知能プログラミング演習 I 演習課題

1 準備

1.1 Anaconda/spyder 用

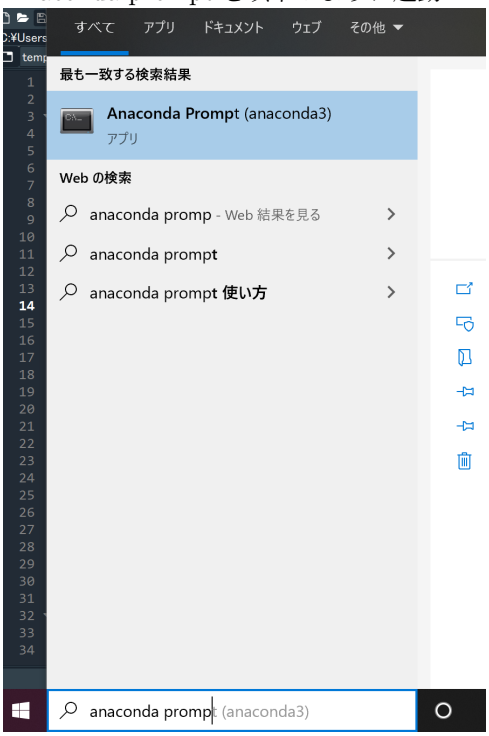
1.1.1 課題ファイルのダウンロードと python のインストール

- 適当なフォルダに今日の課題をダウンロードし, 展開する
- 展開したフォルダの中に, 以下のものがすべて入っていることを確認
  - algebra.py
  - perceptron.py
  - task.pdf
- anaconda をインストールする
  - Anaconda（Python3）インストール手順< Windows 用>
    `https://sukkiri.jp/technologies/ides/anaconda-win_install.html`
  - Anaconda（Python3）インストール手順< macOS 用>
    `https://sukkiri.jp/technologies/ides/anaconda-mac_install.html`
- 次に，ニューラルネット作成用に追加パッケージとして `keras`, `tensorflow` をインストールする

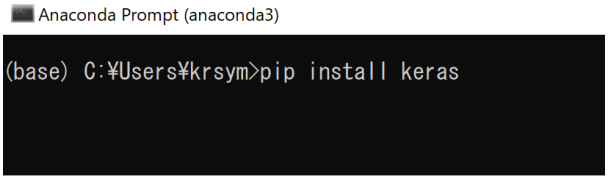
  - win → 2page へ
  - mac → 4page へ

Windows:

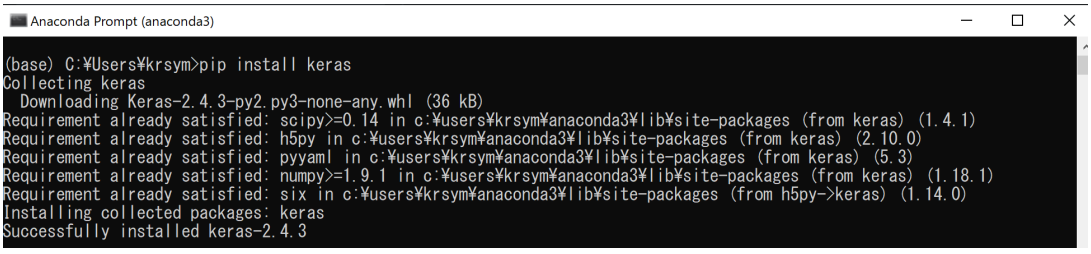
1. Anaconda prompt を以下のように起動



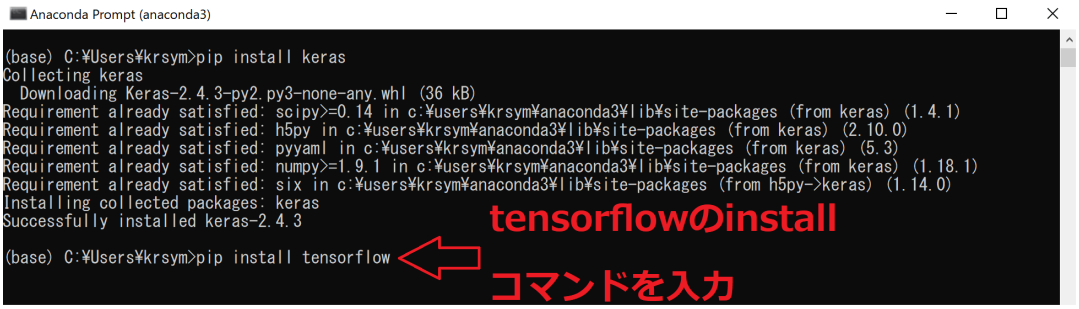
2. コンソールが表示されるので，“pip install keras” と入力してリターン



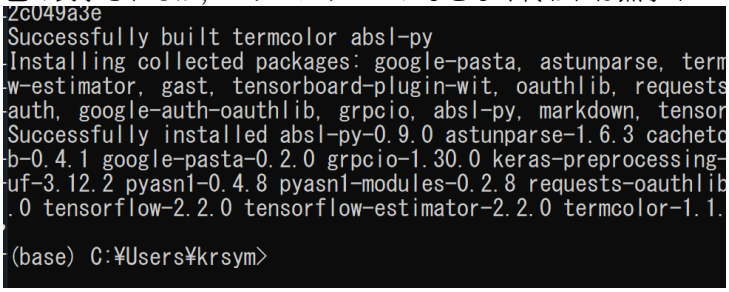
成功すると以下のような表示になる



3. さらに, 引き続き “pip install tensorflow” と入力してリターン



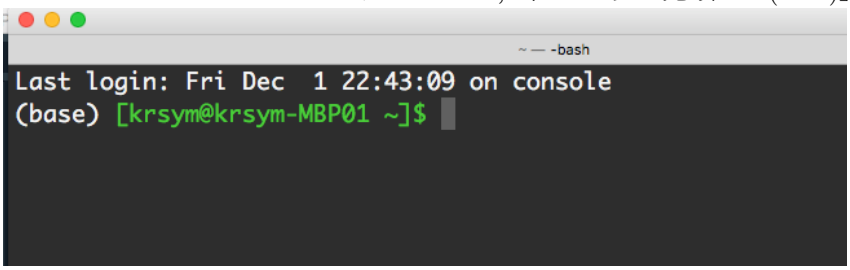
色々表示されるが，エラーメッセージなどなく終われば無事インストールされているはず



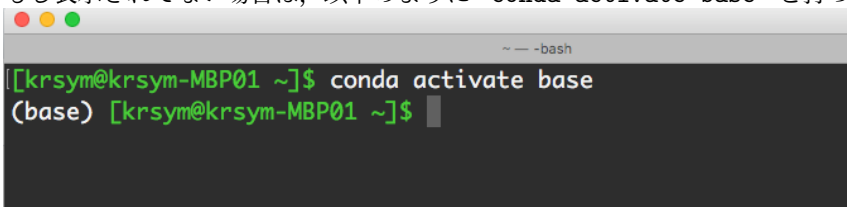
4. 他にも `matplotlib`, `seaborn` というパッケージを使用するが anaconda には通常最初からインストールされている．存在しないパッケージの機能を使おうとすると `ModuleNotFoundError: No module named ‘xx’` のようなエラーメッセージが出るので（xx はパッケージ名），上と同様に Anaconda prompt から適宜 `pip install xx` とすればよい．

Mac:

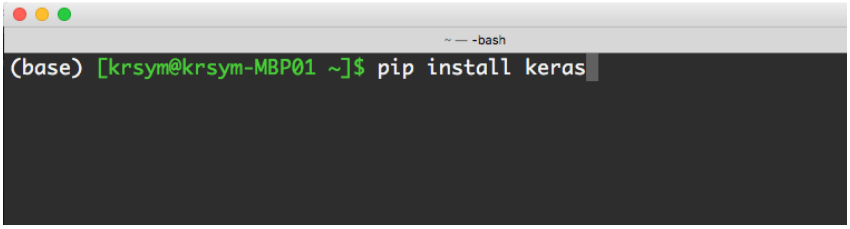
1. Terminal.app を起動 (Mac でターミナルを起動する方法: `https://syncer.jp/mac-terminal`)
2. Anaconda がインストールされていると，下のように先頭に「(base)」と表示されるはず



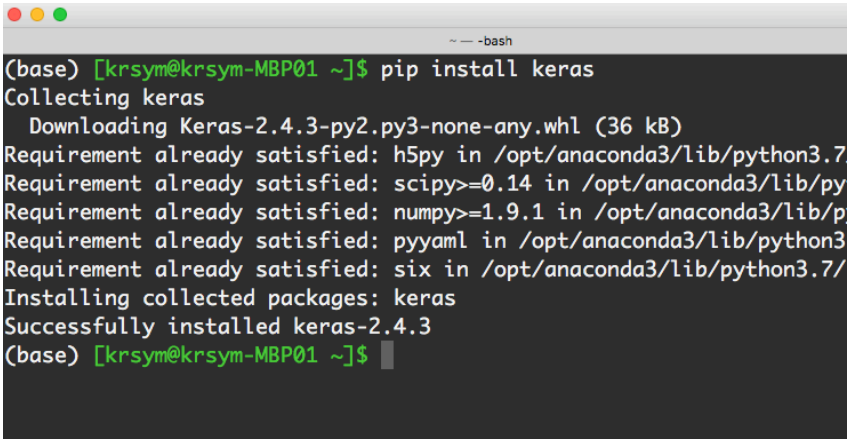
   - もし表示されてない場合は，以下のように “`conda activate base`” と打ってリターンする



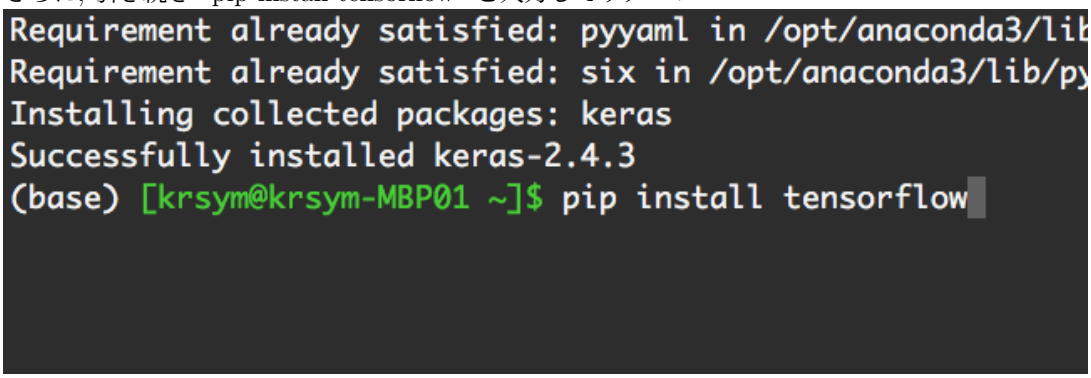
3. “`pip install keras`” と打ってリターンすると keras がインストールされる



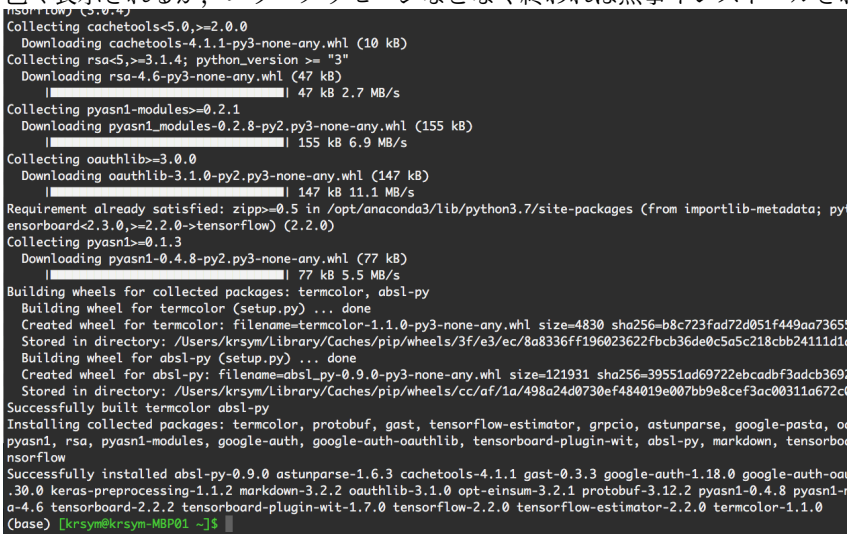
   うまくいくと以下のような表示になる



4. さらに, 引き続き “pip install tensorflow” と入力してリターン



色々表示されるが，エラーメッセージなどなく終われば無事インストールされているはず

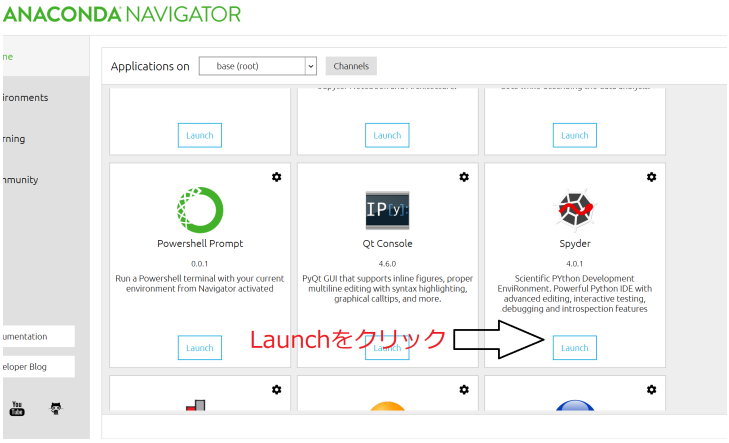


5. 他にも `matplotlib`, `seaborn` というパッケージを使用するが anaconda には通常最初からインストールされている．存在しないパッケージの機能を使おうとすると `ModuleNotFoundError: No module named 'xx'` のようなエラーメッセージが出るので（xx はパッケージ名），上と同様にターミナルから `pip install xx` とすればよい．

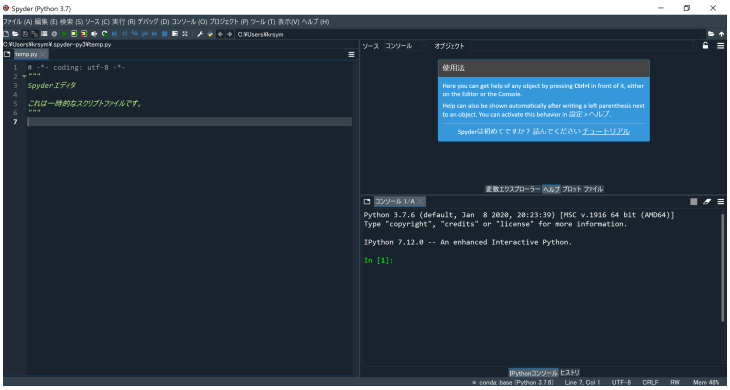
1.1.2 Spyder による python プログラムの実行

- これで準備が整ったので，spyder を使って python programming が可能か確かめる（Video も参照）．以降は，win と mac で違いはほぼない．

1. Anaconda navigator から spyder を起動する

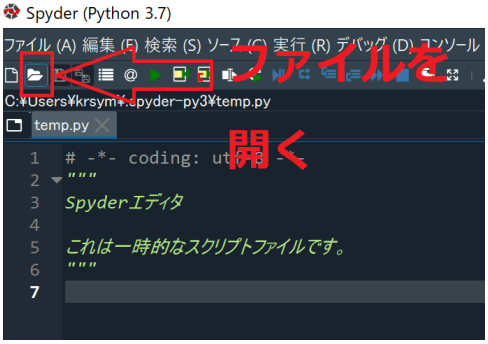


2. 起動すると以下のような画面になる

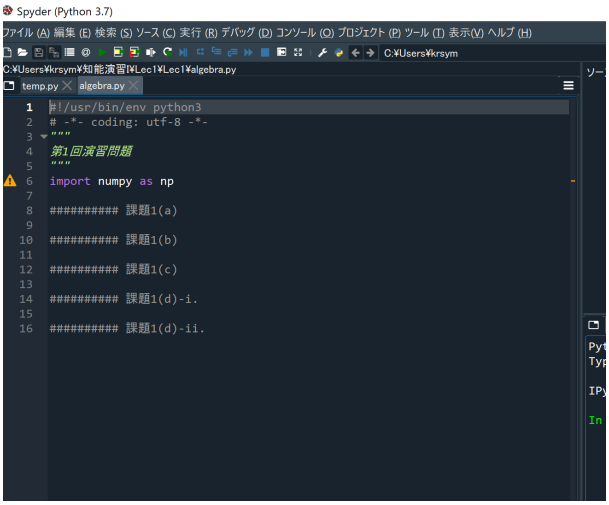


3. 右下のコンソール領域の “In [1]:” の後にコマンドを打ち込んで実行することができる．試しに，“`print("Hello, world")`” とすると以下の結果が得られる

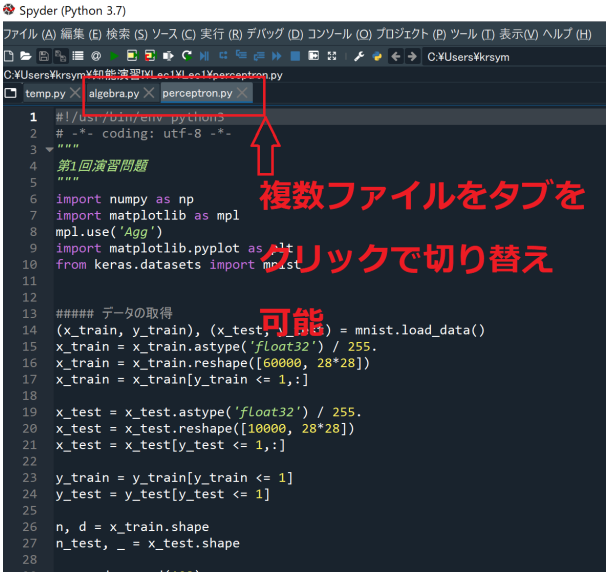
4. 次に，演習課題で使うテンプレート pyhon ファイルが開くか確かめる．左上のアイコンからファイルを開くことができるので，ダウンロードした zip に含まれている “algebra.py” を選ぶ



すると以下のように，画面左のエディタ領域に algebra.py が開かれる

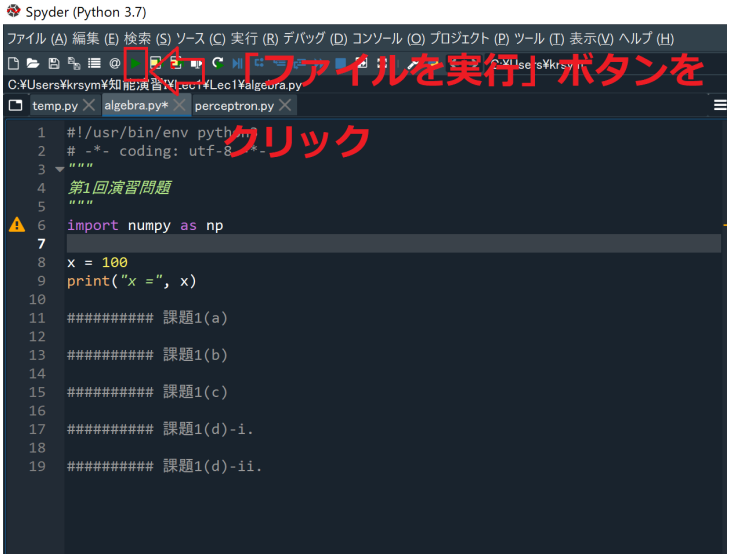


これで，algebra.py が編集できる．さらに，同じように左上の「ファイルを開く」アイコンからもう一つの課題テンプレート “perceptron.py” も開いてみると以下のようになり，タブでファイルを切り替えることができるようになる．

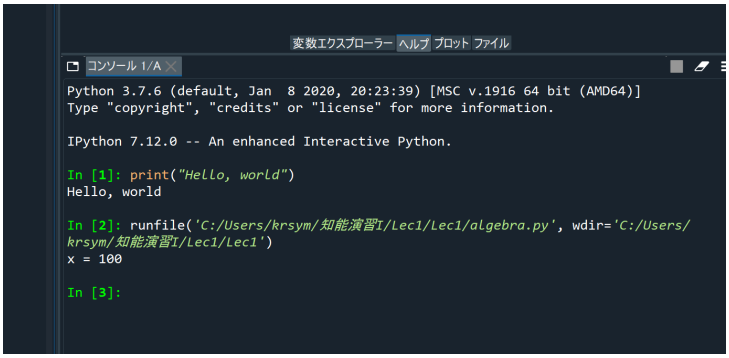


5. 開いたファイルを編集して実行する．下の例では，“algebra.py” のタブをクリックして，“x = 100” と “print("x =", x)” の２行を記載している．

次に上部にある緑色の再生マーク「ファイルを実行」ボタンをクリックすると自動的にファイルが保存されファイル内のプログラムが実行される



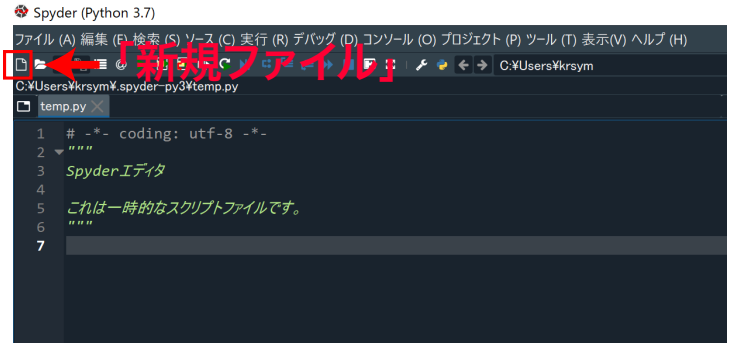
結果は右下のコンソール領域に表示される



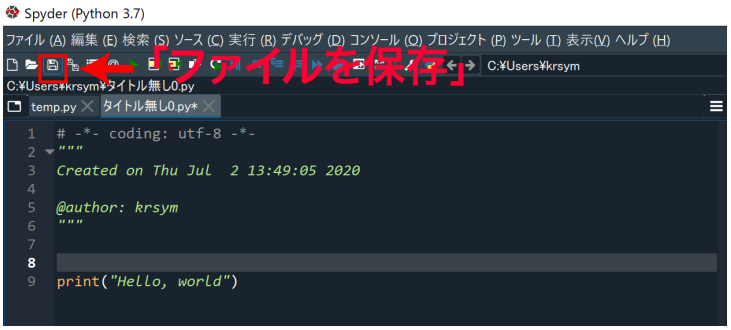
6. 先ほど開いたもう一つのファイル “perceptron.py” のタブをクリックして，こちらも実行できるか確かめる．すでに色々なプログラムが記載されているが中身は一旦気にせずに，「ファイルを実行」ボタンをクリックする（このコードは内部で学習用データをダウンロードするプロセスがあるので少し時間がかかることがある）．perceptron.py は内部的に keras と tensorflow がインポートされているので，これらが**うまくインストールされてないと関連するエラーがでる**ので，その場合はインストールされているか確かめて必要に応じて再度インストール作業を行う．インストールされているパッケージのリストは “`pip list`” で確認できる

```
In [3]: pip list
Package                           Version
--------------------------------- -------------------
alabaster                         0.7.12
anaconda-client                   1.7.2
anaconda-navigator                1.9.12
anaconda-project                  0.8.3
argh                              0.26.2
asn1crypto                        1.3.0
astroid                           2.3.3
astropy                           4.0
atomicwrites                      1.3.0
attrs                             19.3.0
autopep8                          1.4.4
```

7. ここまではこちらで予め用意したファイルを開いたが，新規のファイル作成も当然可能である．その場合は右上の「新規ファイル」ボタンをクリックする．



新規の「.py」ファイルが開かれるので適当に編集し，同様に緑色の再生マーク「ファイルを実行」のボタンをクリックすると実行できる．ただし，新規に作成した場合は，実行するだけでは自動で保存されないので，最初に保存する場合は「ファイルを保存」(Control + S) から保存する（以降は実行時に自動で保存される）．



1.2 CSE 用

- ホームディレクトリに演習用のディレクトリを作成し, DLL に移動
  step1: `mkdir -p DLL`
  step2: `cd ./DLL`
- 今日の課題を DLL にダウンロードし, 展開する
- 展開したフォルダの中に, 以下のものがすべて入っていることを確認
  - algebra.py
  - perceptron.py
  - task.pdf
- インストール済みのパッケージを確認し, `matplotlib`, `seaborn`, `keras`, `tensorflow` を (インストールされていなければ) インストール
  step1: `pip list`
  step2: `matplotlib`, `seaborn`, `keras`, `tensorflow` が確認できなければインストール
    - `pip install matplotlib`
    - `pip install seaborn`
    - `pip install keras`
    - `pip install tensorflow`
- python の起動: `python` または `ipython` (詳細は講義ノート参照)

2 課題

1. 以下のプログラムを作成せよ. ただし, algebra.py にコードを保存すること.
   (a) 講義ノート「1.1 準備」を読んで “Hello, World” を表示するコードを algebra.py 内に記載し, ファイルの内容のコードが実行できることを確かめよ (いくつか実行の方法があるが, いずれかでできることを確認すればよい).
   (b) 講義ノート 「1.2 numpy」「1.2.1 np.array による配列の生成 」「1.2.2 特殊な行列の生成 」「1.2.3 配列の操作」を読み, 書かれているコードを一通り自分の手元で実際に動かして確認してから, 以下のコードを algebra.py 内に記載せよ.
      i. 以下の行列 A とベクトル $\boldsymbol{b}$ を作成して表示せよ. ただし, 出力結果は float 型とすること.

$$A = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 6 & 7 & 8 & 9 & 10 \\ 11 & 12 & 13 & 14 & 15 \\ 16 & 17 & 18 & 19 & 20 \end{bmatrix}, \quad \boldsymbol{b} = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 1 \end{bmatrix}$$

      A の作成には, 「1.2.3 配列の操作」の最後から 2 パラグラフ目のサイズの変更に関する記述を参考にできる（これ以外の方法で作っても構わない）.
      ii. (a) で定義した行列 A とベクトル $\boldsymbol{b}$ の積 $A\boldsymbol{b}$ を計算して表示せよ.
      ヒント: 講義ノート「1.2.3 配列の操作」の補足 1 の後あたりの記述を参照.
      iii. (a) で定義した行列の列和と行和を計算して表示せよ.
      ヒント: 列和は 5 次元配列, 行和は 4 次元配列が出力される. 講義ノート「1.2.3 配列の操作」の補足 1 の後あたりの記述を参照.
   (c) 講義ノート「1.3 for 文 if 文」を読み, 書かれているコードを一通り自分の手元で実際に動かして確認してから, 以下を実施せよ.
   algebra.py には数列

$$\begin{aligned} a_0 &= 0, \\ a_{n+1} &= 2a_n + 1 \ (n \geq 0), \end{aligned}$$

   を第 1 項 ($n = 1$) から第 10 項 ($n = 10$) まで表示するコードが記載されている. これを参考に, for 文や if 文を用いて, 以下の数列の第 1 項 ($n = 1$) から第 10 項 ($n = 10$) を順に出力するコードを作成せよ[*1].

$$\begin{aligned} a_0 &= 6, \\ a_{n+1} &= \begin{cases} a_n/2 & \text{if } a_n \equiv 0 (\mathrm{mod} 2) \\ 3a_n + 1 & \text{if } a_n \equiv 1 (\mathrm{mod} 2) \end{cases} \end{aligned}$$

   ただし, $a_n \equiv 0 (\mathrm{mod} 2)$ や $a_n \equiv 1 (\mathrm{mod} 2)$ は a_n を 2 で割った剰余が 0, ないしは 1 であることを意味する. ヒント: python で剰余は % で計算できる（例えば, コンソールで 10 % 3 などと入力して

[*1] ii の数列は, コラッツの問題 (500 ドルの賞金のかかった未解決問題) として知られており, “どんな自然数を初期値としても, あるステップ n' で必ず $a_{n'} = 1$ となる” と予想されている.

確かめてみるとよい）．ヒント: python での同値関係の判定には ‘==’ を使う (例えば，コンソールで `1 == 1` や `1 == 0` と入力して確かめてみるとよい).

2. 講義ノート 「1.4 関数の定義」を読み, 書かれているコードを一通り自分の手元で実際に動かして確認してから, 以下の問いに従って, perceptron.py にパーセプトロンを学習するコードを作成せよ. perceptron.py は手書き数字のデータセットをダウンロードして, 数値の $0(y = 0)$ と $1(y = 1)$ を分類するパーセプトロンを作成するためのテンプレートである．入力は 28×28 のピクセルの画素値を一列に並べた $784(= 28 \times 28)$ 次元のベクトルである (元の画像を保存するコードがコメントアウトしてあるので興味がある場合はコメントを外して確認してみるとよい)．データをパラメータの訓練用 (12665 枚) と分類精度のテスト用 (2115 枚) に分けて, 学習の過程での訓練データとテストデータの誤差関数の推移を error.pdf に保存している．ただし, デバッグ中の実行時間を短くするために，初期状態では訓練データの一部のみ使用するようにしてある．最後の課題では, 全てのデータを使うこと (perceptron.py の先頭付近 `use_small_data` を False にすると全データを使う).

(a) シグモイド関数 f を定義して，パーセプトロンの出力 $f(\boldsymbol{w}^\top \boldsymbol{x})$ を計算するコードを作成せよ．まず, 実数 x に対して, 以下で定義されるシグモイド関数 `sigmoid(x)` を perceptron.py 内に作成する (中身のない関数がテンプレートに記載済み).

$$f(x) = \frac{1}{1 + e^{-x}}$$

関数の引数はスカラーを想定してよい[*2]．また，e の指数関数には `np.exp` が利用できる (`np.exp(1)` や `np.exp(2)` などとして確認してみるとよい)．関数が正しく定義されている状態でコンソールから `sigmoid` を実行すると例えば以下のような結果が得られる (コンソール上で直接定義するか，perceptron.py をコンソールから実行する (spyder なら実行ボタンを押すだけ) とその後，`sigmoid` 関数にアクセスできる).

```
In [2]: sigmoid(1)
Out[2]: 0.7310585786300049

In [3]: sigmoid(-1)
Out[3]: 0.2689414213699951
```

perceptron.py には訓練とテストで 2 箇所，出力を計算する箇所がある（コード内のコメント参照）．作成したシグモイド関数 `sigmoid(x)` の引数に $\boldsymbol{w}^\top \boldsymbol{x}$ を渡して, それぞれ $f(\boldsymbol{w}^\top \boldsymbol{x})$ を計算すること．パラメータ $\boldsymbol{w}$ は変数 ‘w’ としてすでに定義されている．また，$\boldsymbol{x}$ は定数 1 を先頭に含んだ形で変数 ‘xi’ として定義されているので，これらをそのまま使えばよい.

(b) 2 値分類問題の誤差関数 (`error_function`) を定義せよ．今回の場合, 誤算関数はシグモイド関数の出力を実数値 f, 教師ラベルを y としたとき，以下のように表現される

$$E(f, y) = -y \log f - (1 - y) \log(1 - f)$$

対数関数 log には `np.log` が利用できる．関数が正しく定義されている状態でコンソールから `error_function` を実行すると例えば以下のような結果が得られる.

[*2] 普通に作成すれば実際にはベクトル入力でも動作するようになる．余裕のあるひとは作成後, 関数 `sigmoid(x)` が定義されている状態でコンソールで `sigmoid(np.ones(10))` などとして何が起きているか考えてみるとよい.

```
In [7]: error_function(0.1,1)
Out[7]: 2.3025850929940455

In [8]: error_function(0.9,1)
Out[8]: 0.10536051565782628
```

関数が完成したら，perceptron.py 内で `error_function` の値を誤差記録用の配列に保存している部分のコメントを外して有効化しすること (訓練用 `e_train` とテスト用 `e_test` の 2 箇所あることに注意)．ここまで作成できると実行後に，error.pdf に初期パラメータでの誤差が表示されるはずである（まだパラメータ更新部分を作成してないので, 訓練とテストそれぞれ変化せず定数がプロットされる)．

(c) スライド 17page, 「確率的勾配降下法」の更新式にしたがって, パラメータの更新を行いパーセプトロンの学習アルゴリズムを完成させよ．ただし, 学習率 $\eta_t = 0.01/$エポック数 とする．

- 挙動がおかしい際には, epoch 数を小さくして動作確認するとよい場合もある（例えば，`num_epoch=2` とする．perceptron.py では初期誤差測定のために, 最初の epoch では勾配を計算してないことに注意)．あるいは, spyder や ipython で interactive に実行した場合は，実行後ファイル内で作成した変数にコンソールからアクセスできるので（ただし，関数内の変数はできない)，おかしな値になってないか確認するとよい (例えば, 'w' を確認してみる)．

(d) perceptron.py の先頭付近の `use_small_data` を False にし，全データを使うようにしたうえで, 現在の作業ディレクトリに保存された図 (error.pdf) から, 訓練誤差とテスト誤差がエポック数とともに減少する様子を確認せよ．

3 課題の提出

Moodle を使ってファイルを提出してください. 提出方法は以下の通りです.

- Moodle にログインし, 知能プログラミング演習のページへ移動.
- アルゴリズム第 1 回 Python 入門の項目に, algebra.py, perceptron.py および error.pdf をアップロードする.

6/14(金) の 17:00 を提出期限とします.