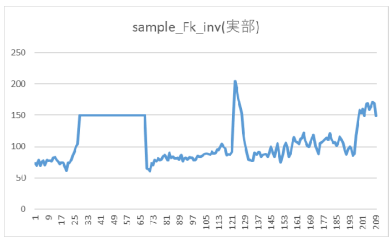
問16で得られたoutput_Fk.txt 内の複素数配列を逆フーリエ変換すると，
output_Fk_inv.txt (複素数配列)が得られる

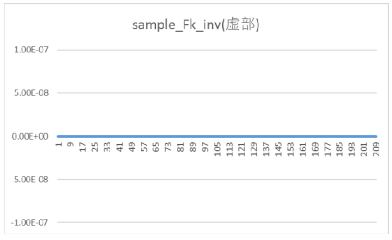
```
$python　exer17.py　fourie/output_Fk.txt　fourie/output_Fk_inv.txt
```





入力複素数列
（実部（上）と虚部（下））





出力複素数列
（実部（上）と虚部（下））

元の実数列（sample_fl）を
フーリエ変換したもの(output_Fk)を
逆フーリエ変換すると(output_Fk_inv)
元の関数に戻ります。

sample_Fk_invの虚部はほぼゼロ（非常に
小さな値）になります

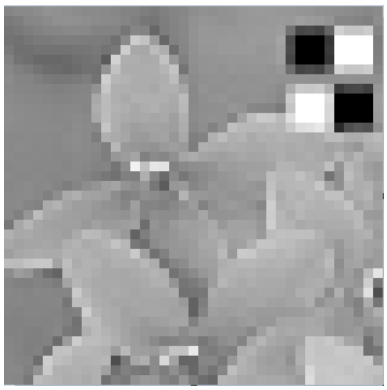
画像データはfourieフォルダにあります

実行例

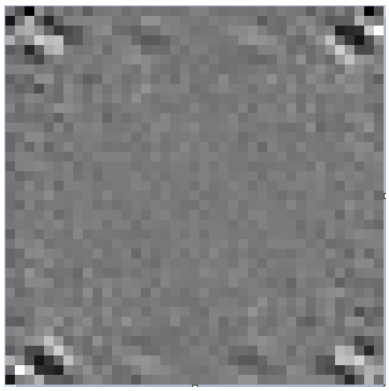
課題18. 2次元フーリエ変換（exer18.py）：実行例

Fourie/img.png をフーリエ変換すると，複素数画像 (Rvu + i Ivu)が得られる．実部と虚部それぞれを画像として出力したものが Rvu.pngとIvu.png

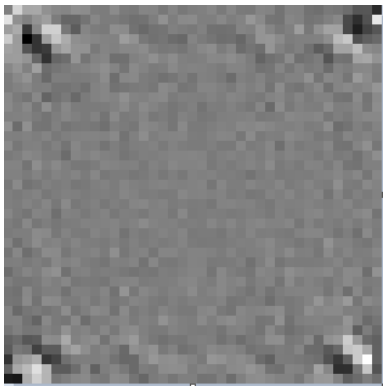
```
$python  exer18.py  fourie/img.png  fourie/Rvu.png  fourie/Ivu.png
```



img.png



Rvu.png



Ivu.png

素朴な実装をすると4重ループになり，pythonではそれなりの時間がかかります。

発展課題

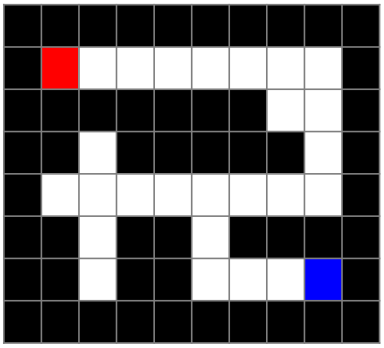
発展課題はscombより提出するのでなく
演習授業中に教員へ直接提出してください

課題19 迷路

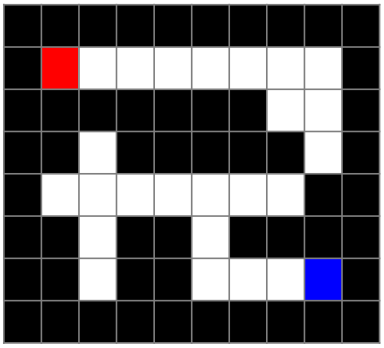
- 迷路を解き，スタートからゴールまでの経路が有る場合は最小歩数を表示し，経路が無い場合は-1を表示するプログラムを作成せよ
- 迷路の仕様は以下の通り
  - 迷路は画像で与えられる
  - 黒画素(r,g,b=　0,　0,　0)は壁で通れない
  - 白画素(r,g,b=255,255,255)は通れる通路
  - 赤画素(r,g,b=255,　0,　0)はスタート
  - 青画素(r,g,b=　0,　0,255)はゴール
  - ある白画素から一歩で，上下左右の白画素に移動できる
- 画像データはコマンドライン引数で受け取り，計算結果は標準出力に表示する

```
$python　exer19.py　meiro.bmp
```

- 標準出力には結果以外を表示しないこと(提出時に不要なprint文は削除してください)
- コールスタックの深さに限界があるので再帰関数を利用して実装すると大きな画像に対しては動かない可能性があります
- meiroディレクトリ内に迷路画像サンプルがある
- 雛形なし



正解は18



正解は－1

課題20：Deconvolution

ボケ画像と点拡がり関数画像を受け取り、deconvolutionにより劣化前の画像を復元せよ

実行コマンドは以下の通りとする. Input_img.pngは入力ボケ画像、input_kernel.pngは点広がり関数

```
$python　exer20.py　input_img.png input_kernel.png
```

- 講義中に扱った「単純な手法」と「wiener filter」両方を実装し、パラメータを変化させながら両者を比較せよ
- 点広がり関数の読み込みについては雛形を参照すること
- Deconvフォルダ内に以下のファイルがある
  - gauss_kernel.png : ガウシアンカーネル
  - gauss_img.png　: ガウシアンカーネルによるボケ画像
  - gauss_dec_simple.png: 「gauss_img.png」を単純な手法により復元したもの
  - gauss_dec_wie.png　: 「gauss_img.png」をWiener filterにより復元したもの
  - line_kernel.png : 直線状の点広がり関数
  - line_img.png　:直線状の点広がり関数によるボケ画像
  - line_dec_simple.png: 「line_img.png」を単純な手法により復元したもの
  - line_dec_wie.png　: 「line_img.png」をWiener filterにより復元したもの