萩原研　プログラミング研修

1日目：Pythonによるプログラミング練習
機械学習や深層学習系のコードの大半はPythonで実装されており、萩原研でメインに使われている言語もPythonです。今日はPythonの基本的な書き方を一通り練習します。

Pythonの基礎の基礎

まず、Pythonのコードを書く上で最低限知っているべき事柄を説明します。
最低限のことしか載せてないので詳細は各自で調べながら進めてください。
今回の演習では3系のPythonを前提としています。

Pythonコードの記述と実行
適当なテキストエディタでコードを書いて、拡張子を".py"にして保存します。
実行時は、ターミナルで次のコマンドを入力します（"aaa.py"はファイル名です）。

```
$ python aaa.py
```

対話モードによる**Python**コードの実行
ターミナルで `$ python` とのみ入力するとpythonの対話モードが起動します。
このモードでは次のように一行ずつコードが実行されるので動作確認などに便利です。

```
$ python
>>> 1 + 2
3
```

Pythonのコメント
`#` から行末までの文字列はコメントとして認識されます。

```
n = 10
print(n ** 2)   # nの2乗を表示する
```

制御構文と関数の書き方
CやJavaではif文や関数の中身を中かっこ{}でくくっていましたが、pythonでは代わりにコロン：とインデント（通常は半角スペース4個）を使って次のように書きます。

```
age = 18
if age < 20:
    print("お酒は２０歳になってから")


# 関数の定義(例：a, b二つの引数を受け取ってその和を返す)
def add(a, b):
    return a + b

# 関数の実行
add(2, 4)   # 6
```

for文の使い方

for文によるループの書き方はCやJavaのそれとやや異なっています。

pythonでは `for <変数> in <リスト>` のように記述します。変数は各繰り返しにおいてリスト中の各要素を前からたどるように順番に変化していきます。

C言語による記述（例：1から10までの数を順番に表示する）

```
int i;
for (i = 0; i < 11; i++) {
    printf("%d\n", i);
}
```

Pythonによる記述

```
# range(1, 11) は1から10までの数字のリスト（※厳密にはイテレータ）を返す
for i in range(1, 11):
    print(i)
```

Pythonでは次のコードのようにループ変数が数値でなくても良いためCよりも柔軟な実装が可能です。

```
# 果物リストの中の文字列を順番に表示する
for fruit in ["apple", "orange", "banana"]:
    print(fruit)
```

演習1. Hello World!

print文を使って "Welcome to Hagiwara Lab." と表示してください。

＜出力例＞

```
Welcome to Hagiwara Lab.
```

演習2. FizzBuzz

数字を1から30まで画面に表示してください。ただし、数字が3で割り切れるときには"Fizz"、5で割り切れるときには"Buzz"、3と5の両方で割り切れる時には"FizzBuzz"と代わりに表示してください。

＜出力例＞

```
1
2
Fizz
4
Buzz
...
29
FizzBuzz
```

演習3. 文字列の逆順

文字列"stressed"の文字を逆順にして表示してください。
＜ヒント＞pythonの文字列やリストには スライス と呼ばれる便利な記法があるのでそれを活用してみましょう。

＜出力例＞

```
desserts
```

演習4. 「パタトクカシーー」

「パタトクカシーー」という文字列の偶数番目の文字、奇数番目の文字をそれぞれ取り出した文字列を得て、それを表示してください。

＜出力例＞

パトカー
タクシー

演習5. リストの作成

1から10の数字をそれぞれ二乗した数字を並べたリストを作成してください。このとき、appendメソッドを用いてリストに順次要素を追加していくという形で実装してください。

＜出力例＞

```
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

演習6. リスト内包表記

1から10の数字をそれぞれ二乗した数字を並べたリストを作成してください。このとき、リスト内包表記 を用いてリストを直接生成する形で実装してください。

＜出力例＞

```
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

演習7. 円周率

"Now I need a drink, alcoholic of course, after the heavy lectures involving quantum mechanics."
という文を不要なカンマ、ピリオドを除去したうえで単語に分解し、各単語の文字列を並べたリストを作成してください。

＜出力例＞

```
[3, 1, 4, 1, 5, 9, 2, 6, 5, 3, 5, 8, 9, 7, 9]
```

演習8. 元素記号

"Hi He Lead Because Boron Could Not Oxidize Flourine. New Nations Might Also Sign Peace Security Clause. Arthur King Can."
という文を単語に分解し、1, 5, 6, 7, 8, 9, 15, 16, 19番目の単語は先頭の1文字、それ以外の単語は先頭の2文字を取り出し、取り出した文字列をキーとして単語の位置（先頭から何番目か）を値とするディクショナリ を作成してください。

＜出力例＞
※中身の順番は違っていてもOKです。あとLiがLeだったりMgがMiだったり一部おかしいですが気にしないでいいです

```
{'K': 19, 'Le': 3, 'Al': 13, 'F': 9, 'Na': 11, 'Cl': 17, 'S': 16, 'B': 5, 'Ne': 10, 'Ca'
```

演習9. タイポグリセミア

スペースで区切られた単語列をコマンドライン引数の形で与え、各単語の先頭と末尾の文字はそのまま残し、それ以外の間の文字の順序をランダムに並び替えて出力するプログラムを作成してください。ただし、３文字以下の単語列は並び替えを行わないものとします。

＜出力例＞

```
$ python ex09.py こんにちは みなさん おげんき ですか ？ わたしは げんき です 。
こんちには みさなん おんげき ですか ？ わしたは げんき です 。
```

演習10. n-gram

シーケンス（文字列やリストなど）と整数nを引数とし、与えたシーケンスのn-gram（連続するn個の要素のリストをすべて並べたリスト）を出力する関数を作成してください。また、この関数を用いてコマンドライン引数で与えた文字列の単語bi-gram(n = 2の場合)、文字bi-gramを出力してください。

＜出力例＞

```
$ python ex10.py I am an NLPer
単語bi-gram: [['I', 'am'], ['am', 'an'], ['an', 'NLPer']]
文字bi-gram: ['Ia', 'am', 'ma', 'an', 'nN', 'NL', 'LP', 'Pe', 'er']
```

演習11. ファイルの読み込み

演習11-17では複数の文が書かれているファイルdata.txtを読み込み、各文の類似度をtf-idf値のコサイン類似度により定量的に求めるという簡単な自然言語処理の一連の過程を実装していきます。
＜前準備＞ 現在のディレクトリに新たにdatasetディレクトリを作り、そこにdata.txtをダウンロードしてきてください。
data.txtファイルを一行ずつ読み込み、読み込んだ行を表示してください。

＜出力例＞

```
リンゴとリンゴとリンゴ
リンゴとレモンとレモンとミカン
リンゴとイチゴとミカン
レモンとイチゴとミカン
ミカンとミカンとブドウとブドウ
```

演習12. 読み込んだデータの処理

data.txtファイルを一行ずつ読み込み、各行からそれぞれ果物の名詞のみを抽出したリストdocsと、ファイル全体に現れる果物の種類を重複なしでまとめたリストtermsをそれぞれ作成してください。

＜出力例＞
※termsの中身の順番は違っていてもOKです。

```
docs : [['リンゴ', 'リンゴ', 'リンゴ'], ['リンゴ', 'レモン', 'レモン', 'ミカン'], ['リンゴ', 'イ
terms: ['リンゴ', 'ブドウ', 'ミカン', 'レモン', 'イチゴ']
```

演習13. tf値の計算

各文書において各単語がどれだけ重要なのかを表す指標として **tf-idf値**
単語termと文書docからtf値(Term Frequency)を計算する関数 `tf(term, doc)` を定義してください。
このとき、テスト用の文書リストdocsと単語リストtermsとして次のデータを使用してください。

```
docs  = [["リンゴ", "リンゴ"], ["リンゴ", "レモン"], ["レモン", "ミカン"]]
terms = ["リンゴ", "レモン", "ミカン"]
```

tf-idf値についての詳しい式や説明はこちらのサイトを参考にしてください。

＜出力例＞

```
tf(リンゴ, ['リンゴ', 'リンゴ']) = 1.0  tf(レモン, ['リンゴ', 'リンゴ']) = 0.0  tf(ミカン, ['リ
tf(リンゴ, ['リンゴ', 'レモン']) = 0.5  tf(レモン, ['リンゴ', 'レモン']) = 0.5  tf(ミカン, ['リ
tf(リンゴ, ['レモン', 'ミカン']) = 0.0  tf(レモン, ['レモン', 'ミカン']) = 0.5  tf(ミカン, ['レ
```

演習14. idf値の計算

単語termと文書リストdocsからidf値を計算する関数 `idf(term, docs)` を定義してください。テスト用のデータは演習13と同じものを使用してください。
＜ヒント＞logを計算するときは数値計算ライブラリnumpyを利用すると便利です。
＜出力例＞

```
idf(リンゴ) = 1.1760912590556813
idf(レモン) = 1.1760912590556813
idf(ミカン) = 1.4771212547196624
```

演習15. tf-idf値の計算

演習13, 14で定義したtf, idf値を計算する関数を用いて、行方向に各文書の、列方向に各単語のtf-idf値をまとめた行列を計算する関数を定義してください。
＜ヒント＞特定サイズの行列を定義するときはnp.zerosなどを使用すると便利です。

＜出力例＞

```
[[ 1.17609126  0.          0.        ]
 [ 0.58804563  0.58804563  0.        ]
 [ 0.          0.58804563  0.73856063]]
```

演習16. コサイン類似度の計算

二つのベクトル$\mathbf{x}_1$, $\mathbf{x}_2$のコサイン類似度

$$cosine(\mathbf{x}_1, \mathbf{x}_2) = \frac{\mathbf{x}_1 \cdot \mathbf{x}_2}{|\mathbf{x}_1||\mathbf{x}_2|}$$

を計算する関数 `cosine_sim(x1, x2)` を定義してください。
テスト用のデータとして次のベクトルを使用してください。

```
x1 = np.array([1, 0, 0, 1])
x2 = np.array([0, 1, 0, 1])
```

＜出力例＞

```
0.5
```

演習17. パッケージ化

演習15, 16で作成したソースコードをnlpパッケージとして管理したいと思います。
そのため、nlpディレクトリを新たに作成し、この2つのソースファイルを移します。その後、nlpディレクトリに **"__init__.py"** という空のテキストファイルを作成します。これがパッケージ化の処理になります。
最後に、演習12で処理したdocsリストとtermsリストに対して、今作成したnlpパッケージのモジュールを用いて各文書間のコサイン類似度をそれぞれ計算し、[i, j]要素が文書iと文書jのコサイン類似度を表しているような行列としてまとめて表示してください。

＜出力例＞

```
[[ 1.          0.37683623  0.56659672  0.          0.        ]
 [ 0.37683623  1.          0.38559553  0.69729605  0.18349782]
 [ 0.56659672  0.38559553  1.          0.64760589  0.27590039]
 [ 0.          0.69729605  0.64760589  1.          0.26315626]
 [ 0.          0.18349782  0.27590039  0.26315626  1.        ]]
```