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_lasthalf.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

ハーフトーン処理
課題12~14

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	5
3	11	1	9
12	4	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）

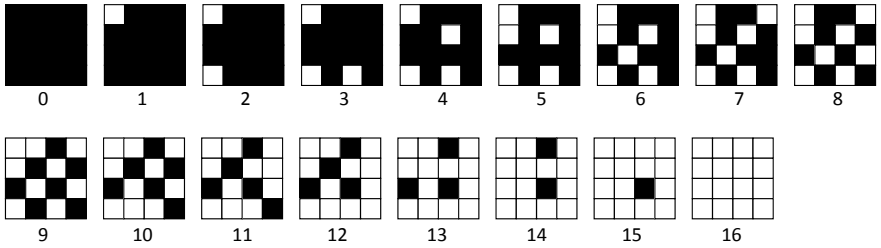
画像を読み込み，グレースケール画像に変換後，濃度パターン法により濃淡を維持した二値画像を作成・保存せよ

- 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*255 ~5/17*255
の範囲なのでパターン4を採用する

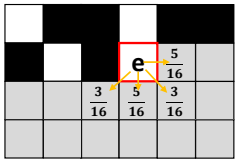
課題14. ハーフトーン – 誤差拡散法（exer14.py）

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
```

- 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などは行わない．
- 下端の画素の誤差拡散についても同様．



誤差拡散する隣接画素

課題12~14の出力例

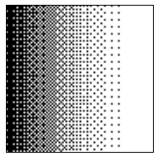
この例にある入力画像・出力画像は雛形フォルダに入っています．これを利用して自身の書いたプログラムが正しく動いているかどうかを確認してください

出力例

課題12, 例1

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

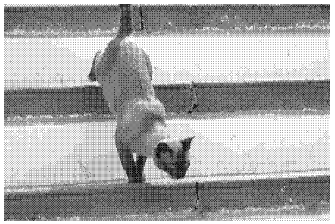

入力画像
grd.png


出力画像
grd_tiza.png

課題12, 例2

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


入力画像
cat.png

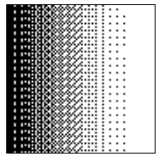

出力画像
cat_tiza.png

出力例

課題13, 例1

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

入力画像
grd.png



出力画像
grd_noudo.png

課題13, 例2

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

入力画像
cat.png

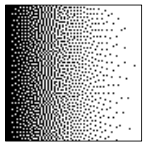
出力画像
cat_noudo.png

出力例

課題14, 例1

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

入力画像
grd.png



出力画像
grd_err.png

課題14, 例2

```
$python  exer14.py  cat.png  cat_err.png
```

入力画像
cat.png

出力画像
cat_err.png

※誤差拡散法ではグリッドパターンが見えないので，よりきれいな結果が得られます

周波数フィルタ
課題15~17

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

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は利用しないこと(もちろん 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）

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

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は利用しないこと(もちろん math.cos()関数などはOK)

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

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，[0,255]に正規化すること（※ 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

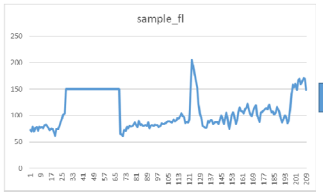
課題15~17の出力例

この例にある入力画像・出力画像は雛形フォルダに入っています．これを利用して自身の書いたプログラムが正しく動いているかどうかを確認してください

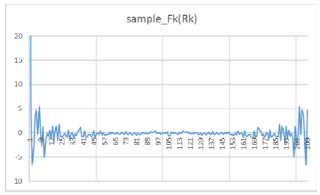
出力例

課題15, sample_fl.txt 内の実数配列をフーリエ変換すると，output_Fk.txt(複素数配列)が得られる

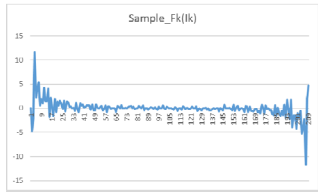
```
$python  exer15.py  sample_fl.txt  output_Fk.txt
```



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）

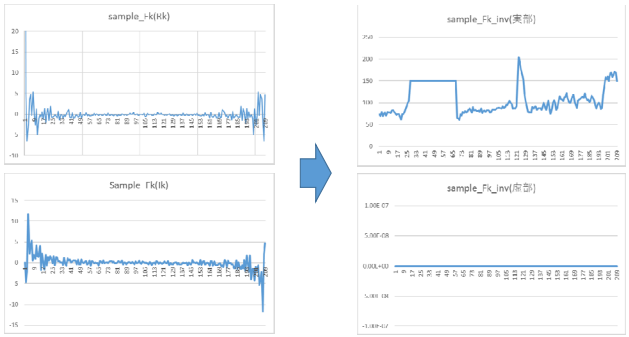


出力の虚部 output_Fk.txt
（見やすさのため値域を[-15,15]にした）

出力例

課題16, 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   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））

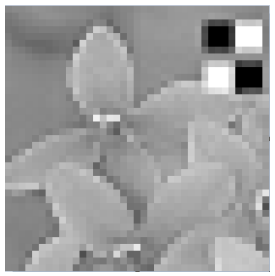
元の実数列（sample_fl）を
フーリエ変換したもの(output_Fk)を
逆フーリエ変換すると(output_Fk_inv)
元の関数に戻ります。

sample_Fk_invの虚部には非常に小さな
値が入ります

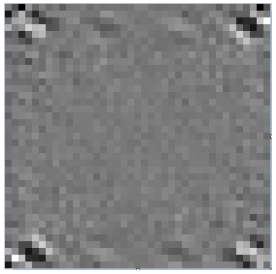
参考 (出力結果の例) ※ファイルはfourieディレクトリにあります

課題17, img.png 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

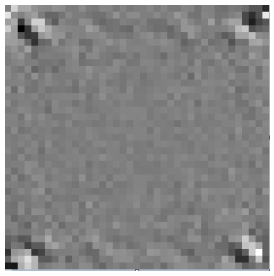
```
$python   exer17.py   img.png   Rvu.png  Ivu.png
```



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
課題18~19

ここまでの課題が簡単すぎた人用

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．また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

※ 名簿の例を“./meibo“フォルダ内に置いておきます（http://www.gaoshukai.com/lab/0003/で作成した架空のデータです）

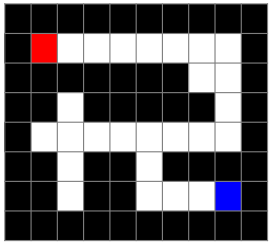
※ 大学教員になって数年目ですが， 学会運営や教育業務のため，機械学習というよりこういう用途にばかりpythonを使っています．

課題19 迷路

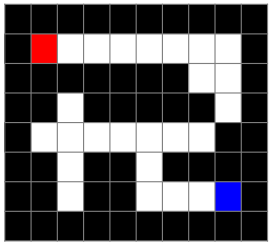
- 迷路を解き，スタートからゴールまでの経路が有る場合は最小歩数を表示し，経路が無い場合は-1を表示するプログラムを作成せよ
- 迷路の仕様は以下の通り
  - 迷路は画像で与えられる
  - 黒画素(r,g,b=　0,　0,　0)は壁で通れない
  - 白画素(r,g,b=255,255,255)は通れる通路
  - 赤画素(r,g,b=255,　0,　0)はスタート
  - 青画素(r,g,b=　0,　0,255)はゴール
  - ある白画素から一歩で，上下左右の白画素に移動できる
- 画像データはコマンドライン引数で受け取り，計算結果は標準出力に表示する

```
$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