補助資料

Windowsユーザ向け Python環境の構築方法

本講義のプログラミング課題を解くための Python入門

担当: 井尻 敬

はじめに

- この資料はPythonを講義で初めて利用する情報工学科学生向けに環境構築方法を紹介するものです.
- 自身で環境を構築できる方，仮想環境を利用したい方などは対象にはしていません（自分で好みの環境を作って講義・演習を実施してください）

手順

1. 準備
2. pythonのインストール
3. 必要なライブラリのインストール

準備 -既存のPythonの削除

- 左下のウインドウズスタートボタン をクリック
- 少し上にある歯車 をクリック
- 『アプリ』をクリック
- 『このリストを検索』窓内に”Python”と記入
  - 「Python *.*.*」や「Python Launcher」があればクリックしアンインストール

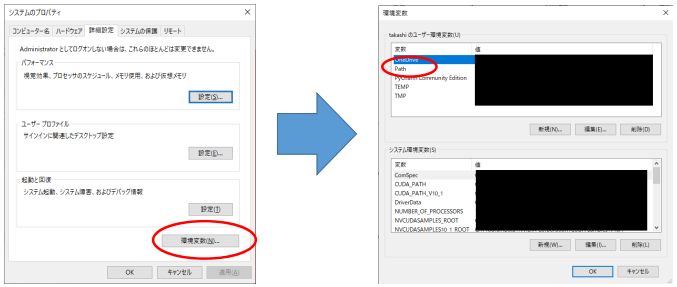
準備 -環境変数: Pathの確認

※コマンドプロンプトにコマンドを打ち込むと，Windowsはそのコマンドに対応する実行ファイルを検索する．検索対象となるフォルダへのpathが，「環境変数のPath」という項目に書かれている．

※以前にPythonをインストールし現在利用していない場合，過去のPathが残っており不具合が起きる可能性があるので確認する必要がある

※不具合が起きる可能性があるのでよくわからなければ何もしない

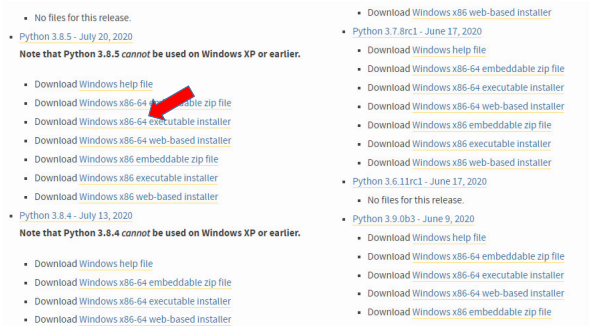
- Windowsのスタートボタンを右クリックし，システムを選択 → 右側の「システム情報」を選択 → 「システムの詳細設定」を選択 → 「環境変数」を選択
- 「Path」を編集状態にして，もしPythonやAnacondaに関するものがあり，過去にインストールしたものなら削除する



Pythonのインストール

1. Pythonのページ（ https://www.python.org/ ）を開く
2. Downloadにカーソルを置き → windowをクリック
3. 下の通り「Download Windows x86-64 executable installer」をクリックしダウンロード

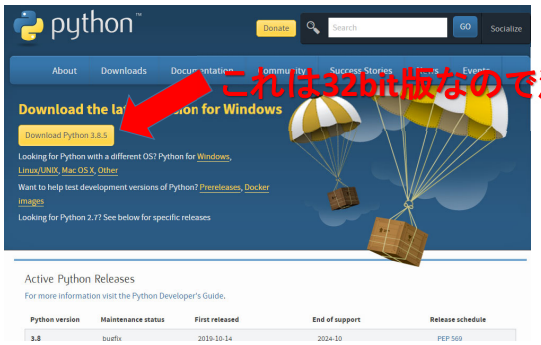
※下はPython 3.8.5を選択したときのもの



Pythonのインストール（注意）

以下のリンクからダウンロードできるのは32bit版．

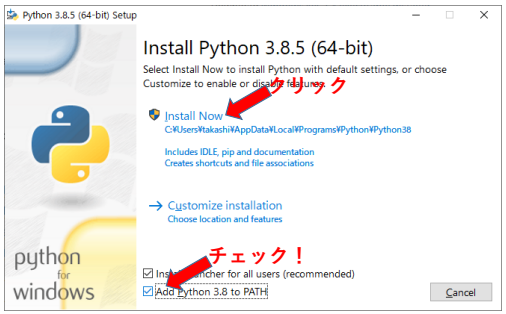
動けばよい方はこれでもOK．機械学習をやる方は，後に問題が起きる（入れられないライブラリがあるとか）ので64bit版を強く推奨．



これは32bit版なので注意！！！！

Pythonのインストール

4. インストーラーを起動する
5. インストーラー起動画面でPath設定（下図参照）にチェックを入れる
6. Install Nowをクリックし，インストールを進める



Pythonの動作確認

1. コマンドプロンプトを起動
   - Windowsの検索ボックス（左下の「ここに入力して検索」）にcmdと打ち込む

2. コマンドプロンプトに「$python -V」と打ち込みpythonのバージョンが表示されればOK
3. コマンドプロンプトに「$pytno」と打ち込みpythonの対話モードが起動するのを確認してもOK (対話モードはCtrl+zで終了できる)

```
C:¥Users¥takashi>python -V
Python 3.8.5

C:¥Users¥takashi>python
Python 3.8.5 (tags/v3.8.5:580fbb0, Jul 20 2020, 15:57:54) [MSC v.1924 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> _
```

必要なライブラリのインストール

- Pythonでは，「pip」コマンドによりライブラリパッケージの管理（インストール・アンインストール）ができる
- コマンドプロンプトに「$pip list」と打ち込むと現在インストールされているパッケージが表示される

```
C:¥Users¥takashi>pip list
Package         Version
--------------- ---------
certifi         2020.6.20
cycler          0.10.0
future          0.18.2
joblib          0.16.0
kiwisolver      1.2.0
Pillow          7.2.0
pip             20.2.3
pyparsing       2.4.7
python-dateutil 2.8.1
scikit-learn    0.23.2
scipy           1.5.2
setuptools      49.6.0
six             1.15.0
threadpoolctl   2.1.0
```

※pipのバージョンが古いとwarningが出る

※初期状態はもう少し入っているパッケージは少ないはず

必要なライブラリのインストール

1. pipのアップグレード．コマンドラインに以下のコマンドを打つ

$python -m pip install --upgrade pip

2. numpy/matplotlib/opencvのインストール．コマンドラインに以下のコマンドを打つ

$pip install numpy

$pip install matplotlib

$pip install opencv-python

3. $pip list　としてパッケージが入ったことを確認

```
C:¥Users¥takashi>pip list
Package         Version
--------------- ---------
certifi         2020.6.20
cycler          0.10.0
future          0.18.2
joblib          0.16.0
kiwisolver      1.2.0
matplotlib      3.3.2
numpy           1.19.2
opencv-python   4.4.0.42
Pillow          7.2.0
pip             20.2.3
pyparsing       2.4.7
python-dateutil 2.8.1
scikit-learn    0.23.2
scipy           1.5.2
setuptools      49.6.0
six             1.15.0
threadpoolctl   2.1.0
```

OpenCVの動作確認

- テスト用ディレクトリを用意
- 画像ファイルを用意し「img.png」という名前でこのディレクトリに保存

※異なる形式・ファイル名を利用するときには右のコードの該当部分を書き換える

- ファイル「test.py」を作成し，右のコードをコピー
- このディレクトリがカレントのコマンドプロンプトを起動（エクスプローラのアドレスバーに「cmd」と書く）
- 「$ python test.py」とコマンドプロンプトに打ち込み画像が表示されたらOK

```
# -*- coding: utf-8 -*-
import cv2

img = cv2.imread("img.png")
print(img.shape)
cv2.imshow("sample", img)
cv2.waitKey(0)
```

本講義のプログラミング課題を解くための Python入門

※ 複数人で協力して実習・課題を進めることを奨励します
- 教わる方は，何がわからないかを言語化できるようになってください
- 教える方は，何がわかってないかを引き出してください
- 教えるのも教わるのも非常に良い勉強になります

※ 分からない事や気になる事があれば，井尻やTAに聞いてください

Python

- 最近流行りのスクリプト言語
- 機械学習関連のライブラリが充実
- 画像処理関連のライブラリも充実（OpenCV）
- 開発コストが低い（井尻が普段利用しているC++に比べて）

- コードの可読性が高い
- インデントでブロックを強制
  → 変なコードが生成されにくく，学生のコードを読む側としてはとてもありがたい．

準備

- Python 3のインストールされたマシンを用意する
  - 学情PCはそのまま利用できます
  - python起動のコマンドは、『python』ではなく『py』です
  - 自分のWindows PCを利用する人は前述の方法でインストールしてください
- 作業ディレクトリを用意してください
  - どこでもよいです
- サンプルコードを講義web pageよりダウンロードしてください
  - この資料を写経してもよいのですが面倒だと思うので用意しておきました

Ex1.py “Hello world”

実習: pythonで書かれた右のコードを動かしてください

0. 作業用ディレクトリを作成
1. “ex1.py”というファイルを作成し作業ディレクトリに配置
2. “ex1.py”に右のコードを記入
3. コマンドプロンプトを起動し，作業ディレクトリへ移動

※次ページ参照

4. コマンドプロンプトにおいて，以下のコマンドを入力

```
> python ex1.py
```

5. hello, worldと出力されたら成功

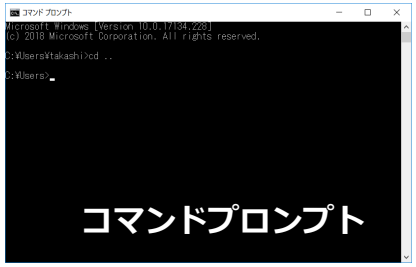
※大学のPCを利用する場合
> py ex1.py
としてください

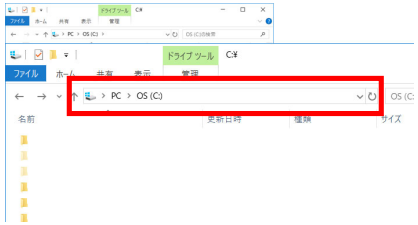
```
# ex1.py

print("hello, world")
```

※ 『#』でその行をコメントアウト
※ 『print(“文字列”)』で文字列を出力
※ 『# -*- coding: utf-8 -*-』は文字コード指定．日本語を利用可能に．

コマンドプロンプトについて

- 使ったことがない人もいると思うので解説します
- コマンドプロンプトとは，コマンドで実行ファイルを起動できるCUIアプリです（windowsにおけるUnixターミナルみたいな認識でOKです（右図））
- Windowsのタスクバーの検索ウインドウに『cmd』と打ち込むと起動できます
- Unixのターミナル同様に『cd』コマンドでディレクトリを移動できます
- ディレクトリ内のフォルダ・ファイルを参照するには『dir』コマンドを打ち込みます（UnixのlsのようなもOの）
- エクスプローラのアドレスバー（右図）に『cmd』と書いてエンターを押すと，開いたディレクトリをカレントとするコマンドプロンプトを起動されます（とても便利！）





Ex2.py 変数の型

- **int, float, string, boolなどの型を利用可能**
- **変数の型は代入する値に応じて自動で決まる**
  （型を明示した変数宣言は行わない!）
- **後から異なる型に変更することも可能**
  （その都度新しい変数が生成される!）

実習：右のコードを動かしてみてください

実習：右のコードを色々と編集し型の挙動を確認してください

※ type（変数名）：変数型を取得する関数

※ id（変数名）　：オブジェクトidを取得する関数（idを見ると，数値代入のたびに新たなオブジェクトが生成されているのが分かる．意味が分からない人は，とりあえず無視してOK．）

※ print（変数1, 変数2）で複数変数を出力可能

※ 型変換も可能

```
# ex2.py

#int
a = 1234
print(a, type(a),id(a))

#float
a = 1.234
print(a, type(a),id(a))
a = 1.2345
print(a, type(a),id(a))
a = 1
print(a, type(a),id(a))

#bool
a = True
print(a, type(a),id(a))

#string
a = "hello, world"
print(a, type(a),id(a))

#型変換例: float->int, string->int, int->float
a = int(16.2)
a = int('16')
a = float(16)
```

始めにC言語を学んだ皆さんからすると…

- 型指定が暗黙的に行なわれる
- int型の変数にfloat型を突っ込むとfloat型になる

あたりが気持ち悪いと感じるかもしれません

これは以下の二つの機能に拠るものです

型推論：変数の型を明示的に指定しなくても，コンパイラやインタープリタが型を決めてくれる機能

例）pythonの変数 や C++のautoなど

動的型付け：変数の型が実行時に決まる機能．柔軟な書き方ができる一方で．一つの変数に複数の役割を持たせるなど複雑なことをすると実行中に型が分からなくなるようなこともありうるので注意が必要．．

```
# ex2.py

#int
a = 1234
print(a, type(a),id(a))

#float
a = 1.234
print(a, type(a),id(a))
a = 1.2345
print(a, type(a),id(a))
a = 1
print(a, type(a),id(a))

#bool
a = True
print(a, type(a),id(a))

#string
a = "hello, world"
print(a, type(a),id(a))

#型変換例: float->int, string->int, int->float
a = int(16.2)
a = int('16')
a = float(16)
```

Ex3.py コマンドライン引数

コマンドライン引数とは，コマンドラインからpythonを起動する際に，下のように与える引数の事です

```
> python ex3.py  arg1  arg2 5
```

この例では，3個の文字列『arg1』『arg2』『5』を引数として与えています

実習: 3個の引数を受け取る右のコードの動作を確認してください．また，引数を変化させてみてください．実行コマンドは以下の通り;

```
> python ex3.py  aaa  bbb ccc
```

```
# ex3.py

import sys

a1 = sys.argv[1]
a2 = sys.argv[2]
a3 = sys.argv[3]
print(a1, a2, a3)
```

※ 『import sys』は引数読み込みのためのライブラリを利用する準備

※ 『sys.argv[i]』にi番目の引数が入る

※ 引数は文字列型

Ex4.py 配列

Pythonでは, tuple / list / np.array という3種類の配列表現が利用可能

(1,2,3)　　tuple　　：**長さ&値 変更不可**の配列
[1,2,3]　　list　　：可変長配列（要素を後から追加削除可）
np.array(1,2,3)　　np.array: *n*次元配列（画像などはこれで表現される）

- np.arrayは高速処理のための制約がある配列
  - np.array では要素がメモリ内の連続領域に配置される
  - np.array では各次元の要素数は等しい（行列の形になる）
  - np.array では原則的に要素は同じ型
  - 参考：http://www.kamishima.net/mlmpyja/nbayes1/ndarray.html

Ex4.py 配列

実習：右のコードの出力を予想してください

output1
output2
output3
outout4

実習：コードを実行し，予想と比べてください
実習：コードの中身を色々変化させ，配列とタプルの挙動を確認してください

※ 配列は角括弧 [ ] で表現される
※ tupleは丸括弧 ( ) で表現される
※ 配列要素の変更・追加・削除ができる
※ len(配列名)で長さを取得できる
※ 2次元配列（行列）も表現可能

```
# ex4.py

#1D array
A  = [1,3,5,7]
N  = len(A)  #要素数
print("output1:", A, N )

a2 = A[2]     #2番目の要素を参照
A.append(4)  #後ろに"4"を挿入
a = A.pop(2) #2番目の要素をpop
A.remove(3)  #値が3の最初の要素を削除
print( "output2", a, a2, A )

#2D array
A = [[1,2],[3,4],[5,6]]
print( "output3", A[0][1], A, len(A))

#tupple
T = (1,2,3)
# T[1] = 2  #error tupleは変更不可
print("output4",T, T[1])
```

Ex5.py np.array

np.arrayを含むコードを右に示す

実習：右のコードにprint文を挿入し，C,D,…,Iの計算結果を確認してください

実習：コードの中身を色々変化させ，np.array の挙動を確認してください

※『import numpy as np』はnumpy関連モジュールを利用する準備

※ python & openCV環境では，np.arrayで画像を表現

※ 要素ごとの演算が一行で書ける（画像と画像の和など）のでとても便利

※ **np配列名.shape**で配列サイズを取得

```
# ex5.py
import numpy as np

A = np.array([5,6,7,8])
B = np.array([1,2,3,4])
print(A.shape, A)
print(B.shape, B)

#要素ごとの演算(和差積商余べき)
C = A + B
D = A - B
E = A * B
F = A / B
G = A % B
H = A ** B
I = A + 3  #スカラーとの和

#2D
A=np.array([[1,2,3],[4,5,6]])
B=np.array([[1,2,3],[4,5,6]])
C=A+B
print(C,C.shape)
```

Ex6.py np.array

要素の総和・平均・分散の計算など，多様な便利機能が用意されている

Np.array には便利な初期化方法が用意されている

実習：右のコードを実行し結果を確認してください
実習：配列の分散が計算されていることを確認してください

※右の構文は，無理に覚える必要がありません．このスライドを辞書として使ってください

```
# ex6.py
import numpy as np

A    = np.array([1,2,3,4,5,6,7,8])
mean = np.mean( A ) #全要素の平均
sum  = np.sum ( A ) #全要素の総和
vari = np.var ( A ) #全要素の分散
print( mean, sum, vari)

#分散は以下の方法でも計算可能
A = A-mean # 全要素からmeanを引く
A = A**2   # 全要素を二乗
print( np.sum(A)/A.shape[0])

#np.arrayの初期化方法
A = np.array([1,2,3,4]) #listで初期化
B = np.array((1,2,3,4)) #tupleで初期化
C = np.zeros(3)         #要素数指定，要素は0
D = np.ones(3)          #要素数指定，要素は1
E = np.ones((2,3))      #[[1,1,1][1,1,1]]
F = np.zeros_like(E)    #Eと同じサイズの0配列
F = np.identity(3)      #3x3 単位行列
```

Ex7.py　for文

実習：コードを実行し結果を確認してください

実習：右のコードを少し変更し，for分を使って Aの分散を計算してください

※インデントによりブロックを定義する （Cでは{}でブロックを定義した）

※インデントは半角スペース4個を推奨

※ブロック開始部分に 『:』が必要

※ 『for p in A :』でAのすべての要素に順にアクセスできる

※ 『for i in range(a, b) :』で i = a~b-1をループできる

```
# ex6.py
import numpy as np

A = np.array([1,2,3,4,5,6,7,8])

#Aの全要素をまわる
sum = 0
for p in A:
    print(p)
    sum += p

print( sum / A.shape[0])

#添え字を利用し要素を参照する
sum = 0
N   = A.shape[0]
for i in range(N):
    print( A[i] )
    sum += A[i]

print(sum / A.shape[0])
```

インデントについて

大事な事なので繰り返します

- **Pythonではインデントによりブロックを定義する**
  - **Pytyonでは右の場所がブロック**
  - **参考: Cでは{}でブロックを定義した**

```
for( i=0;i<N;++i){
   // Cではここがブロック
}
```

- **インデントは半角スペース4個を推奨**
- **インデント内にスペースとタブが混在するとエラーが出るので注意**
- **※ if文やfor文はスコープを作らないのでブロック内で定義した変数を外から参照できる（講義中に説明します）**

```
A = [1,2,3,4,5,6,7]
N = len(A)
sum = 0

for i in range(N):
    print(i)
    print(A[i])
    sum += A[i]

print(sum)
```

ブロック

Ex8.py　if文

- if 文は右のコードの通り定義できる
- else if (条件) は elif 条件: と書く
- for文と同様にインデントでブロックを定義する

実習：右のコードを実行し挙動を確認してください

※『AかつB』　→ if 条件A and 条件B:

※『AまたはB』→ if 条件A or 条件B:

```
# ex8.py
import numpy as np

A = [1, 2, 4, 2, 1, 1, 3, 4]

for p in A :
    if p == 1:
        print( "a" )
    elif p == 2:
        print( "b" )
    else:
        print( "c" )
```

Ex9.py　関数

実習：コードを実行してください.

実習：aとbの積も返すよう関数を修正してください

※以下の構文で関数を定義できる

```
def 関数名 (引数1, 引数2) :
    処理1
    処理2
      :
    return 変数
```

※関数は複数の引数を受け取れ, **複数の戻り値を返せる**

※関数はスコープを作る：関数内部で定義した変数は外に漏れない

※引数は参照渡し

- ただし, 組み込み型int, float, bool, str, tuple, unicodeは、値渡しのように振舞う）
- ある引数aがあるとき，aに代入をしない場合はaへの参照が保持される．関数内でaへの代入が起こると，その瞬間に新たに変数aが生成される．そのため，関数外部からみると引数変数の変化は起きないため，値渡しのように見える．
- 意味が分からない人はとりあえず無視してOK
- 講義中に詳しく説明する予定

```
# ex8.py
import numpy as np

def func( a, b):
    print("a is ", a)
    print("b is ", b)
    wa = a+b
    sa = a-b
    return wa, sa

a = 1
b = 2
sum, sub = func(a,b)

print( a,b, sum, sub)
```

ex9main.py main 関数 (使わないので飛ばしてもOK)

Pythonでは，スクリプトが.pyファイルの上から順に実行される

ある.pyファイル内に定義された関数を，他の.pyファイルから呼び出せる(importできる)．このとき，関数だけ読み込みたいのに，importしたファイルのスクリプトが実行されてしまうと困る

スクリプト部分を『if __name__ == '__main__':』に入れると，外からimportされた時には実行されない

```
# -*- coding: utf-8 -*-
# ex9.py
import numpy as np

def func(a,b) :
    print("a is ", a)
    print("b is ", b)
    wa = a+b
    sa = a-b
    return wa, sa

sum, sub = func(1,2)
print(sum, sub)
```

```
# ex9main.py
import numpy as np

def func(a,b) :
    print("a is ", a)
    print("b is ", b)
    wa = a+b
    sa = a-b
    return wa, sa

if __name__ == '__main__':
    sum, sub = func(1,2)
    print(sum, sub)
```

PythonとOpenCVを利用した画像処理

- OpenCVとは
  - Open sourceの画像処理ライブラリ群
  - 多様な画像処理ツールを提供する
  - BSDライセンス：『「無保証」であることの明記と著作権およびライセンス条文自身の表示を再頒布の条件とするライセンス規定』(https://ja.wikipedia.org/wiki/BSDライセンス より)
- C++, Python, java, unityなどから利用可能

※以降のコードは学内環境にて動作することを確認しています
※もし途中で落ちる場合は画像データの読み込みに失敗している場合があります
ファイル名や配置するフォルダを確認してください

Ex10.py 画像の入出力

実習:

- **適当な画像を準備し，名前を「img.png」としてコードと同一フォルダに配置してください**
- **コードを実行し画像が表示/保存されることを確認してください**

※**cv2.imread**("ファイル名")で画像読み込み
※**cv2.imshow**("キャプション", img)で画像表示
※**cv2.imwrite**("ファイル名", img)で画像書き出し

※画像は np.array 形式で表現されます
img.shape : 画像サイズ
→ 幅128, 高さ512, カラー画像なら img.shapeは (512, 128, 3)というタプルになる
img.dtype : 画像データの型

```
# ex10.py
import numpy as np
import cv2

#load image
img = cv2.imread("img.png")
print( img.shape, type(img), img.dtype )

#save image
cv2.imwrite("img_save.png", img);

#display img
cv2.imshow("show image", img)
cv2.waitKey()
```

Ex11.py 画像に図形を書き込む

実習:

- **コードを実行し画像に図形が書き込まれることを確認してください**
- **注意)img.pngが小さいとうまく書かれない**

※手軽に画像への書き込みが行なえます
cv2.line　　(画像, 点1, 点2, 色, 太さ)
cv2.rectangle(画像, 点1, 点2, 色, 太さ)
cv2.circle　　(画像, 中心, 半径, 色, 太さ)

※その他の図形描画関数は以下を参照
http://docs.opencv.org/2.4/modules/core/doc/drawing_functions.html

```
# ex11.py
import numpy as np
import cv2

#load image
img = cv2.imread("img.png")
print( img.shape, type(img), img.dtype )

#draw rect and dots
cv2.line      (img,(100,100),(300,200), (0,255,255),2)
cv2.circle    (img,(100,100), 50,       (255,255,0),1)
cv2.rectangle(img,(100,100),(200,200), (255,0,255),1)

#display img
cv2.imshow("show image", img)
cv2.waitKey()
```

Ex12.py 画素へのアクセス

実習：右のコードを動かして，画像のred値を取り出した画像が表示されることを確認して下さい

課題: 右のコードの一部を編集し画像をグレースケール化してください

- グレースケール値は，ｒｇｂの平均とする
  l = (r+g+b)/3
- 画像の(y,x)画素の(r,g,b)値は
  r = img[y,x,2]
  g = img[y,x,1]
  b = img[y,x,0]

※途中計算時のオーバフローを避けるため，画像 imgと img_grayは，float型に変換されており，可視化時にuint8型に変換されている．

img = np.float64(img) #float64に変換
img = np.uing8 (img) #uint8に変換

```
# ex12.py
import numpy as np
import cv2

#load image
img = cv2.imread("img.png")
img = np.float64( img ) #要素をfloat型に
H   = img.shape[0]
W   = img.shape[1]

img_gry = np.zeros( (H,W), float ) #空の画像を用意

for y in range(H) :
    for x in range(W) :
        r = img[y,x,0]   #画素(y,x)のred値
        g = img[y,x,1]   #画素(y,x)のgreen値
        b = img[y,x,2]   #画素(y,x)のblue値
        img_gry[y,x] = r #red値を代入

#display img
cv2.imshow("gray  image", np.uint8( img_gry) )
cv2.waitKey()
```

Ex13.py 画像の作成

実習：右のコードを実行し，縦じま画像が現れることを確認してください

※グレースケール画像は2次元行列となります

スライス表現

※ img[10:20, 30:40] とすると
y = 10~19, x=30~39
の矩形画像にアクセスできます

※ img[10:20, 30:40] = 10 とすると矩形領域を値10で埋められます

```
# ex13.py
import numpy as np
import cv2

#サイズ200x200の画像を作成
N = 200
img = np.zeros( (N,N), np.uint8 )

#画素値代入(縦じま)
for y in range(N) :
    for x in range(N) :
        if( x % 2==0) :
            img[y,x] = x
        else :
            img[y,x] = 255

#矩形領域に一度で代入も可能(スライス表現)
img[50:60, 50:70] = 255

#display img
cv2.imshow("image", np.uint8(img) )
cv2.waitKey()
```

スライス

スライスとは，配列のある部分にアクセスできる表現です．便利なので使って慣れてください．

右はサンプルコードです

※配列 a=[1,2,3,4,5,6,7] の2番目の要素から5番目の要素までの連続部分を取り出したい場合には

a[2:6]

とします．この a[2:6] には値の代入も可能です．

※ 2次元配列 imgに対して，

img[10:20, 30:40]

とすると y=10~19, x=30~39 の矩形画像にアクセスできます

※ img[10:20, 30:40] = 10 とすると矩形領域を値10で埋められます

```
# ex13slice.py
import numpy as np
import cv2

#1D
a1 = np.array([0,1,2,3,4,5,6,7,8,9])
a2 = np.array([2,2,2,2,2,2,2,2,2,2])

#2番目から5番目の要素を取り出す
print(a1[2:6])

#2番目から5番目に代入にする
a1[2:6] = a2[2:6]
print(a1[2:6])

#2D
#100x100の配列を作製
img1 = np.zeros( (100,100), np.uint8 )

#x=10～50, y=20～30の部分を取り出す
img2 = img1[20:31,10:51]
print(img2.shape)

#x=10～50, y=20～30の部分を白く塗る
img1[20:31,10:51] = 255

cv2.imshow("image", np.uint8( img1) )
cv2.waitKey()
```

Ex14.py 平滑化フィルタ

- **実習：右のコードの一部を編集し,平滑化フィルタを完成させよ．**
- **ただし，『注目画素を中心とする9画素の平均値』を注目画素に格納するものとする**

```
# ex14.py
import numpy as np
import cv2

#load image, convert to grayscale float
img = cv2.imread("img.png")
img = cv2.cvtColor(img, cv2.COLOR_RGB2GRAY)
img = np.float64(img)

img_out = np.zeros_like( img )

for y in range( 1, img.shape[0]-1 ):
  for x in range( 1, img.shape[1]-1 ):
    # ここを編集し平滑化フィルタを完成させる
    img_out[y,x] = 128

cv2.imshow( "output", np.uint8( img_out ) )
cv2.waitKey()
```

まとめ

- 画像処理プログラミングの雰囲気を味合うために，Python & OpenCV環境で，初歩的なコードを書いた.

- このPython & OpenCV環境は，比較的手軽にプロトタイピングが行えるので，興味がある方は是非学修を進めてほしい