

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_AL150999

zipファイル名の例: dm_AL150999.zip

課題雛形: <http://takashijiri.com/classes/dm2020/>

入出力 : 各課題において詳細な仕様が定義されるので正しく従うこと

注意 :

- 採点が自動化されているため, 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ないため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01～11 **11/30 23:59**

後半：課題12～19 **12/14 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

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
- 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を併用
 - 50個のブレイクアウトルームを自由に利用してよい
 - 1~3はTAが待機する
- 質問はslackへ
 - 簡単なものはTAが解答, ややこしいものはTA分室にて
 - 質問は講義時間内でなくてもOK, ただし解答は基本的に講義中に
- 分室の利用を
 - 知人と好きなブレイクアウトルームへ行く
 - メインの部屋でやる
- 約束
 - 正解コードを共有しないでください（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

前半

- 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 画像のグレースケール化 easy
- exer5 画像データの2値化 easy
- exer6 モザイク画像 normal
- 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

後半

- exer12 ティザ法 easy
- exer13 濃度パターン法 easy
- exer14 誤差拡散法 normal
- exer15 フーリエ変換 normal
- exer16 逆フーリエ変換 normal
- exer17 2Dフーリエ変換 hard?
- exer18 名簿処理 normal
- exer19 迷路 hard?

課題1~6 pythonの基本と画像の入出力

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

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
```

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事

課題1. 標準出力 - 雛形 exer1.py : 実行例

```
$ 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たまにいます.
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

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5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5両方の倍数の時には、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2. 標準出力 - 雛形 exer2.py 出力零

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
```

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・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
```

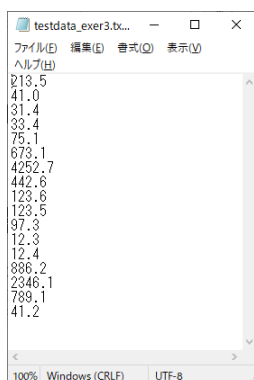
ファイル入出力については雛形を参照

課題3. ファイル入力と配列 - 雛形 exer3.py 実行例

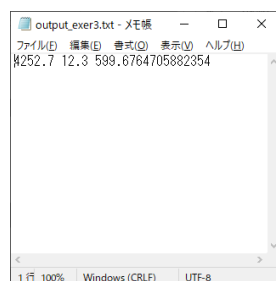
実行コマンド

```
$ python exer3.py basic/testdata_exer3.txt basic/output_exer3.txt
```

入力



出力



testdata_exer3.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

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4. 画像のグレースケール化 - 雛形 exer4.py：実行例

雛形フォルダ内のbasic/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 basic/img1.png basic/img1_gray.png
```



img1.png
入力



img1_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，適宜ファイル名を変えてください

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5. 画像の2値化 - 雛形 exer5.py：実行例

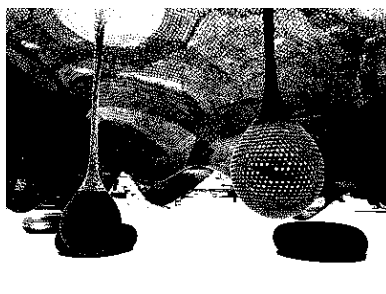
雛形フォルダの basic/img1.png に対して閾値を95として実行すると，basic/img1_bin.pngが出力される

実行コマンド

```
$ python exer5.py basic/img1.png 95 basic/img1_bin.png
```



入力 img1.png



出力 img1_bin.png

入力・出力画像はbasicフォルダ内にあります

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 $R \times R$ 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：

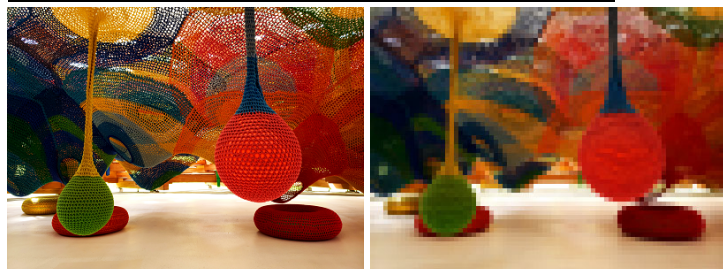
スライスを活用してください

スライスによる指定が，一部，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

課題6. モザイク画像の作成 (exer6.py)：実行例

実行コマンド

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

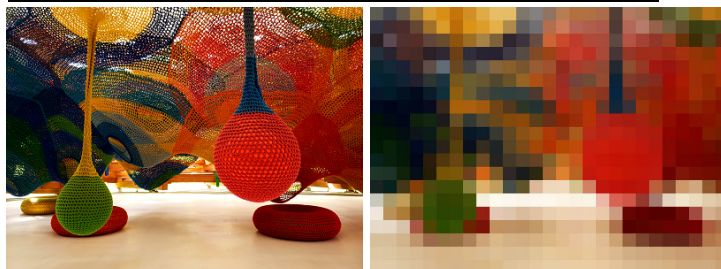


img.png
入力

img_mo5.png
出力

実行コマンド

```
$ python exer6.py basic/img1.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7~11 フィルタ処理

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7. 色の変換 (exer7.py) : 実行例

```
$python exer7.py filter/img1.png filter/out_exer7.png
```



img1.png
入力



out_exer7.png
出力

ファイルはfilterフォルダ内にあります

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
```

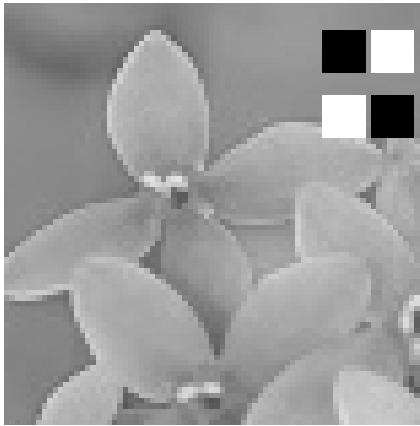
- ・ グレースケール化の方法についてはひな形を参照
- ・ 右図のガウシアンフィルタを利用すること
- ・ 画像の周囲1pixelは計算せず0を入れること
- ・ 今回はプログラミング練習が目的なので, 次の関数『cv2.filter2D() / cv2.GaussianBlur() / cv2.Sobel() / np.convolve()』は利用しないこと

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

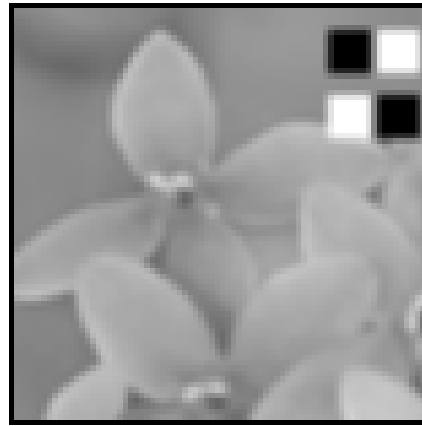
フィルタの係数

課題8. ガウシアンフィルタ (exer8.py) : 実行例

```
$python exer8.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

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

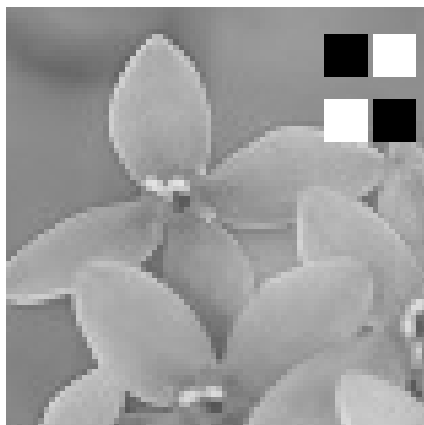
```
$python exer9.py fname_in.png fname_out.png
```

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 倍して正值に変換すること
- 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9. ソーベルフィルタ (exer9.py) : 実行例

```
$python exer9.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

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
```

- 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	0	1
-2	0	2
-1	0	1

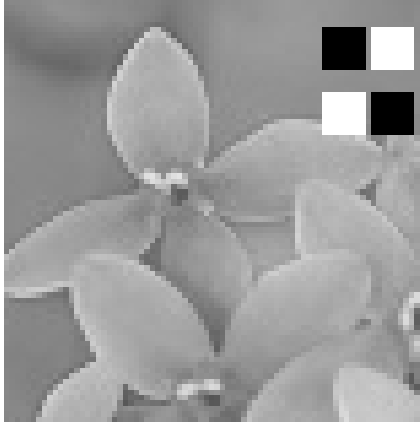
横方向
ソーベルフィルタ

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

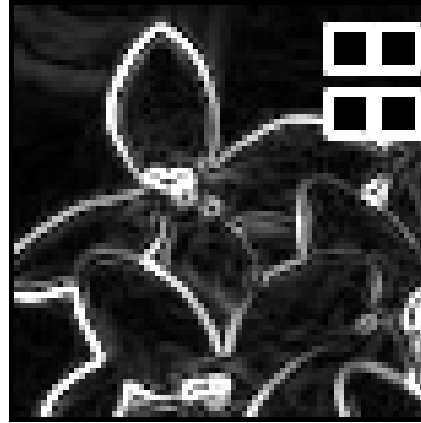
縦方向
ソーベルフィルタ

課題10. 勾配強度画像の作成（exer10.py）：実行例

```
$python exer10.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

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

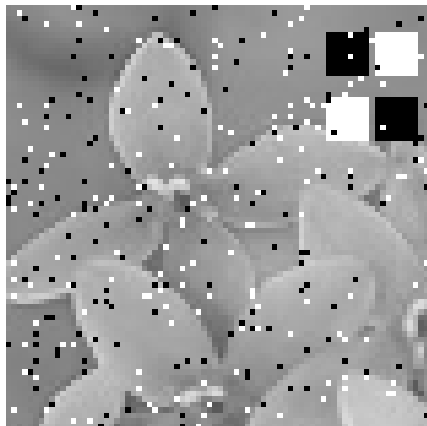
- ファイル名は exer11.py とし、実行コマンドは以下の通り

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

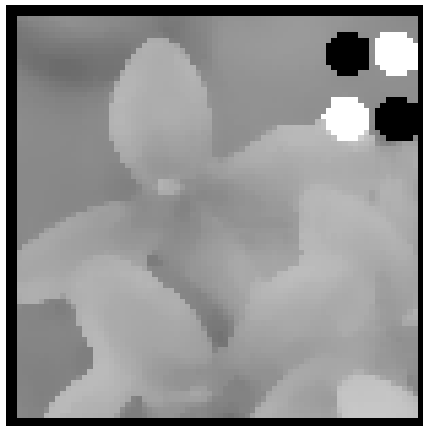
- np.arrayには、配列の平均・分散・中央値を求める関数があるので利用方法を調べて活用するとよい（自分で計算してもよい）
- フィルタサイズは**5x5**とする
- 画像の周囲2pixelは計算せず0を入れること
- 今回はプログラミング練習が目的なので、cv2の関数『cv2.medianBlur』は利用しないこと

課題11. メディアンフィルタ（exer11.py）：実行例

```
$python exer11.py filter/img_noise.png filter/img_noise_med.png
```



Img_noise.png
入力



img_noise_med.png
出力

ファイルはfilterフォルダ内にあります

課題11. メディアンフィルタ（exer11.py）：実行例

課題11

```
$python exer11.py filter/img4.png filter/img4_median.png
```



入力画像
img4.png



出力画像
img4_median.png

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。。

後半

課題12~14 ハーフトーン処理

課題12. ハーフトーン – ティザ法 (exer12.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 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
```

- 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、右図のテ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ティザパターンと比較し、画素値がテ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、そのままテ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

ティザパターン
講義資料とは少し
違うので注意

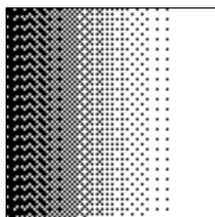
課題12. ハーフトーン - ティザ法 (exer12.py) : 実行例

課題12, 例1

```
$python exer12.py ht/grd.png ht/grd_tiza.png
```



入力画像
grd.png



出力画像
grd_tiza.png

課題12, 例2

```
$python exer12.py ht/cat.png ht/cat_tiza.png
```



入力画像
cat.png



出力画像
cat_tiza.png

画像データはhtフォルダにあります

課題13. ハーフトーン - 濃度パターン法 (exer13.py)

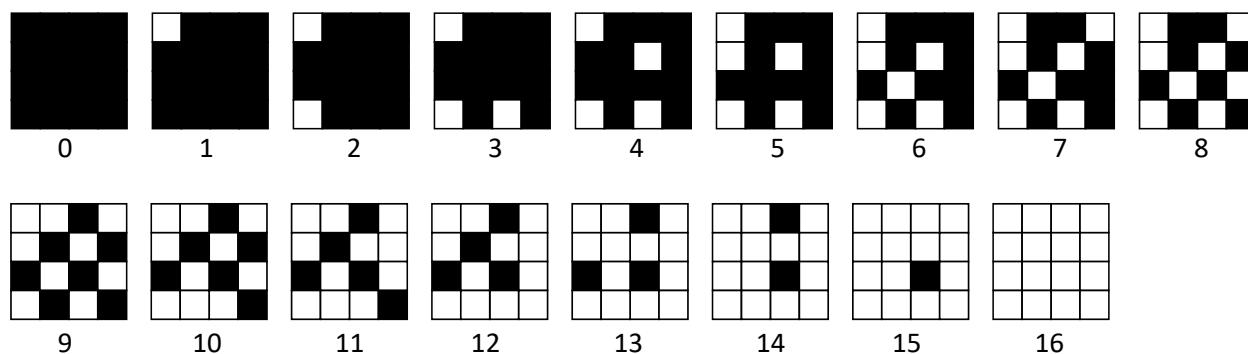
画像を読み込み、グレースケール画像に変換後、濃度パターン法により濃淡を維持した二値画像を作成・保存せよ

- ファイル名は 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 x 4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13. ハーフトーン - 濃度パターン法 (exer13.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
 $4/17 \times 255 \sim 5/17 \times 255$
の範囲なのでパターン4を採用する

課題13. ハーフトーン - 濃度パターン法 (exer13.py) : 実行例

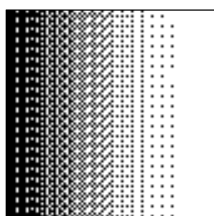
課題13, 例1

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



入力画像

grd.png



出力画像

grd_noudo.png

課題13, 例2

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



入力画像

cat.png



出力画像

cat_noudo.png

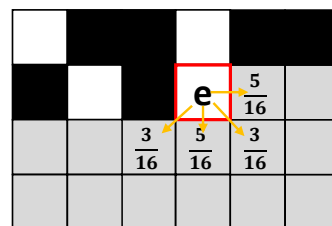
画像データはhtフォルダにあります

課題14. ハーフトーン - 誤差拡散法 (exer14.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 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
```



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.

課題14. ハーフトーン - 誤差拡散法 (exer14.py) : 実行例

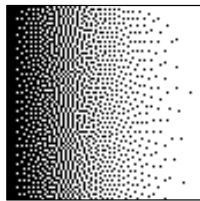
課題14, 例1

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



入力画像

grd.png



出力画像

grd_err.png

課題14, 例2

```
$python exer14.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15~17 フーリエ変換

課題15. 1次元離散フーリエ変換 (exer15.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exer15.pyとし、入出力の詳細は雛形と出力例を参照せよ

```
$python exer15.py sample_fl.txt fname_out.txt
```

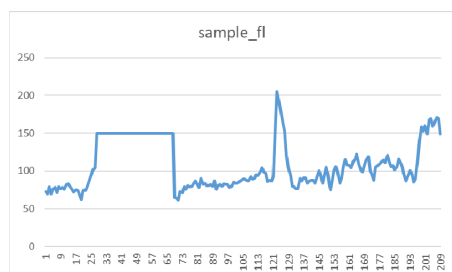
- 得られる周波数係数 F_k は複素数となる。Pythonには複素数型が存在するがこれは利用せず、 $F_k = R_k + i I_k$ と実部 R_k と虚部 I_k に分けて保持し、それぞれをテキスト形式で出力せよ
- 離散フーリエ変換には複数の定義が存在するが以下のものを利用すること

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

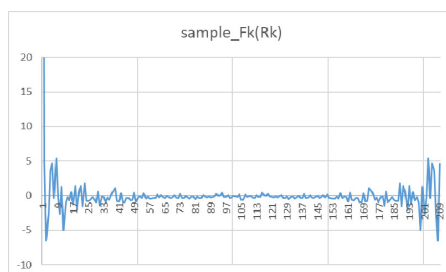
※今回はプログラミング練習が目的なので、フーリエ変換自体を行うライブラリ関数 (`np.fft` など) は利用しないこと。もちろん `math.cos()` 関数などは利用してよい。

課題15. 1次元離散フーリエ変換 (exer15.py) : 実行例

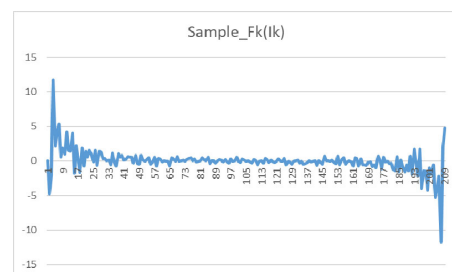
```
$python exer15.py fourie/sample_fl.txt fourie/output_Fk.txt
```



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

課題16. 1次元逆離散フーリエ変換 (exer16.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、入出力ファイル形式は雛形と入出力例を参照のこと

```
$python exer16.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

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

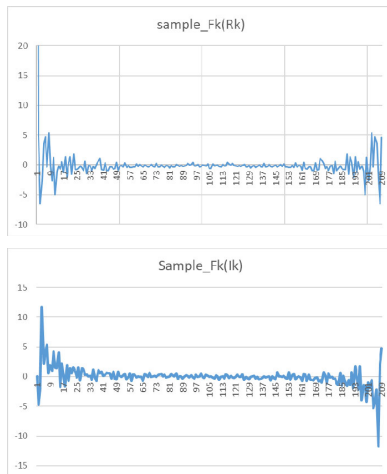
- 入力される配列 F_k と、得られる配列 f_l は複素数となる。Pythonには複素数型が存在するがこれは利用せず、 $f_l = r_l + i i_l$ と実部 r_l と虚部 i_l に分けて保持し、それぞれをテキスト形式で出力せよ
- 前の課題で作成したデータを逆フーリエ変換し、ほぼ元に戻ることを確認せよ（虚部はほぼ0になる）

※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 ($\text{math.cos}()$ 関数などはOK)

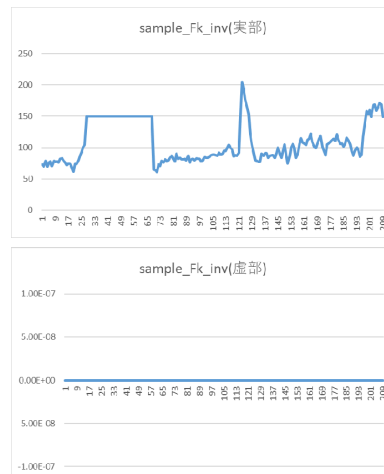
課題16. 1次元逆離散フーリエ変換 (exer16.py) : 実行例

問15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 exer16.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

課題17. 2次元フーリエ変換 (exer17.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, 入出力ファイルの詳細は雛形を参照せよ

```
$python exer17.py fname_in.png Ruv.png Iuv.png
```

- 以下の式を利用すること

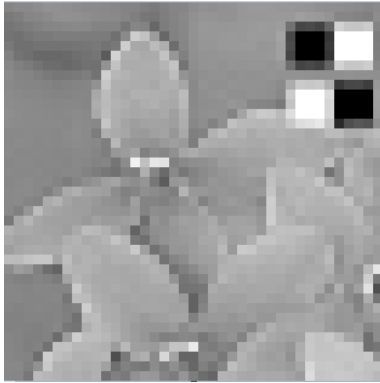
$$\text{フーリエ変換: } F_{uv} = \frac{1}{WH} \sum_{y=0}^{H-1} \sum_{x=0}^{W-1} f_{xy} e^{-\frac{2\pi xu}{W}i} e^{-\frac{2\pi yv}{H}i}$$

- 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, $(\text{値} - \text{最小値}) / (\text{最大値} - \text{最小値}) * 255$ 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 ($\text{math.cos}()$ 関数などはOK)

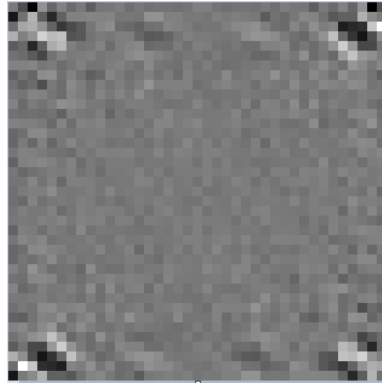
課題17. 2次元フーリエ変換 (exer17.py) : 実行例

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

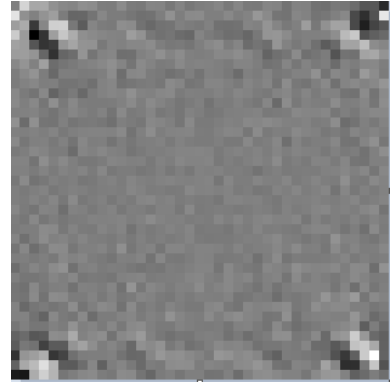
```
$python exer17.py fourie/img.png fourie/Rvu.png fourie/Ivu.png
```



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。

課題18~19
発展課題

課題18：名簿データ処理

- あるディレクトリ複数の名簿ファイルが入っており、各名簿ファイルの各行には学生データが書かれている。学生データは、“学籍番号”、“名前”、“テストの点数”であり、この順に、カンマで区切られて書かれている
- あるディレクトリ内の全ての名簿ファイルを読み込み、全ての学生データをひとつのファイルにまとめて出力せよ。ただし、以下の仕様を満たすこと。
 - “名前”は削除し、学籍番号と点数をカンマ区切りで書き込む
 - 学籍番号順にソートする
- 実行コマンドは以下の通り

```
$python exer18.py in_dir output.txt
```

- in_dir は、名簿フォルダを含むディレクトリパス、output.txtは出力ファイル名

※ 名簿の例はmeiboフォルダにある

※ (<http://www.gaoshukai.com/lab/0003/>で作成した架空のデータです)

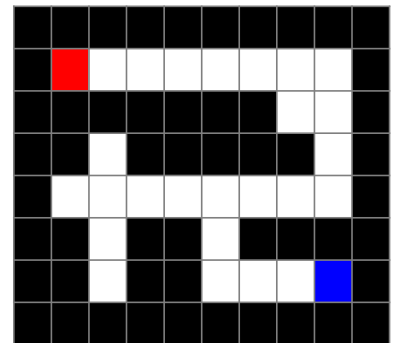
※ 大学教員になって数年目ですが、学会運営や教育業務のため、機械学習というよりこういう用途にばかりpythonを使っています。

課題19 迷路

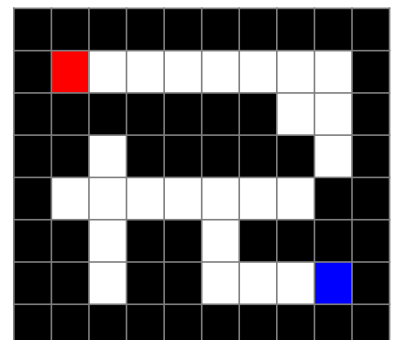
- 迷路を解き、スタートからゴールまでの経路が有る場合は最小歩数を表示し、経路が無い場合は-1を表示するプログラムを作成せよ
- 迷路の仕様は以下の通り
 - 迷路は画像で与えられる
 - 黒画素(r,g,b= 0, 0, 0)は壁で通れない
 - 白画素(r,g,b=255,255,255)は通れる通路
 - 赤画素(r,g,b=255, 0, 0)はスタート
 - 青画素(r,g,b= 0, 0,255)はゴール
 - ある白画素から一歩で、上下左右の白画素に移動できる
- 画像データはコマンドライン引数で受け取り、計算結果は標準出力に表示する

```
$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)
- meiroディレクトリ内に迷路画像サンプルがある



正解は18



正解は-1