Python Tutorial #1
Python Tutorial （始めに）
インストールと起動の仕方

米国の主要大学で1年生向けのコンピュータ教育で教えられているオブジェクト指向言語Pythonを習得することを目指しましょう。オブジェクト指向言語が何であるかも知らなくても大丈夫です。

Pythonは通常の計算機言語と比較して極めて単純で覚えるべき予約命令語(command)の数が30個程度しかないので、アルファベットを覚える程度である。Pythonは、アプリの開発に使用されることは当然として、数値計算、データ処理、画像・音声処理、グラフィックの作成・編集など、ほとんどすべての作業をシンプルに遂行できる。

Pythonのバージョンは2.xと3.xに分かれている。初心者に取っては、print文が2.xでは print x と書き、3.x版ではprint(x)と書くという程度の相違と思っていれば良い。ここでは、最新バージョンのPython3.4.2を使用するが、Python3.3版を用いても同じ結果となります。

Pythonはフリーソフトであるので、無料でダウンロードできます（Macには最初から梱包されている）。日本でのPythonサイトは、www.python.jpです。各自のPCにインストールされていないときは、このサイトからダウンロードして下さい。日本語のサイトでPython3.4.2のダウンロードをクリックすると、Pythonの英文ホームページのダウンロードページに飛びます。このページの下の方に位置するFilesという見出しの下にあるMac OSX (64bit) installerあるいはWindows (X86) MSI installerをダウンロードして下さい。ダウンロードした後の操作は各PCの手順に従ってインストールして下さい。

インストールが完了すると、IDLE iconがデスクトップ（MacではLaunchpad）に現れる。このIDLEをクリックすると、Pythonのshellコンソールがデスクトップに出ます。(以下のようなWARNINGが出ても無視して、先に進んで下さい。Python3.3をインストールしたときは出ないと思います。)

```
Python 3.4.2 (v3.4.2:ab2c023a9432, Oct  5 2014, 20:42:22)
[GCC 4.2.1 (Apple Inc. build 5666) (dot 3)] on darwin
Type "copyright", "credits" or "license()" for more information.
>>> WARNING: The version of Tcl/Tk (8.5.9) in use may be unstable.
Visit http://www.python.org/download/mac/tcltk/ for current information.

>>>
```

これでPythonを使用可能な状態、つまり、コマンドが入力できるようになります。このshellコンソール上で、プロンプト（>>>）の後にPythonのコマンドを入力して操作します。すべてのコマンドの入力では英文入力に切り替えてください。最初に、

```
>>>x=’Hello World’
```

と入力して下さい。次のコマンドライン(>>>)が出ます。そこで

```
>>>print(x)
```

と入力して下さい。Hello World と返されます。（日本語の処理については後で説明します。）ここまで来れば大成功です。

なお、スマホでも同じように Python を使用できます。この場合は、それぞれのスマホの対応 Python アプリをインストールします。ですが、PC での操作を理解した後にすることを勧めます。 今回はここまでですが、成功を祈ります。次回は変数の取り扱い方法について学習します。

Python Tutorial 第 1 回（変数と四則演算）

Python を起動して、IDLE のシェルウインドウでプロンプトの後に命令を入力する所までは成功したと思います。Python は入力された命令文（コード）を上から順番に一個ずつ処理して行くので、スクリプト言語とも言われる。命令文に間違いがあれば、そこでエラーがでます。

Python3 をスマホやタブレットでも活用できる。iOS では、iOS のための Python3.3(有料 100 円、3.4 は 200 円です)がシンプルで便利です。Android 系では、QPython3（無料）をインストールすると便利です。基本的には会話型シェル上のプロンプトの後に命令文を入力する形式です。インストールした後、前回取り上げた命令文を入力してみて下さい。エラーが出なければ、OK です。

Python の命令文や関数が処理できるデータは数値データと文字データである。これらの処理対象データを一括してオブジェクトと呼ぶ。オブジェクトを処理する方法を指示するのが命令文であり、関数である。数値データや文字データに操作を加えるためには、これらのオブジェクトを区別して保存する入れ物（器）が必要で、この入れ物（器）に名前（名称）をつける必要がある。名前をつけた入れ物（器）が変数と呼ばれる。

x=’Hello World’という命令文は、x という名称の入れ物（変数）に Hello World という文字データを格納しなさいと指示している。引用符’…‘（”…“でもよい）はオブジェクトが文字データであることを指定している。x=”Hello World”と書いても同じです。数値データの場合は、x=2.46 などと書く。print(x)は変数 x の内容をシェルウインドウに印字することを指示している。（なお、文字データは unicode の体系を使用し、符号化形式は utf-8 ですが、この段階では理解不能でも結構です。）

変数を処理するに当たり、使用可能は四則演算の演算子は以下の通りです。

足し算	+	2.0+4.5 =6.5
引き算	-	2.0-4.5 = -2.5
掛け算	*	2.0*4.5 = 9.0
割り算	/	2.0/4.5 = 0.4444…
累乗	**	2.0**4.5 = 22.6274…
余り	%	4.5%2.0 = 0.5

足し算をさせるときは、+ を使用、引き算をするときは - を、掛け算をするときは * を、割り算をするときは / を、累乗を計算するときは ** を使う。（なお、excel の表計算関数では累乗は ^ を使うので、注意し下さい）オブジェクトが数値データのときは、数値計算そのものです。Python を関数電卓として活用することができます。

オブジェクトが文字データのとき、様相は相当異なります。例えば、

```
>>>x=’Hello World.’
>>>y=’Good Day!’
>>>z=x+y
>>>z
```

と入力すると、’ Hello World. Good Day!’とシェル・ウインドウに計算結果が表示される。確認してみて下さい。

Python を電卓として活用する方法を説明します。ファイナンス理論の最初に登場する投資プロジェクトの現在価値の計算式 $PV = C\frac{1}{r}(1 - \frac{1}{(1+r)^t})$ を取り上げる。これは、C のキャッシュフローが t 期間続くときの現在価値で、r は割引率（金利）である。C=100 万円、t=5 年、金利 3%(r=0.03)とする。

```
>>>c=100
>>>r=0.03
>>>t=5
>>>pv=c*1/r*(1 - 1/(1+r)**t)
>>>pv
```

と順番に入力すれば、答えが返ってくる。答えは 457.97…となる筈です。

通常、数値計算は四則演算だけでは済まない場合の方が多い。様々な数値計算をするためには、数学関数を使用する必要に迫られる。一般的な対数・指数関数や三角関数は Python の標準モジュールに組み込まれている。このモジュールは math という名称で呼ばれている。この math モジュールを使用するための準備を以下のようにする。最初に、Python shell コンソールで、

```
>>>from math import *
```

と入力する。この命令文は、math モジュールに組み込まれているすべての関数を Python shell で使用することを宣言している。Math モジュールにある自然対数関数を使って計算するときは、

```
>>>log(15.2)
```

と入力する。log(15.2)を計算して、答えをシェルコンソールに返してくれます。試してみて下さい。2.7212… と返されたら成功です。Math モジュールにある関数などの標準モジュールの内容については、以下のサイトで説明されています。

http://docs.python.jp/3/

Python 3.4 バージョンに対応する日本語ドキュメントがアップされました(2015 年 1 月 18 日付け)。一般的な三角関数、指数・対数関数はもちろん使用できます。ちなみに、ファイナンス理論での連続複利を用いた現在価値計算式、例えば、t 年後のキャッシュフローC の現在価値は $PV = Ce^{-rt}$ と計算される。C=5,r=0.03,t=5 とするとき、

```
>>>100*exp(0.03*5)
```

と入力すれば、計算できます。実行してみて下さい。Math モジュールを使用するときは、その前に必ず「from math import *」を宣言する必要があります。

Math モジュールを使用するために、以下のように最も簡単な宣言文

>>>import math
を入力することもできます。が、この場合には、モジュールの math を必ず関数の前に付ける必要があります。上の例でいうと、
>>>100*math.exp(0.03*5)
と入力する必要があります。モジュール名 math を付けたくないときは上で説明した宣言文「from math import *」を使って下さい。

Python Tutorial　第２回　（リスト型変数とスライス）

　オブジェクトの型の種類について若干説明します。変数に単一の数値列、文字列を格納することは単純で分かり易いが、一つの変数に複数のデータを格納できればより便利さが増す。数学でいう集合を考えると分かり易い。複数のデータを順番に配列して格納した変数をリスト型という。リスト型変数はデータを一列に並べたオブジェクトで、
>>>a=[1.0,2.0,3.0,4.0]
の形式で、各データをコンマ,で区切って順番に並べて、鍵括弧[]で括ります。文字データも同じように、
>>>b=[‘spam’, ‘eggs’, 100, 1234]
と定義します。リスト内の配列の番地（インデックス）は 0 から始まる（1 からではないので、注意が必要）。番地 0 には’spam’、番地 1 には’eggs’、番地 2 には数値 100 となっている。だから、変数 b のインデックス 0 に対応するデータは’spam’、インデックス 2 に対応するのは数値 100 です。インデックス 0 の内容を b[0]、インデックス 1 の内容を b[1]と書く。だから、
>>>b[1]
と入力すると、’eggs’とコンソールに表示される筈です。以上をやってみて下さい。これができれば、インデックス番号を指定して、その内容を入れ替えることが簡単にできます。
>>>a[2]=20.0
とデータを入れ替えて、変数 a の内容を印字してみて下さい。
>>>a[2]=a[2]+20.0
としたらどうなりますか？面白いですね。

　リスト型変数の内容を配列番地（インデックス）を利用して、様々なデータの処理を行うことをスライシングという。
>>>x = [1.0,2.0,3.0,4.0,5.0]
としたとき、この変数の中から幾つかの要素だけを取り出したい、例えば、配列インデックス 1 からインデックス 2 までのデータを取り出したいときには、
>>>x[1:3]
とします。1：3 の意味は、インデックス 1 から 3 未満までの要素を取り出すことを意味する。インデックス i から j までの要素を取り出すときは、i:j+1 とおきます。要注意です。
インデックスの指定を省略することができます。例えば、
>>>x[:4]
とおくと、: の前の 0 が省略されていると見なされる。この結果、[1.0,2.0,3.0,4.0]が取

り出され、印字される。反対に、
>>>x[2:]
とおくと、インデックスの 2 から最後の配列要素まで取り出される。[3.0,4.0,5.0]と表示される。データ数を増加させることも容易にできる。
>>>y=[0.1,0.2]
>>>z=x+y
>>>z
とおくと、z の要素が[1.0,2.0,3.0,4.0,5.0,0.1,0.2]となっていることが確認できる。要素を削除するときには、del 文を使う。例えば、
>>>del z[2]
>>>z
とおくと、変数 z の中から 3.0 が削除されて、データの数が減っていることが確認できる。z の中で最大値を取り出す関数は max(z)であり、最小値を取り出す関数は min(z)である。ちなみに、
>>>max(z)
とおくと、5.0 と表示される。関数の前に余計な空白をおくと、エラーとなります。データの数、つまり変数に格納されている要素数を調べるには
>>>len(z)
とおく。

データ処理では、要素を並べ替える必要性が頻繁に起きます。Python では、このような操作はオブジェクト変数の後ろに、メソードと呼ばれる命令文（関数）を付けて処理される。例えば、数値を小さい順から並べ替えるには
>>>z.sort()
とおくと、z の内容が小さい順から配列される。z の内容は並べ替えられているので、
>>>z
とおいてみると確認できる。命令文 sort()の括弧 () は必ず付けて下さい。大きい順から並べ替える方法は、z.reverse()と入力すれば良い。試してみて下さい。
多次元配列及び辞書型変数も活用できますが、ここでは簡単なリスト型変数の処理についてまでにします。次回はプログラムをファイルとして保存して、このファイルを呼び出して実行する方法を学びます。

再度、配列番地の理解を復習しましょう。オブジェクトのリスト型配列のスライスの働きかたをおぼえる良い方法は、インデクスが要素と要素のあいだ（between）を指しており、最初の要素の左端が 0 になっていると考えることです。そうすると、 n 個のデータからなる配列中の最後の要素の右端はインデクス n となります。x=[a,b,c,d]のとき

| a | b | c | d |
0 1 2 3 4

とインデックスは割り振られる。i から j までのスライスは、それぞれ i と付いた境界から j と付いた境界までの全ての文字から成っています。

単一の数値データではスライスはできませが、文字列データならば、スライスが可能となる。例えば、word='Python'であるときを考える。インデクスが文字と文字のあいだを指しており、最初の文字の左端が 0 になっていると考える。そうすると、 n 文字からなる文字列中の最後の文字の右端はインデクス n となります。インデックスは以下のように割り振られる。

```
 | P | y | t | h | o | n |
 +---+---+---+---+---+---+
 0   1   2   3   4   5   6
-6  -5  -4  -3  -2  -1
```

1 行目の数字は文字列の 0 から 6 までのインデクスの位置を示している。(2 行目は対応する負のインデクスを示している。)

Python Tutorial　第 3 回　(スクリプトファイルの作成)

一度作成したプログラムをファイルとして保存して、再度このプログラムを使用する必要が生じる。プログラム言語を用いて何かの処理をしたいときは、文法と呼ばれるルールに従って、多数の命令の集まりからなるプログラムを作成する。これをソースコード、あるいはコードという。Python では、適当な text editor を用いて、ソースコードからなるファイルを作成し、それを拡張子.py を付けたファイルとして保存する。実行可能なコードを記述したプログラム・ファイルをスクリプトファイルという。ですから、Python のスクリプトファイルを作成するためには、text editor が必要です。

Python の起動画面 Python Shell から、保存したスクリプトファイルを呼び出して実行する。IDLE の Python Shell 画面上で、「file」→「open」でこのファイルを呼び出して、「run」→「run module」とクリックしてプログラムを実行する。

日本語処理ができる text editor が重要です。Python Shell はエディターの機能も併せ持っているが、日本語の処理には不向きです。初心者向けのフリーソフトとして、Windows 向けには TeraPad、Mac 向けには mi という text editor が操作し易いと思います。上記のソフトをそれぞれダウンロードして、PC にインストールする必要がある。使用方法は簡単ですから、トライアンドエラー方式で使ってみて下さい。エンコード方式は unicode で utf-8 に設定して下さい。Notepad は使用しない方が無難です。Python のアプリ開発のためのフリーソフトの Eclpise や Pyscripter は初心者には取り扱いが難しいでしょう。

スマホの Python アプリには通常 text editor が内蔵されていて、多くの Python アプリでは日本語の処理も可能になっている。ただし、スマホの Python アプリは Python 本体だけがインストールされていて、複雑な数値計算用関数や高度な作図機能を持つモジュールは組み込まれていない。

text editor を用いて Python のスクリプトファイルを作成することから始めよう。text editor を用いて、以下のような文を書いて下さい。

```
# coding:utf-8
# 日本語のエンコーディングと印刷例
a='Console 上には日本語で「赤い花」と表示できますが、'
b="matplotlib.pyplot のグラフィック出力では日本語は使用できません。日本語の部分が空白となります"
print(a)
print(b)
```

第 1 列の#記号はその後がコメントであることを示します。coding:utf-8 は符号化方式が utf-8 であることを明記するため書いていますが、省略してもかまわない。Python は#の後の文章を読飛ばし、何の処理もしません。改行 (CR) 入力が各命令文の終わりを表します。このファイルに名前＋拡張子 (.py) を付けて (例えば、test.py として) 保存して下さい。拡張子 (py) が Python のスクリプトファイルであることを表現する。Python のスクリプトファイルを保存するディレクトリを作成し、常にその下のフォルダーに保存するようにすると他のファイルとの区別が容易です。

保存したファイルを IDLE コンソールで開いて下さい。その後、[run],[run module]の順に実行して下さい。ファイルの命令文が順番に実行され、日本語の文章が Python シェルのコンソールに表示されます。エラーが出なければ、成功です。

Python Tutorial 第 5 回　(条件文)

条件文は命令実行の順番を管理するため使用されます。すべてのプログラミング言語と同じく、この説明は抽象的になるので、分かりにくいと思いますが、思考のトレーニングだと思って我慢してください。具体的な例を理解すれば良いと思います。最初に、条件文の一つとして while 文を説明します。while 文は以下のような構造をしている。

```
while condition 文 :
 (インデントの空白)  while-body の記述
```

上記の while 文は、condition 文で記述されている条件が成立している間は、while-body で記述された文を実行することを指令している。while-body の記述はすべて 1 字以上の空白でインデントされなければならない。例として、以下のスクリプトを取り上げる。

```
# Fibonacci series: the sum of two elements defines the next
a, b = 0, 1
while b < 10:
    print(b)
    a, b = b, a+b
print('the end of computations')
```

while b < 10: はbの値が10未満の間は、空白によってインデントされた記述群を実行させる命令である。whileのループは、while文以降に記述された空白のインデントがなくなる最後の行で終わります。ここでは、print('the end of ...')の行で、while文のループから抜けます。始めに、a=0, b=1と与えられているので、while文を始めるときはb=1なので、print(b)を実行し、次に、a=b, b=a+bを実行する。ここで、b=1なので再び、同じ計算がおこなわれ、a=1,b=2となる。b=9になるまで、これが繰り返される。b=10になると、while文から抜け出して、インデントがなくなった行に行き、print('the end of ...')文が実行される。

このスクリプトファイルをテキスト・エディタで自ら作成・保存して、IDLEで開いて実行してみて下さい。成功を祈ります。

つぎに、条件文forの使用方法について説明します。条件文の一つであるfor文は以下のように使用される。

```
for condition文 :
    for-bodyの文
```

Condition文の条件が成立している間は、for-bodyで記述された文を実行する。ここでも、for-bodyの記述はすべて1字以上の空白でインデントされなければならない。以下に簡単な例を示す。以下のファイルを作成し、保存して下さい。

```
# ″for″ sentences
a = ['dog', 'window', 'mountain']
for x in a:
    print(x, len(x))

# Print out the value from 0 to 49 inclusive.
for value in range(0,50):
    print(value)
```

len(x)は変数xの長さ(length)を計算する命令である。range(0,50)は0から49までの数値を発生する関数である。この保存したファイルを開いて、実行して下さい。

```
dog 3
window 6
mountain 8
0
1
2
.
.
.
49
```

と表示されれば、成功です。

つぎに、if 文について説明する。if 文は以下のように使用される。

```
if condition-1 :
    if-body の文
elif condition-2:
    elif-body の文
else:
    els-body の文
```

もし condition-1 が真であるならば、if-body の文を実行する。condition-1 が成立しないとき、condition-2 が成立するならば、els-body の文を実行する。condition-1 も condition-2 も両方とも不成立ならば、els-body の文を実行する。elif は else if の省略形である。elif 文と else 文は省略可能である。以下のように、スクリプトファイルを作成して、保存して下さい。

```
# ″if″ sentences
x=int(input('整数を入力して下さい: '))
if x < 0:
  x = 0
  print('負の整数なので、ゼロとした')
elif x == 0:
  print('ゼロ')
else :
  print(x,'は正の整数です')
```

ここで、int(..) は数値を整数にする命令であり、input はキーボードからの入力を受け取る命令である。この例にあるように、条件文において使用可能な論理式は、
< (lesss than),
> (greater than),
== (equal to),
<= (less than or equal to),
>= (greater than or equal to),
!= (not equal to)
等である。このスクリプトファイルを実行してみて下さい。Please enter an integer と表示されたら、好きな整数を入力してみて下さい。成功を祈る。

Python Tutorial　第６回　（関数の定義）

同一の処理を何度も繰り返すとき、この処理方法を関数として定義しておいて、使用するごとにこの関数を呼び出して使うことは便利である。関数を定義する方法について説明する。関数を定義する形式は以下の通りである。

```
def function_name(args_list):
(インデント)function_body の文
```

第 1 行目 function function_name:は function_name という名称の関数を定義する文である。function_name は、例えば、fibonachi というような英文の適当な名前をつけて良い。ただし、Python で組み込み関数で使用されている関数名（予約語、例えば、cos、log など）は使用できない。args_list は、後の例で説明されるように、関数で使用される変数のリストを入力する。以下のフィボナッチ数列を定義する例を取り上げる。

```
# Fibonachi series

def fib(n):
    a,b=0,1
    while b < n:
        print(b)
        a,b=b,a+b

# Now call the function we just defined.
fib(2000)
```

Def fib(n):文は関数名 fib で関数を定義し、変数が n であることを記述する。＃Now … の行の上までが関数の定義になっている。最後の行にある fib(2000)は、上で定義した関数 fib を呼び出して、変数の値を n=2000 とした時の関数値を求めている。このスクリプト作成し、保存して下さい。このスクリプトを開いて、実行させて下さい。

1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 ¥¥

という結果が返されれば、成功です。

また、関数スクリプトを独立したスクリプトとして、つまり、モジュールとして作成することもできる。この場合、通常のスクリプトファイルと同様に、関数スクリプトを拡張子.py を付けて保存する。以下のような関数スクリプトを作成しよう。ここで、result =[]文は変数 result を空集合のリスト型変数として定義することを宣言しているが、内容は空集合となっている。result.append(a)は、変数 result に a の値を一つの要素と付け加えることを実行させる。したがって、result はフィボナッチ数を要素としてもつリスト型変数となる。

```
def fib(n):
    result = [ ]
    a, b  = 0, 1
    while a < n:
        result.append(a)
        a,b=b,a+b
    return result
```

return 命令は、変数 result の内容を関数値として返すことを実行している。この関数スクリプトを、例えば、fibo.py というファイル名で保存する。Python はこの関数スクリプトをモジュールの一つと理解し、標準モジュールに加える。従って、この関数を使用するときは、この関数スクリプトをモジュールとして呼び出す必要がある。「from fib import *」と宣言して、fib(200)と入力すると、関数値をスクリプトに返してくれる。以下のような別のスクリプトからこのモジュールを呼び出して使う。

```
from fibo import *
y=fib(200)
print(y)
```

これを実行すると、関数スクリプトで計算された result の内容が変数 y に引き渡される。フィボナッチ数列が表示されれば成功です。