**Pythonの基本変数型**

**変数の型は、代入する値の型によって決まる**

**整数型 (integer)**

n = 10

**浮動小数点 (実数) 型 (floating point)**

x = 10.0

x = 10.0e5

**文字列型 (string) "" あるいは ’’ でくくる。どちらも意味は同じ。**

s = “abc”

s = ‘abc’

　文字列内に ’ や ” が含まれる場合、くくる引用符を変えることで表現できる。  
s = “a’bc”

あるいは、’\’ を使って特殊文字をエスケープする。

s = ‘a\’bc’

　　　改行文字 (LF) \r

　　　行頭復帰文字 (CR) \r

　　　タブ \t

直後の1文字 \\ \ 文字自身

\’ ‘

\” “

s = ‘a\’bc’ 文字列内で ‘\’ があると、直後の1文字と組み合わせて

**リスト型 (list) 複数の要素を持つ配列**

d = []

　要素を追加する場合

d.append(3.0)

　　　追加する値の型により、その要素の型は変わる => 一つのリストに、異なる型の要素を持たせられる

　i 番目の要素を参照

　　　d[i]　　　　　　注意：Basic, Fortranなど以外のプログラム言語では、  
要素番号 i は0から始まり、N-1 まで (Nは要素数)

**タプル型 (tupple)　リスト型に似ているが、要素の書き換えができない => 定数リスト**

d = ()

　書き換えができないので、.append() は使えない

　i 番目の要素を参照

　　　d[i]

**オブジェクト**

**オブジェクト (Object): 複数の属性 (attribute, property) と 操作 (method) を結び付けた変数**

**pythonの場合は、属性と操作をいずれもattributeと呼ぶ**

**Pythonの変数はすべてオブジェクトである。**

**オブジェクト変数の属性へのアクセス:**

**インスタンス変数名.属性名**

**インスタンス変数名.操作名(引数)**

　オブジェクトの定義： クラス (class)

　クラスを変数として実体化： インスタンス (instance)

　オブジェクト指向プログラミング: クラス、オブジェクトを中心としたプログラム言語、プログラミング

　　　　　⬄　関数型プログラミング: Cのように、関数、サブルーチンで構造化したプログラミング

例：

catオブジェクト： propertyとして、color, weight, methodとして move(), position()などが考えられる。

クラスの定義 (概念的なプログラムリストなので以下のままでは動かない)：

class cat():

# pythonでは、 # 以降は行末までがコメント文として、無視される

# 関数は def文で定義し、: で終わる

# 変数は 変数名 = 初期値 とすることで、初期値を設定できる

def \_\_init\_\_(self, color = 'white', weight = 20.0):

color = color # ややこしいけど注意。左のcolorはcatのattribute変数

# 右辺のcolorは\_\_init\_\_関数の引数変数

weight = weight

x = 0 # 位置変数 x,y,zは0で初期化しておく

y = 0

z = 0

def move(dx, dy, dz): #位置をdx,dy,dzだけ移動させる

…

def position():

return x, y, z #位置を [x,y,z]リストで返すメソッド

　インスタンスの生成（変数の作成）

　　nyanko = cat()

catクラスで定義されたオブジェクトのインスタンス変数nyankoをつくる。

引数を指定していないので、初期値が使われ、

naynako.color は 'white'、nyanko.weightは 20kgになる。

　　nyanko = cat(color =’black’)

catクラスで定義されたオブジェクトのインスタンス変数nyankoをつくる。

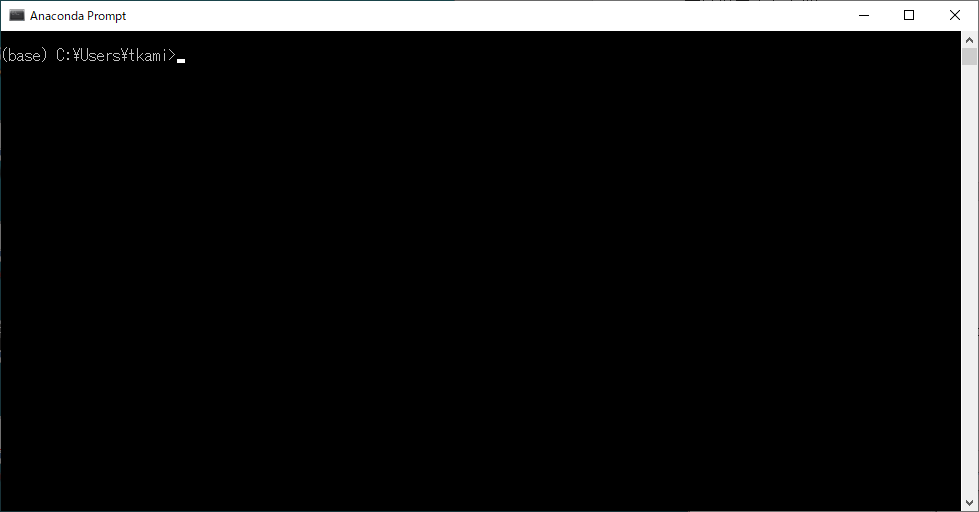
引数で指定されたcolor変数は初期値から変更され、naynako.color は 'black' になる

　　nyanko.move(3, -1, 0)を実行すると、nyanko.x, nyanko.y, nyanko.zはそれぞれ 3, -1, 0 になる

　　nyanko.move(-1, 3, 2)を実行すると、nyanko.x, nyanko.y, nyanko.zはそれぞれ 2, 2, 2 になる

**pythonの起動**

1. Windowsのスタートメニューから “Anaconda Prompt” を起動  
(pythonへのPATH変数が通っていれば、通常のコマンドプロンプトでも同じ)



2. 作業フォルダーに移動。例えば、F:\workへ移動するなら、  
(赤字部分を入力し、Enterを押す)

(base) C:\Users\tkamiya>f:

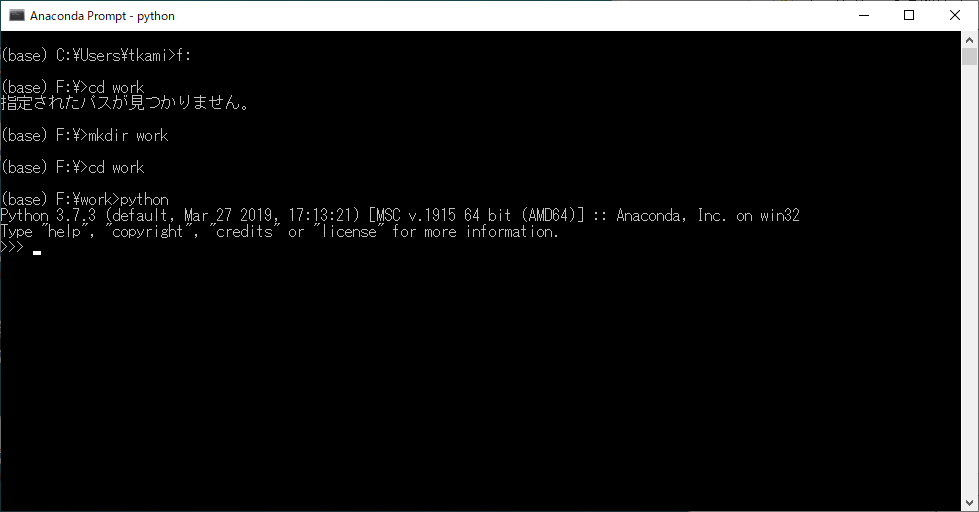
(base) F:\>cd work

(base) F:\work>

**対話モード (interactive mode)**

1. Pythonを引数なしで起動

(base) F:\work>python



“>>>” は、pythonの入力プロンプト

ここで、pythonのプログラムと同じコマンドがすべて使えるので、文法の確認などをするといい。

**対話モードで変数の型、扱いを確認してみる**

**>>> a=10 整数の代入**

>>> print(a)

10 整数として表示

**>>> a=10.0 実数として代入**

>>> print(a)

10.0 実数として表示。デフォルトでは最小の小数点で表示される。

>>> a=10.00 桁数が違う実数として代入。

>>> print(a)

10.0 実数として表示。デフォルトでは最小の小数点で表示される。

**>>> a="10.00" 文字列として代入**

**>>> print(a)**

10.00 文字列のため、代入した値そのままが表示される

**>>> a=[10.00, 10, '10.000'] リストとして代入**

>>> print(a)

[10.0, 10, '10.000'] リストの要素が [] にくくられて表示。要素ごとに型が違うことがわかる

>>> a[1]=3 要素番号 (index) 1 の要素を 3 に変更してみる

>>> print(a)

[10.0, 3, '10.000'] 2つめの要素が3に変更されている（indexは0から始まることに注意）

>>> a.append(5) リストに要素を追加

>>> print(a)

[10.0, 3, '10.000', 5]

**>>> a=(10.00,10,'10.000') タプルとして代入**

>>> print(a)

(10.0, 10, '10.000') タプルの要素は () にくくられて表示される。

>>> a[1]=3 index 1 の要素を 3 に変更してみる

Traceback (most recent call last): Trace: "痕跡をたどる” => incidentが生じた場所を探すこと

File "<stdin>", line 1, in <module>

TypeError: 'tuple' object does not support item assignment

タプルの要素は変更できないのでエラーになる

>>> a.append(3) タプルに要素を追加してみようとする

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

AttributeError: 'tuple' object has no attribute 'append'

appendという属性(attribute)がないのでエラーになる

**>>> exit() pythonスクリプトを抜ける関数。この場合は対話モードから抜ける**

(base) F:\work>

**CSVファイル**

Comma Separated Valuesの略。

1行ごとに複数の値を‘,’でくぐって並べる。

先頭行に、各値のラベル (Label) を書くことが多い

Excelで作成 => ファイルの保存で “CSV” を選ぶ

Excelで直接読み込める

例：\work\data.csv

rho(ohm),T(K)

0,1

1,2

2,5

3,8

4,15

**CSVファイルの読み込み**

**注意： pythonでは、追加機能を module で追加する。**

**どのようなmoduleが使えるかは、とにかく検索するに限る**

**“python csv” で検索**

**\work\01-readcsv.py**

import csv csvを読み込むためのmoduleを読み込む

from pprint import pprint リストを表示する際に成形して表示するmodule pprintを読み込む。

import pprintでもよいが、実際のpprintは、pprintモジュール内の pprint関数であるので、  
呼び出す際には pprint.pprint (モジュール名.関数名、あるいはモジュール名.クラス名) とする必要がある。  
from module import XXX 文を使うと、module.py中のXXXという関数だけをインポートし、関数名で呼び出すことができる。この場合は、下記のように、pprintだけで呼び出しができる。

infile = 'data.csv' 読み込むcsvファイルのパスを infile 変数に設定

lines = [] csvから読みこむ各行のデータのリスト変数を作成

with open(infile, "r") as f: infile を読み込み専用モード ‘r’ で開き(open)、

ファイルアクセスに必要なファイル変数を f に代入する。

Openしたファイル変数は、使用が終わったらcloseしなければならない

(closeしないと、他のプログラムからファイルにアクセスできなくなる)。

with 文を使うと、withブロックを抜けた時に、

自動的にファイルをcloseしてくれるので便利

reader = csv.reader(f) csvモジュールの属性.readerを使い、fに紐づけられているファイルを  
csvとして読み込むためのreader変数を設定

pythonでは、インデント (indent, 字下げ)数でブロックを区別する。  
同じブロックは同じだけのインデントをする必要がある

for row in reader: csvファイルを1行ずつ読み込む定型文。

Readerから1行ずつ読み込み、row リスト変数に各要素を代入

lines.append(row) とりあえず、各行のリスト変数を、linesリスト変数に追加しておく

linesは、2次元の配列 (リストのリスト) になっていることに注意。

　 j行目のデータのリストは lines[j] で取得できる。

j行目のk番目のデータは lines[j][k] で取得できる。

(j,kのいずれも、0 から始まることに注意)

　ブロックが終わったら、for文にもどり、readerから次の行を読み込み、

row変数に代入、以下を繰り返す

header = lines[0] csvファイルの1行目はラベル行なので、header変数に代入しておく

xy = lines[1:] データ行は2行目以降なので、xyリスト変数に代入しておく。

[1:]は「リストのスライス」と呼ばれ、index 1から最後までの部分リストを返す

[3:6]とすると、index 3 から 5 (2番目のindexより１少ないことに注意)の部分リスト

print("header:", header) headerの中身を確認するため、print文で表示する

print分では、表示したい変数を , で区切って並べられる

print("lines (print):", xy) xyの中身をprint文で確認

print("lines (pprint):") xyの中身をpprint文で確認したいが、pprintでは文字列とリストを並べて

表示できないので、まず、print文で文字列を表示

pprint(xy) xyの中身をpprint文で確認。この場合はprint文を同じ表示。

**CSVファイルのデータを実数型のx,yリストに読み込み、グラフに表示**

**\work\01-readcsv.py**

# グラフにプロットしてみる

# http://ailaby.com/least\_square/

import csv

from pprint import pprint

from matplotlib import pyplot as plt

グラフに表示するため、matplotlibモジュールの pyplot クラスをimportする。

importしたクラス、変数、関数（オブジェクト）には、as 文により別名をつけられる。

この場合、 pltという別名でpyplotオブジェクトにアクセスできる

infile = 'data.csv'

i = 0 csvの変数をheaderと区別して読み込むため、行カウンタ変数を使う

まず、0に初期化する (先頭行を i == 0 で区別できるようになる)

x = [] csvの各行の変数を、x,yというリスト変数に別々に取り込むため、x,yリスト変数を作成する

y = []

with open(infile, "r") as f:

reader = csv.reader(f)

for row in reader:

if i == 0: i == 0、つまり、最初の１行のデータはheader変数に代入

　　 == 演算子は、両辺の変数の値が等しいか等しくないかを判断し、

bool値 True, Falseを返す。

if文は、与えられた値がTrueであれば次のブロックを実行する。

header = row

else: i == 0がFalseである場合に実行される

x.append(float(row[0])) xリスト変数に、各行の index 0 (最初の要素) を追加する。

ファイルから読み込んだ値は 文字列 になっているため、float()関数を使って

実数変数に変換する。

y.append(float(row[1])) yリスト変数に、各行の index 1 (２番目の要素) を追加する。

i = i + 1 行カウンタを1増加させる

print("header:", header)

print("x:", x)

print("y:", y)

plt.scatter(x , y) x,yリスト変数を.scatterメソッドに渡し、

散布図として(x,y)をプロットすることを指示する

plt.show() グラフを表示