たぶんけっこう楽しい

Python入門

本講座の目的

本講座の目的は、全くプログラミングをしたことがないという方に、

少しわかった気分になってもらい、

Pythonやプログラミング全般に興味を持ってもらうことです。

これを聞いて興味を持った方は是非Pythonで遊んでみてください！

Python 2.7ja1 documentation(http://docs.python.jp/2.7/)

Pythonスタートブック(書籍)

注意

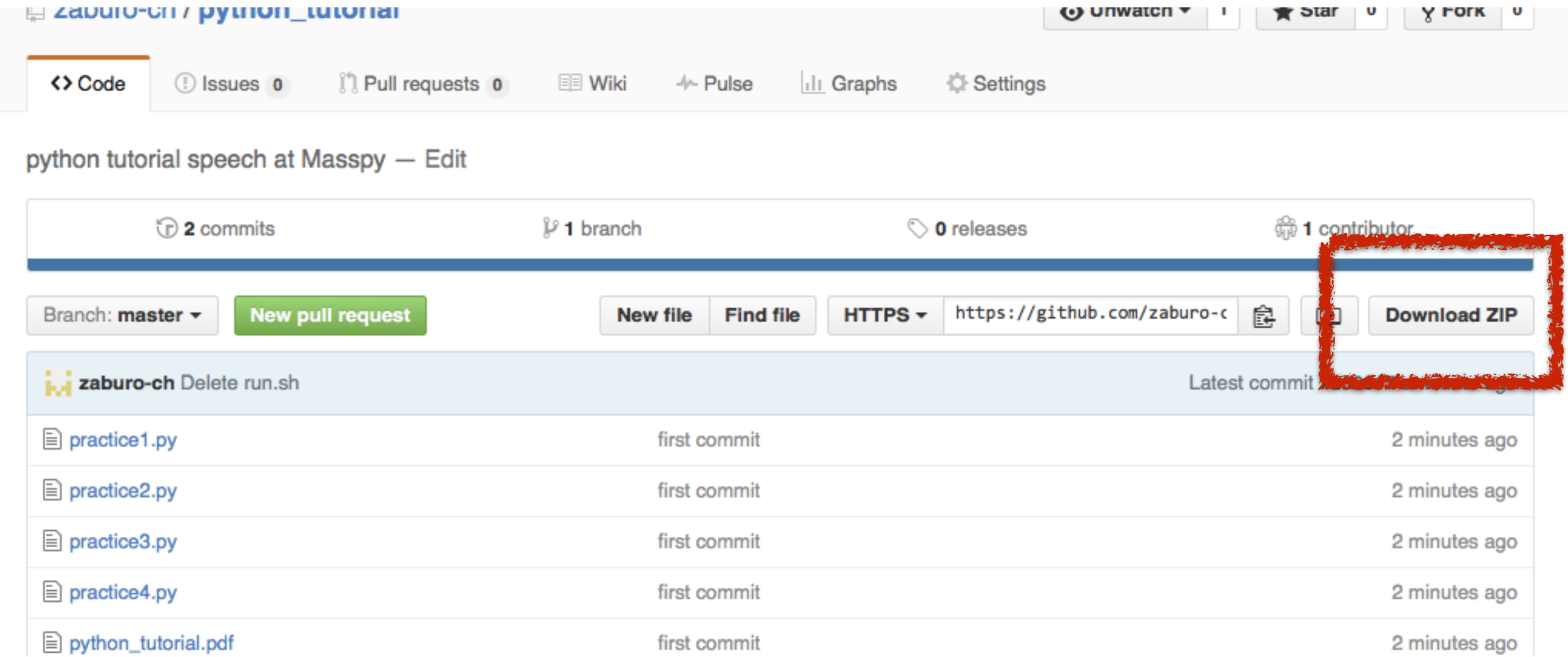
スライドだけでは足りない情報がいっぱいあります。

適宜、画面や口頭で補足します。

また、簡単のためにどのチュートリアルにもあるような重要な事項をすっ飛ばしていたりするのでご理解ください。

ファイルのダウンロード

https://github.com/zaburo-ch/python_tutorial



何はともあれ実行してみよう①

コマンドプロンプト(Macではターミナル)を開き、

先ほとダウンロードしたフォルダに移動します。

使用するコマンド(括弧内はMac)	動作
cd foldername	foldernameというフォルダに移動
cd (pwd)	現在どこにいるか確認
dir (ls)	ファイル一覧を取得

何はともあれ実行してみよう②

dir (ls) をして weblio.py があることを確認し

下記のコマンドを入力して実行！

```
python weblio.py
```

一般にPythonのプログラムは次のように実行します。

```
python .pyで終わるファイル名
```

何はともあれ実行してみよう③

```
% python weblio.py
Search word :
```

こんな感じになります。

適当な英単語を入力してEnterを押してみてください

一応中身を見てみる

```
import requests
from bs4 import BeautifulSoup


print "Search word :",
word = raw_input()
r = requests.get("http://ejje.weblio.jp/content/"+word)
soup = BeautifulSoup(r.text, "lxml")
dicts = soup.find_all("div", class_="hideDictPrs")

if len(dicts) == 0:
    print "There's no such word."
else:
    top_dict = dicts[0]
    statements = top_dict.find_all("div", class_="level0")

    for s in statements:
        print s.get_text()
```

こんな感じのものが
サクッと作れたら楽しいよね

目次

基礎編

目次

応用編

- スクレイピングしてみよう

画面に文字を出力しよう

プログラミングを始める前に

プログラミングは
コンピュータにやって欲しいことを順に書いていく
ものだと考えてください。

画面に文字を出力しよう①

この場合は、

画面に出力(print)して！と書けばいいので、

例えば「Hello World!」と出力してほしいなら、

```
print "Hello World!"
```

と書きます。このとき、ルール(文法)があって

- まずprintと書く
- その後に1つスペースをあけて出力したいものを書く
- 文字列は””で囲う

画面に文字を出力しよう②

練習1

practice1.py(何も書いてないです)を編集して

好きな文を出力させてみましょう

ヒント

```
print "Hello World!"
```

計算してみよう

計算してみよう①

突然ですが、

$x = 30$

$y = 20$

$z = 3x + 2y$

のとき、zはいくつでしょう？

計算してみよう②

Pythonを使って確かめてみましょう。

```
x = 30
y = 20
z = 3*x + 2*y
print z
```

	加算	減算	乗算	除算
数学	+	-	×	÷
Python	+	-	*	/

計算してみよう③

ここで、x, y, zは変数と呼ばれ、

値を格納する入れ物のような働きをします。

先ほどは、30や20などの整数を入れましたが、

文字列なども変数に格納することができます。

```
a = "mojiretsu"
print a
print "a"
```

```
% python sample2-2.py
mojiretsu
a
```

計算してみよう④

練習2

practice2.pyに変数 a と変数 b が書いてあります。

新たな変数 c を a と b の和として作って、

cを出力しましょう。

補足

Pythonでは = (イコール)は等号ではなく

代入を示します。そのため、次のような書き方もあり

また、同じ数字でも整数と整数でない数(実数)を

区別していて微妙に計算結果が異なります。

3 / 2 は 1 になります（整数同士は切り捨て）。

3.0 / 2.0 は 1.5 になります。

3.0 / 2 は 1.5 になります（実数になる）。

条件分岐させてみよう

条件分岐させてみよう①

先ほど学んだ内容で計算ができる様になりましたが、
このままでは電卓くらいにしか使い道はありません。

計算結果をそのまま出すのではなく、
ちょっと賢くして、
結果を判断して教えてくれる様なプログラムに
改良しましょう

条件分岐させてみよう②

今、z には何が入っているかわからないものとします。

この z が負の時 fu と出力して、
それ以外の時 hifu と出力するには
どうすればいいでしょうか？

この場合は、if文と呼ばれるを条件分岐を使うことで、
z がどの様な条件を満たすかによって
処理を変えることができます。

条件分岐させてみよう③

if文はelseを伴って次の様に書くことができます。

```
if z < 0:
    print "fu"
else:
    print "hifu"
```

もし(if) z < 0 ならば print “fu” を実行して、

そうでなければ(else) print “hifu” を実行して！

という意味になっています。

条件分岐させてみよう④

z を 指定して実行してみるとこんな感じ。

```
z = -10
if z < 0:
    print "fu"
else:
    print "hifu"
```

```
% python sample3-1.py
fu
```

```
z = 10
if z < 0:
    print "fu"
else:
    print "hifu"
```

```
% python sample3-2.py
hifu
```

条件分岐させてみよう⑤

練習3

変数 mid と final には、けんたろうくんの

中間試験と期末試験の成績が入っています。

両方とも100点満点です。最終的な成績は中間40%、

期末60%で、単位は60点以上で浮きます。

浮いてたらpass、ダメならfailを出力してください！

ヒント

```
result = 0.4*mid + 0.6*final
```

補足

実際には2通り(ifとelse)より多く分岐することもあります。

その時はif文の中にまたif文を書くか、

elif (else ifがくっついたやつ)を用います。

```
if z == 10:
    print "ten"
else:
    if z == 20:
        print "twenty"
    else:
        if z == 30:
            print "thirty"
        else:
            print "other"
```

```
if z == 10:
    print "ten"
elif z == 20:
    print "twenty"
elif z == 30:
    print "thirty"
else:
    print "other"
```

一般に elif で書ける時は、
elif で書く方が好まれます。

リストとforに触れてみよう

リストとforに触れてみよう①

リストは他の言語で配列と呼ばれるものに近いです。

例えば、5人の学生のテストの点数を記録したい時、

```
score0 = 80
score1 = 70
score2 = 60
score3 = 100
score4 = 40
```

こんな感じで扱うこともできますが、

何回もscoreって書くの面倒ですよね。

ましてや学生が100人だったら……

リストとforに触れてみよう②

これをリストを用いて簡単に書くことができます。

```
score = [80, 70, 60, 100, 40]
```

そして、0番から始まる番号で指定して

中身を見たり、変更することができます。

```
score = [80, 70, 60, 100, 40]
print score[0]
print score[1]
score[0] = 8
print score[0]
```

```
% python sample4-1.py
80
70
8
```

リストとforに触れてみよう③

また、リストを使うことによって

リスト内の要素の和などが簡単に求められます。

```
score = [80, 70, 60, 100, 40]
```

求めるもの	求め方	結果
要素の和	sum(score)	350
要素数	len(score)	5
最大値	max(score)	100
最小値	min(score)	40

sumなどの()がつくものは関数と呼ばれますが、

今は便利な呪文くらいの認識で構いません。

リストとforに触れてみよう④

ループ(繰り返し)との相性が良いことも
リストを使うのメリットの一つです。

特にfor文と呼ばれる繰り返し処理とともに使うことで
面倒な処理を非常に簡潔に書けるようになります。

リストとforに触れてみよう⑤

for文は次のように書きます。

```
score = [80, 70, 60, 100, 40]
for a in score:
    print a
```

for a in scoreを英語っぽく訳すと

「scoreの中のaについて」ですね。

つまりこの文は、

scoreの中のaについてprint aを実行して！

という意味だと思ってください。

リストとforに触れてみよう⑥

よって、これを実行すると、

```
score = [80, 70, 60, 100, 40]
for a in score:
    print a
```

```
% python sample4-2.py
80
70
60
100
40
```

となります。

つまり、リストの要素を一つずつ取って a に代入し、

以降の文を実行する、という動作をします。

リストとforに触れてみよう⑥

よって、これを実行すると、

```
score = [80, 70, 60, 100, 40]
for a in score:
    print a
```

ここのスペースは忘れずに

```
% python sample4-2.py
80
70
60
100
40
```

となります。

つまり、リストの要素を一つずつ取って a に代入し、

以降の文を実行する、という動作をします。

リストとforに触れてみよう⑦

最初の方針だと、こうなるので随分簡単になりました。

```
score0 = 80
score1 = 70
score2 = 60
score3 = 100
score4 = 40
print score0
print score1
print score2
print score3
print score4
```

```
% python sample4-3.py
80
70
60
100
40
```

リストとforに触れてみよう⑧

練習4

practice4.pyに適当なリスト a があります。

先ほどのサンプルと同様に

a の要素を一つずつ出力した後、

a の平均を出力してください。

ヒント

　正解は右図の通り

　最後が50になった人はおしい！

```
14
40
50
60
90
50.8
```

補足

リストの要素を取り出すのとは別に、
「5回ループする」というような動作も
もちろんfor文で実現することができます。

この時はrangeという関数を使って次の様に書きます。

```
for i in range(5):
    print "Hello"
```

```
% python sample4-4.py
Hello
Hello
Hello
Hello
Hello
```

基礎編終わり

関数やディクショナリ、クラスなどPythonを効果的に使う上で非常に大事な概念を本当に何個も飛ばしました。これで興味を持ってくれた方は、関数やディクショナリくらいは理解しておくと非常に便利なので最初に示したページなどで学習されるのをお勧めします。

応用編

スクレピングしてみよう

スクレピングとは？

スクレイピング（ウェブスクレイピング）は、
Webページの内容を取得してきて、
そのHTMLデータから必要な情報を抽出する事です。

最初に実行した weblio.py は
weblioでの単語の検索結果をスクレイピングして
いい感じに出力しています。

もう一度中身を見てみる

```
import requests
from bs4 import BeautifulSoup


print "Search word :",
word = raw_input()
r = requests.get("http://ejje.weblio.jp/content/"+word)
soup = BeautifulSoup(r.text, "lxml")
dicts = soup.find_all("div", class_="hideDictPrs")

if len(dicts) == 0:
    print "There's no such word."
else:
    top_dict = dicts[0]
    statements = top_dict.find_all("div", class_="level0")

    for s in statements:
        print s.get_text()
```

実際のウェブページのHTMLを
見ながら進めていきます。

補足

BeautifulSoupには便利な関数がいっぱいあるので
実際にスクレイピングを行う場合は、
下記のページを参考にすると良いと思います。

Beautiful Soup 4.2.0 Documentation の日本語訳
http://kondou.com/BS4/

PythonとBeautiful Soupでスクレイピング
http://qiita.com/itkr/items/513318a9b5b92bd56185