ディジタルメディア処理 2023

担当：井尻敬

締め切り：

課題前半 （問01~問12）12/15 24:00

課題後半 （問13~問23）01/19 24:00

提出方法:

1.『dm_学籍番号』というフォルダを作成し、

2. その中に解答となるコードをすべて入れて、

3. zip圧縮（フォルダを右クリック>送る>圧縮）し、

3. 出来上がったファイルをscombzより提出してください

※フォルダ名の例: dm_AL200999 / zipファイル名の例: dm_AL200999.zip

※ 発展課題は、教員（またはTA）に確認を受けてください。

雛形:

こちら (http://takashiijiri.com/classes/dm2023/) に雛形があるので、必ず利用してください

雛形のファイル名は変更しないでください（課題XXのファイル名はexerXX.pyのまま）

解答に関する注意:

- 各課題について、コマンドライン引数の詳細な仕様など、入出力の仕様が指定されます。正しく従ってください。
- 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極端に長い時間のかかるもの（20秒以上）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、自身で作成したプログラムの評価を行ってください。ただし、評価には配布したものとは別のデータを使います。
- 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)に公開する事は避けてください。
- 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、しかるべき処置をとります。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。

準備

Step1 : 本科目のページ (http://takashiijiri.com/classes/dm2023/)より、下記のファイルを取得してください。

- python_install.pdf
- template.zip

Step2 : python_install.pdf に従い Pythonをインストールしてください。

Step3 : template.zipを解凍してください（フォルダをダブルクリック > 全て展開 を選択）

Step4 : コマンドプロンプトを開き、解凍してできたtemplate フォルダのトップをカレントにしてください。（フォルダを開いて アドレスバーに「cmd」と入れてもOK）

Step5 : 下記のコマンドを実行し、テンプレートを初期化してください。

```
>python dm_starter.py
```

Step6: 初期化キーの提出

実行に成功すると、

```
>python dm_starter.py
initialize successXXXXXXXXXXX.XXXXXXXXX
```

と出力されます。この出力された一行『`initialize successXXXXXXXXXXX.XXXXXXXXX`』をscombzの『テスト : 課題準備』より提出してください。「XXXXXXXXXXX.XXXXXXXXX」には、数値が入ります。この数値を不正行為防止に利用するため、提出がない場合は採点できません。また、ここで初期化した雛形を必ず利用して解答して下さい（ファイル紛失などの際はご連絡ください）。

目次

day1

day2

day3

day4

day5

day6

易	★
普通	★★
やや難	★★★
難	★★★★

※ 後半は多少難易度が高いので、早め早めに演習を進めてください。

※ 雛形に多くのヒントがあります。まずは雛形にあるコードを読んでください。

課題01 hello world

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。

```
python  exer01.py  a  b
```

仕様

- 非負整数値 aとbをコマンドライン引数より受け取る
- a x b 回『hello, world』と出力する

※ すべての課題にひな形 exerXX.pyが用意してあるので参考にしてください。

※ 毎年数名スペルミスをする方がいます。カンマは半角で、その後に半角スペースが一つです．ご注意を。

入出力例 ―----------

```
> python exer01.py 2 3
hello, world
hello, world
hello, world
hello, world
hello, world
hello, world
```

a=2, b=3なら、6回『hello, world』と出力します。

※ 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注意

課題02 hogefuga :

非負整数値Nをコマンドラインから受け取り、下記のルールに従い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。実行コマンドは以下の通り。

```
python exer02.py N
```

仕様 :

- 入力Nはコマンドライン引数として受け取る
- 1からNまでの整数を順番に標準出力に表示する。ただし、以下のルールに従う。
  - 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

入出力例 —----------

N=16なら下記のような出力になります。

```
> python exer02.py 16
1
2
hoge
4
fuga
hoge
7
8
hoge
fuga
11
hoge
13
14
hogefuga
16
```

※ hogeとfugaのスペルが違う方が例年数名いるので注意してください（本質的な間違いではないのですが仕様は仕様なので。。）。

課題03 ファイル入力と配列1：

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。実行コマンドは以下の通り 。

```
python exer03.py file_in.txt file_out.txt
```

仕様

- 入出力ファイル名（file_in.txt、file_out.txt）は，コマンドライン引数として取得
- 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
- 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

※ ファイル入出力については雛形を参照

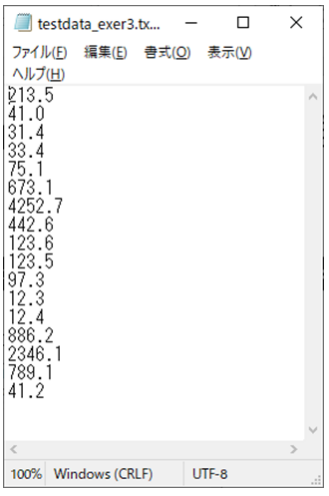
入出力例 ―----------

basicフォルダ内に入出力例があります。下記のコマンドにより入力データ『basic/testdata.txt』の処理を行うと、出力データ『./output_exer3.txt』が得られます。

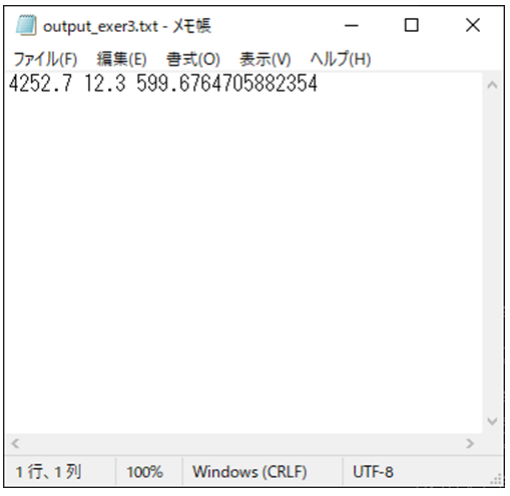
```
python exer03.py basic/testdata.txt ./output_exer3.txt
```

※ baseフォルダの basic/testdata.txt を読み込み、./output_exer3.txt に書き出すコマンド

※正解データへの上書き防止のため、カレントへ出力するコマンドを提示します。



入力例



出力例

※ 実行環境・利用するライブラリに依存して多少の誤差が出る場合がありますが、小さな誤差であれば正解扱いとするのでピッタリ同じになる必要はありません。

課題04 ファイル入力と配列2:

ファイルから複数の数値データを読み、１番目・２番目・３番目に小さい数値をファイルに出力するプログラムを作成せよ。実行コマンドは以下の通り。

```
python exer04.py file_in.txt file_out.txt
```

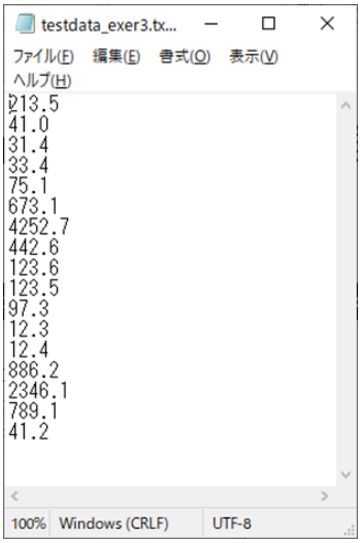
仕様:

- 入出力ファイル名（file_in.txt、file_out.txt）は、コマンドライン引数として取得
- 入力ファイルには３個以上の実数が、１行に１つずつ記載されている
- 出力ファイルの一行目に、最小値、2番目に小さい値，3番目に小さい値 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

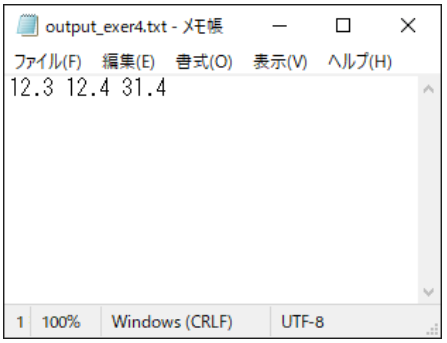
入出力例 ―----------

basicsフォルダ内に入出力例があります。下記のコマンドにより、入力データ『basic/testdata.txt』の処理を行うと、出力データ『output_exer4.txt』が得られます。

```
python exer04.py basic/testdata.txt ./output_exer4.txt
```



入力例



出力例

「basic/testdata.txt」の最も小さな3つの数字が、output_exer4.txtに出力されます。

課題05. 画像のグレースケール化

画像データを読み込み、グレースケール化して保存するプログラムを作成せよ。実行コマンドは以下の通り。

```
> python exer05.py file_in.png file_out.png
```

仕様:

- 入出力ファイル名（file_in.png, file_out.png）は，コマンドライン引数として取得
- グレースケール値は、赤・緑・青チャンネルの平均を整数キャストしたものとする

```
グレースケール値　= (int)(赤 + 緑 + 青)/3
```

入出力例 ----------

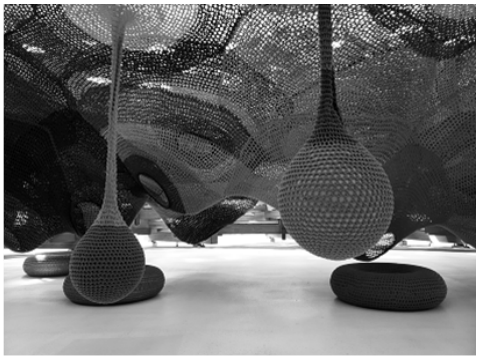
入力・出力画像はbasicフォルダ内にあります。以下のコマンドを実行すると、basicフォルダ内のbasic/img.png がグレースケール化されたファイル ./img_gray.png が出力されます。

```
> python exer05.py basic/img.png ./img_gray.png
```

※正解データの上書き防止のため、カレントに出力するコマンドを提示しています。

※ basicフォルダ内に正解データがあるので自身で出力結果を確認してください。

img.png
入力



img_gray.png
出力

課題06　画像の2値化

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。実行コマンドは以下の通り。

```
python exer06.py file_in.png thresh file_out.png
```

仕様：

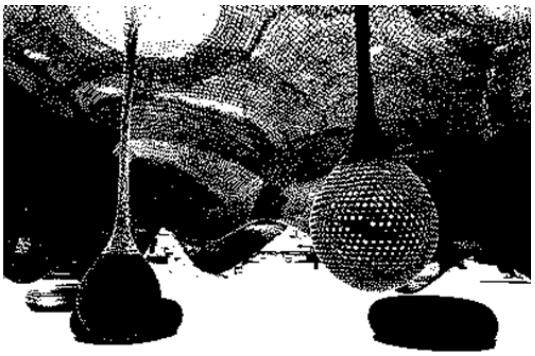
- グレースケール化は前課題のものを利用すること
- 入力ファイル名(file_in.png)，閾値(thresh)、出力ファイル名(file_out.png)は、コマンドライン引数より取得する
- 画素値が閾値以上なら、その画素値を255にする
- 画素値が閾値より小さいなら、その画素値を 0 とする

入出力例 ―----------

bascフォルダ内に入出力例があります。以下のコマンドにより、basicフォルダの basic/img.pngに対して閾値を95として実行すると、img_bin.pngが出力されます。

```
python exer06.py basic/img.png 95 ./img_bin.png
```

入力 img.png



出力 img_bin.png

課題07. 画像の色の変換.

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。実行コマンドは以下の通りとする。

```
python exer07.py file_in.png fname_out.png
```

仕様 :

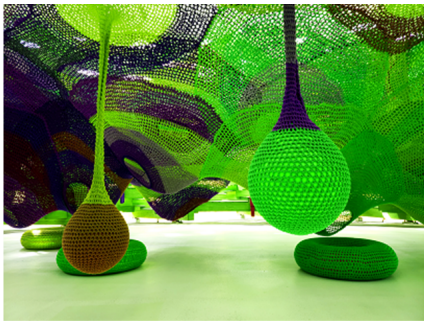
- 入出力ファイル名（file_in.png file_out.png）はコマンドライン引数として取得
- 入力画像の各画素において、赤チャンネルと緑チャンネルの値を入れ替えた画像を出力する。

入出力例 ―----------

bascフォルダ内に入出力例があります。以下のコマンドにより、basicフォルダの basic/img.pngに対してこのプログラムを適用すると、basic/out_grb.pngが出力されます。

```
python exer07.py basic/img.png ./img_grb.png
```

入力 : img.png



出力 : img_grb.png

課題08. 素数の分布画像の作成.

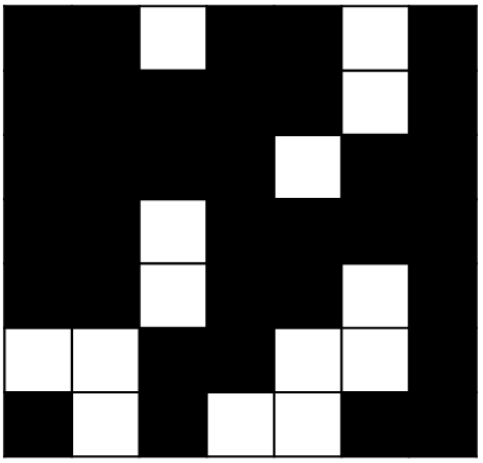
仕様を満たす画像を生成せよ。実行コマンドは以下の通り（Nは画像サイズを表す非負整数値、fname_out.pngは出力ファイル名)。

```
python exer08.py N fname_out.png
```

仕様 :

- 画像サイズは N x N とする（Nはコマンドライン引数より受け取る）
- 下図（左）のように、左下から順番に画素にインデックスを振り、このインデックスが素数の画素は白（値 255）、そうでない画素は黒（値0）とする。

28	35	43	52	62	73	85
21	27	34	42	51	61	72
15	20	26	33	41	50	60
10	14	19	25	32	40	49
6	9	13	18	24	31	39
3	5	8	12	17	23	30
1	2	4	7	11	16	22



- インデックスは、下辺が起点、左辺が終点となるように、左上方向にインクリメントされる。また、左辺まで到達したらそのインデックスに 1 を加えて、一つ右の列にインデックスを配置する構造とする。

入出力例 —----------

N = 7 のときの画像のインデックスは上図（左）、インデックスが素数の画素を白、そうでない画素を黒とした画像（出力画像）は、上図右となります。

※ Nに大きな値を入れると斜め方向に素数が連続して分布する箇所があることを確認してください。参考 : ウラムの螺旋

basic/exer8_100.png に、N=100として生成した画像があります。

課題09. モザイク画像の作成.

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。実行コマンドは以下の通り。

```
python exer09.py fname_in.png R fname_out.png
```

仕様 :

- 入力ファイル名（fname_in.png）、ブロックサイズ（R）、出力ファイル名（fname_out.png）は、コマンドライン引数として取得
- 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
- 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）が発生する場合は，その領域を平均画素値で塗ること

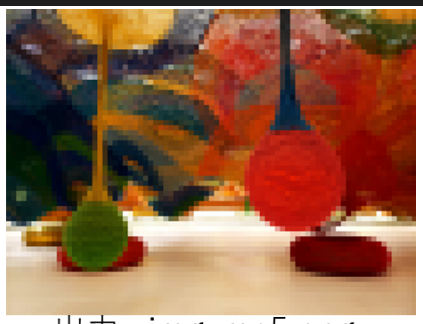
※ヒント : スライスを活用してください。スライスで配列の範囲外を指定しても、その部分は無視されるだけなので、右端や左端の領域についても簡潔に記述できます。

入出力例 ----------

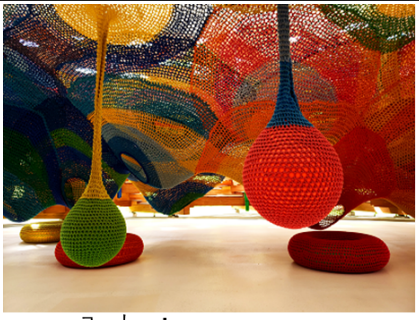
Basicフォルダ内に入出力例があります。以下のコマンドによりbasicフォルダ内の basic/img.png を異なる解像度でモザイク化できます。

```
python exer09.py basic/img.png 5 ./img_mo5.png
```


入力 : img.png


出力 : img_mo5.png

```
python exer09.py basic/img.png 15 ./img_mo15.png
```


入力: img.png


出力 : img_mo15.png

課題10：ガウシアンフィルタ

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適用した画像を保存するプログラムを作成せよ。実行コマンドは以下の通り。

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

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

仕様 :

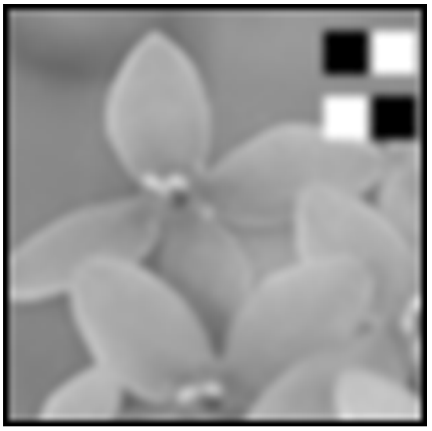
- 入出力ファイル名 (fname_in.png、fname_out.png) はコマンドライン引数より取得
- グレースケール化の方法については雛形を参照
- 右図のガウシアンフィルタを利用する
- 画像の周囲1pixelは計算せず0を入れる
- 今回はプログラミング練習が目的なので、次の関数『cv2.filter2D() / cv2.GaussianBlur() / cv2.Sobel() / np.convolve() 』は利用せず、for文を用いてフィルタ処理を自作すること

入出力例 ―----------

filterフォルダ内に入出力例があります。以下のコマンドによりfilterフォルダ内のfilter/img_small.pngに処理を施すと、filter/img_small_gauss.pngが出力されます。

```
python exer10.py filter/img_small.png ./img_small_gauss.png
```

入力：img_small.png



出力：img_small_gauss.png

課題11：Sobelフィルタ

画像を読み込み、グレースケール画像に変換後、縦方向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。実行コマンドは以下の通り。

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

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

仕様：

- 入出力ファイル名 fname_in.png / fname_out.png はコマンドライン引数より取得
- グレースケール化の方法については雛形を参照
- 画像の周囲1pixelは計算せず0を入れる
- フィルタ適用後，負値となる画素は ー1 倍して正値に変換する、かつ、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、次のフィルタ処理の関数『cv2.filter2D() cv2.GaussianBlur() cv2.Sobel() np.convolve() 』は利用せず、自作すること

入出力例 ー----------

filterフォルダ内に入出力例があります。以下のコマンドによりfilterフォルダ内の
filter/img_small.png に処理を施すと、img_small_sobel.pngが出力されます。

```
python exer11.py filter/img_small.png ./img_small_sobel.png
```

入力 img_small.png

出力 img_small_sobel.png

課題12: 迷路 (発展)

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。実行コマンドは以下の通り。

```
python exer12.py meiro.bmp
```

仕様

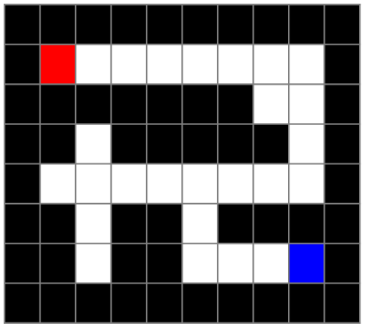
- 迷路の仕様は以下の通り
    - 迷路は画像で与えられる
    - 黒画素 (r,g,b) = (0, 0, 0)は壁で通れない
    - 白画素 (r,g,b) = (255,255,255)は通れる通路
    - 赤画素 (r,g,b) = (255, 0, 0)はスタート
    - 青画素 (r,g,b) = (0, 0,255)はゴール
    - ある白画素から一歩で，上下左右の白画素に移動できる
- 画像ファイル名はコマンドライン引数より取得する
- 計算結果は標準出力に表示する

※ 再帰関数は利用しないでください。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

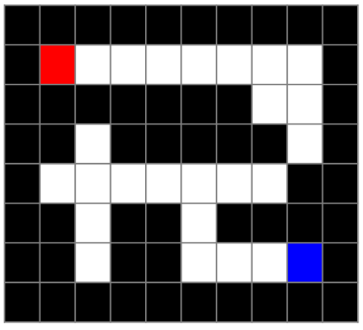
※ meiroディレクトリ内に迷路画像サンプルがあります

※ 雛形なし

※ 発展課題は、講義中に教員 or TAに見せ、チェックを受けてください。



正解は18



正解は－1

課題13: 勾配強度画像の生成

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。実行コマンドは以下の通り。

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

仕様:

- 入出力ファイル名 （fname_in.png fname_out.png） はコマンドライン引数より取得
- グレースケール化の方法については雛形を参照
- 画像の周囲1pixelは計算せず0を入れる
- 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
- ※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

入出力例 ―----------

filterフォルダ内に入出力例があります。以下のコマンドによりfilterフォルダ内の filter/img_small.png に処理を施すと、img_small_gm.pngが出力されます。

```
python exer13.py filter/img_small.png  ./img_small_gm.png
```

入力: img_small.png

出力: img_small_gm.png

課題14: メディアンフィルタ

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。実行コマンドは以下の通り。

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

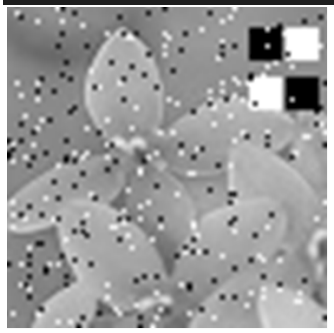
仕様:

- 入出力ファイル名 （fname_in.png fname_out.png）はコマンドライン引数より取得
- グレースケール化の方法については雛形を参照
- フィルタの窓サイズは 5 x 5 とする
- 画像の周囲2pixelは計算せず0を入れること
- cv2の関数『cv2.medianBlur』は利用しないこと
- 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）

入出力例 ----------

filterフォルダ内に入出力例があります。以下のコマンドによりfilterフォルダ内の filter/img_noise1.png や filter/img_noise2.pngに処理を施すとノイズ除去を行えます。

```
python exer14.py filter/img_noise1.png ./img_noise1_med.png
```



入力img_noise1.png

出力img_noise1_med.png

```
python exer14.py filter/img_noise2.png ./img_noise2_med.png
```

入力 img_noise2.png

出力 img_noise2_med.png

課題15：ランレングス符号化

4種類のアルファベット {a, b, c, d} から構成される文字列を読み込み、ランレングス符号化をせよ。実行コマンドは以下の通りとする。

```
python exer15.py fname_in.txt fname_out.txt
```

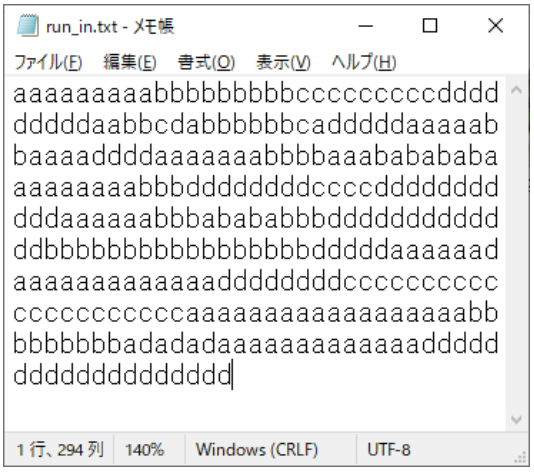
仕様

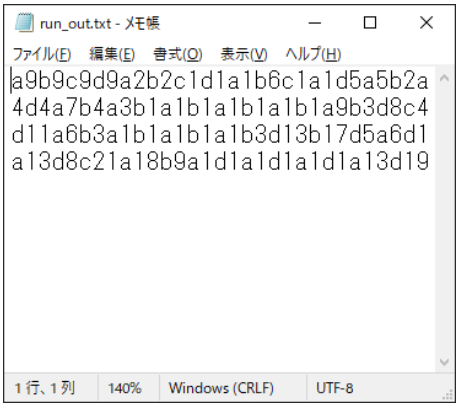
- 入力テキストファイル名（fname_in.txt）、出力テキストファイル名（fname_out.txt）はコマンドライン引数より取得する
- 入力テキストファイルには、4種類のアルファベットからなる文字列が格納される (スペースはなく、改行コードは最後に一つだけ。)
- 出力ファイルには、ランレングス符号化結果（アルファベットと連続数）を書き込むこと。
    - 本来は、バイナリデータを出力する必要があるが本課題では文字列により符号化結果を出力すること。
    - この課題では連続数の上限を考慮しなくてよい（連続数部分に割り当てられたビット数により連続数として表現できる上限が決まるが、ここでは無視してよい）
    - 例: 入力が『aaaaabbccaaaaad』ならば『a5b2c2a5d1』と出力すること

入出力例 ―----------

入出力例をbasicフォルダのrun_in.txtとrun_out.txtに示します。

```
python exer15.py basic/run_in1.txt ./run_out1.txt
```





課題16：ハフマン符号化 （発展）

8種類のアルファベット {a, b, c, d, e, f, g, h} から構成される文字列を読み込み、各アルファベット出現確率に基づきハフマン符号化を行い、各アルファベットの符号化結果を出力するプログラムを作成せよ。実行コマンドは以下の通りとする。

```
python exer16.py fname_in.txt fname_out.txt
```

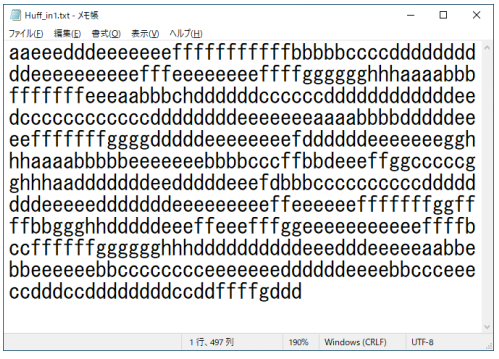
仕様

- 入力テキストファイル名（fname_in.txt）、出力テキストファイル名（fname_out.txt）はコマンドライン引数より取得する
- 入力テキストファイルには、8種類のアルファベットからなる文字列が格納される
- 出力ファイルの各行には、アルファベット｛a, b, c, d, e, f, g, h｝のハフマン符号化結果を順番に書き込む
- 符号化結果は｛0, 1｝を用いて表現すること（本来はビット列だが簡単のため）
- ハフマン木を生成後、出現確率の大きな方に『0』を、小さな方に『1』を割り当てること
- プログラミング課題なので、ハフマン符号化そのものを行うライブラリ関数は試用せず、符号化処理を自作すること

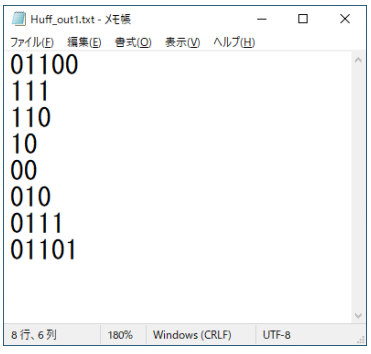
入出力例 ―----------

入出力例をbasicフォルダのHuff_in1.txtとHuff_out1.txtに示します．入力ファイルには、8種類のアルファベットからなる文字列が入っており、出力ファイルには各行にa~fそれぞれの符号化結果が記載されています。

```
python exer16.py baskc/Huff_in1.txt ./Huff_out1.txt
```



Huff_in1.txt



Huff_out1.txt

※ 発展課題は、講義中に教員 or TAに見せ、チェックを受けてください。

課題17: ハーフトーン処理 ディザ法 :

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

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・保存するプログラムを作成せよ。実行コマンドは以下の通り。

```
python exer17.py fname_in.png fname_out.png
```

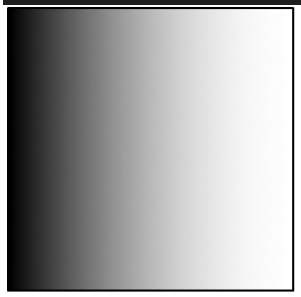
仕様

- 入出力ファイル名（fname_in.png、fname_out.png）はコマンドライン引数より取得
- 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，右上図のディザパターンを利用すること
- 画素値 x を 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，255を代入する
- 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．この領域についても、そのままディザパターンを重ね合わせて計算すること。

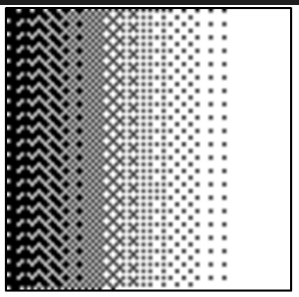
入出力例 ―----------

htフォルダ内に入出力例があります。以下のコマンドによりhtフォルダ内の ht/grd.png や ht/cat.png に処理を施すとハーフトーン処理結果が出力されます。

```
python exer17.py ht/grd.png ./grd_dither.png
```



入力 grd.png



出力 grd_dither.png

```
python exer17.py ht/cat.png ./cat_dither.png
```

入力 cat.png

出力 cat_dither.png

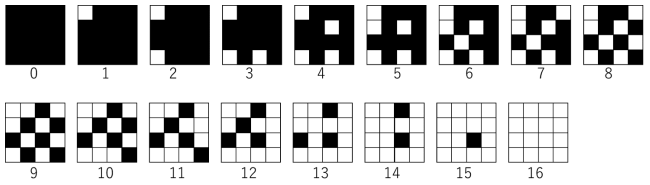
課題18: ハーフトーン処理 濃度パターン法

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・保存するプログラムを作成せよ。実行コマンドは以下の通り。

```
python exer18.py fname_in.png fname_out.png
```

仕様:

- 入出力ファイル名 (fname_in.png fname_out.png)は、コマンドライン引数より取得
- 出力画像は2値画像（0か255）とする
- ブロックサイズは４x 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。4x4ブロックの平均画素値xが，i×255 / 17 ≤ x < (i+1) x 255 / 17 の範囲に入っていれば，下図の i 番目のパターンを適用する

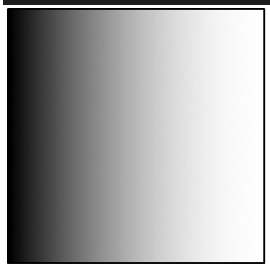


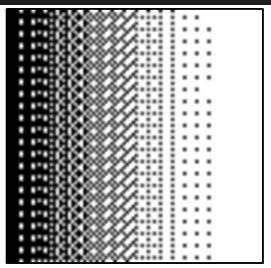
- 画像の幅・高さが4の倍数でない場合、画像の右端・下端に余る領域が発生する．この領域は計算せず0を代入すること。

入出力例 —----------

htフォルダ内に入出力例があります。以下のコマンドによりhtフォルダ内の ht/grd.png や ht/cat.png に処理を施すとハーフトーン処理結果が出力されます。

```
python exer18.py ht/grd.png ./grd_noudo.png
```


入力 grd.png


出力 grd_noudo.png

```
python exer18.py ht/cat.png ./cat_noudo.png
```


入力 cat.png


出力 cat_noudo.png

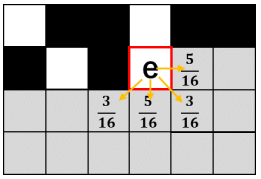
課題19: ハーフトーン処理　誤差拡散法

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生成・保存するプログラムを作成せよ。実行コマンドは以下の通り。

```
python exer19.py fname_in.png fname_out.png
```

仕様 :

- 入出力ファイル名 (fname_in.png fname_out.png)は、コマンド引数より取得
- 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より大きい場合はドライその画素を255に，そうでない画素を0にする

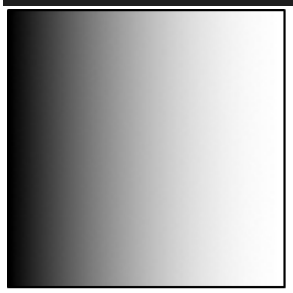


- 拡散する誤差の割合と拡散先は右図の通りとする
- 右端の画素を計算する際、その右隣に画素が存在しないため右へ誤差を拡散できない．この場合は、単純に右隣への誤差拡散を省略せよ．左端・下端の画素の誤差拡散についても同様に拡散を省略すること。（右へ送るべき誤差を他の画素へ拡散する必要はない）

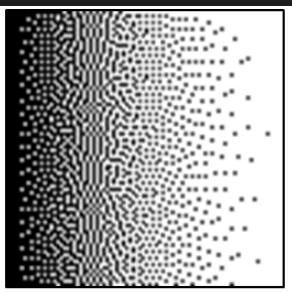
入出力例 ―----------

htフォルダ内に入出力例があります。以下のコマンドによりhtフォルダ内の ht/grd.png や ht/cat.png に処理を施すとハーフトーン処理結果が出力されます。

```
python exer19.py ht/grd.png ./grd_err.png
```



入力 grd.png



出力 grd_err.png

```
python exer19.py ht/cat.png ./cat_err.png
```

入力 cat.png

出力 cat_err.png

課題20: 1次元離散フーリエ変換

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。実行コマンドは以下の通り。

```
python exer20.py sample_fl.txt fname_out.txt
```

仕様 :

- 入出力ファイル名（sample_fl.txt fname_out.txt）はコマンドライン引数として取得
- フーリエ変換の結果得られる周波数係数 Fk 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、Fk = Rk + i Ik と実部Rkと虚部Ikに分けて保持し、それぞれをテキスト形式で出力せよ。
- 入出力の詳細は雛形および出力例に従うこと
- 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　$$F_k = \frac{1}{N}\sum_{l=0}^{N-1} f_l \left(\cos\frac{2\pi kl}{N} - i\sin\frac{2\pi kl}{N}\right)$$

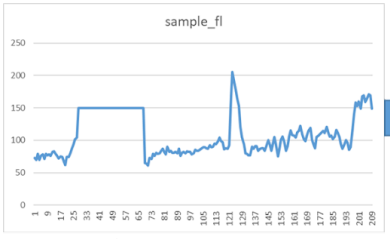
※今回はプログラミング練習が目的なので、フーリエ変換自体を行うライブラリ関数（np.fftなど）は 利用しないこと．math.cos()関数などは利用してよい.

※ ここではnp.radians()は使用しないこと。パイ（π）には、math.piを利用せよ。

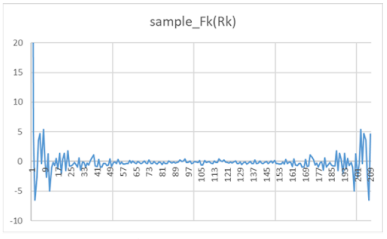
入出力例 ―----------

fourierフォルダに入出力例があります。以下のコマンドで、fourier/sample_fl.txtをフーリエ変換すると、フーリエ変換後の複素数列を格納した output_Fk.txtが出力されます。出力データをexcelで可視化すると以下のようなグラフが得られます。

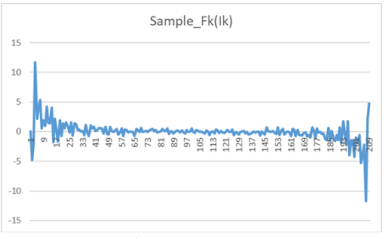
```
python exer20.py fourier/sample_fl.txt ./output_Fk.txt
```



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）

課題21: 1次元 逆離散フーリエ変換

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。実行コマンドは以下の通り。

```
python exer21.py Fk.txt fname_out.txt
```

仕様

- 入出力ファイル名（Fk.txt / fname_out.txt）はコマンドライン引数として取得
- 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，$f_l = r_l + i\,a_l$ と実部r_lと虚部a_lに分けて保持し、それぞれをテキスト形式で出力せよ
- 入出力ファイル形式は雛形と入出力例に従うこと
- 逆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

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

※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)。

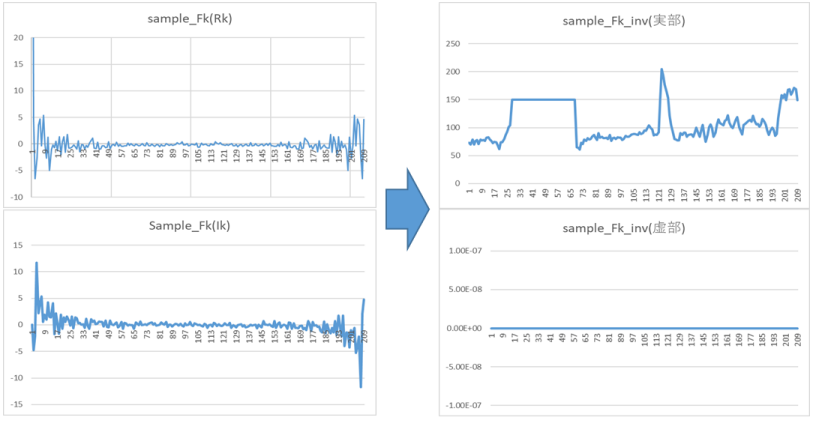
※上式のπは、math.piを利用すると良い。今回はnp.radians()は利用しないこと。

※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

入出力例 ―----------

fourierフォルダに入出力例があります。以下のコマンドで、先の課題の出力 fourier/output_Fk.txtを逆フーリエ変換すると、その結果が output_Fk_inv.txt として出力されます。出力データをexcelで可視化すると以下のようなグラフが得られます。

```
python exer21.py fourier/output_Fk.txt ./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』元の数列に戻ります。sample_Fk_invの虚部はほぼゼロ（非常に小さな値）になります。

課題22: 2次元フーリエ変換

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。実行コマンドは以下の通りとする。

```
python exer22.py fname_in.png Ruv.png Iuv.png
```

仕様

- 入力画像ファイル名（fname_in.png）、出力画像ファイル名（Ruv.png, Iuv.png）はコマンドライン引数より取得
- 結果は $F_{uv} = R_{uv} + iI_{uv}$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。入出力ファイルの詳細は雛形を参照。
- 実部R_{uv}と虚部I_{uv}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（※ RuvとIuvは個別に正規化すること）
- 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： $$F_{uv} = \frac{1}{WH}\sum_{y=0}^{H-1}\sum_{x=0}^{W-1} f_{xy} e^{-\frac{2\pi xu}{W}i} e^{-\frac{2\pi yv}{H}i}$$

※ 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

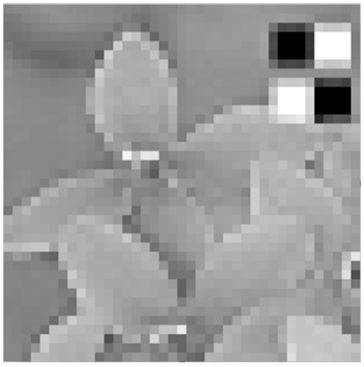
※ 今回はプログラミング練習が目的なので，フーリエ変換自体を行うライブラリ関数（np.fftなど）は利用しないこと（math.cos()関数などはOK)

※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．

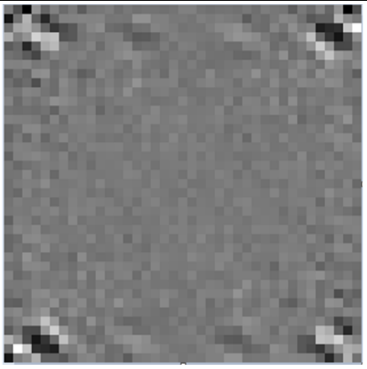
入出力例 ―----------

fourierフォルダに入出力例があります。以下のコマンドで、fourier/img.png をフーリエ変換すると、フーリエ変換後の画像（実部Rvu.png と虚部Ivu）が得られます。

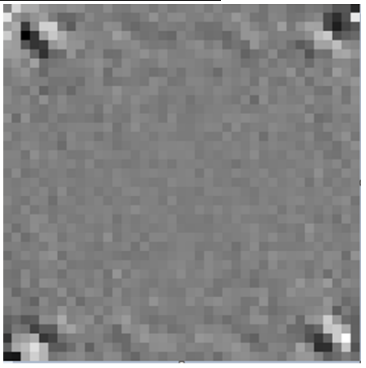
```
python exer22.py fourier/img.png ./Rvu.png ./Ivu.png
```



img.png



Rvu.png



Ivu.png

課題23: Deconvolution (発展)

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するプログラムを作成せよ。実行コマンドは以下の通りとする。

```
python exer23.py input_img.png input_kernel.png
```

仕様

- 入力となる 観測画像ファイル名（Input_img.png）と点広がり関数ファイル名（input_kernel.png ）はコマンドライン引数より取得する
- 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
- 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

※ 発展課題は、講義中に教員 or TAに見せ、チェックを受けてください。
※ こちらは、他の課題が簡単すぎる方のためのものです（これも簡単だったらすみません。。）。