チーム AID Python 勉強録

Ver. 0.0.1

井上裕太
http://y-ino.com

2015 年 1 月 16 日

目次

第 1 章

環境構築

1.0.1 Mac

http://qiita.com/WizowozY/items/4c0b1c0dc93017855429 <- ここを見てください

第 2 章

基礎文法編　第一回 (2014/1/16)

2.1 コメント

```
'''
この部分はドキュメント文字列として解釈される
一応複数行コメント
'''

while True:
'''
これはエラー（インデントがない）
'''

while True:
    '''
    これはおっけー
    '''

# コメント
spma = 'ham' # コメント
```

2.1.1 スクリプト構造

```
#コメント
spam = 'ham' #コメント

printf(1+1);printf('hello') #一行で複数処理を書く場合は「;」で区切る

'spam' + (  #括弧（）,[],{}で囲まれている部分ならば、その内側では自由に改行できる
   'ham' + 'egg')
```

```
#インデントによるブロック
def foo(a):
    if a > 100:
        return 1
    else:
        return 0
#コーディング規約ではインデントはスペース 4 つ分
```

2.1.2 比較演算子

演算子	比較条件
x is y	x と y は同じオブジェクト
x is not y	x と y は同じオブジェクトではない
x in y	x は y に含まれる)
x not in y	x は y に含まれない

2.1.3 辞書オブジェクト

連想配列、ハッシュテーブル
{ キー:値, キー:値,... }

```
>>> {1:'one',2:'two'}
{1: 'one', 2: 'two'}

>>> d = {1:'one',2:'two'}
>>> d[1] #要素へのアクセスはブラケット演算子
'one'

>>> del d[1] #要素の削除は del 文で行う
```

2.1.4 リストとタプル

タプル：更新不能なオブジェクトで、あとから要素を追加したり変更したりできない（辞書のキーとして使用）リスト：更新可能なオブジェクト、辞書のキーとしては指定できない
リスト [要素, 要素, 要素,...]

```
l = [1,2,3]
l = []
l = [1,'two',[3,4,5]]
```

タプル 要素, 要素, 要素,...

```
t = (1,2,3)
t = 1,2,3
```

```
t = ()
t = (1,'two',[3,4,5])
```

2.1.5　シーケンスのアクセス

シーケンス：文字列、リスト、タプルなどの配列のこと [] でアクセスする

```
>>> s = 'abcdefg'
>>> s[0]
'a'
>>> s[-1] # 後ろからの位置（最後から 1 番目 => 一番後ろ）
'g'
>>> s[1:4] # [開始位置:終了位置] 開始<=  要素 <終了
'bcd'
>>> s[-3:-1]
'ef'
>>> s[:3] # 開始、終了のどちらかを省略すると、シーケンスの先頭、末尾までを意味する
```

2.1.6　イテラブルオブジェクト

```
for 変数名 in 式:
    ...
    ...

l = [1,2,3]
for i in l: #i に l の要素が一つずつ代入される
    print(i)


----
import sys
for line in sys.stdin:#標準入出力から一行ずつ読み込む
    print(line,end='')
```

2.1.7　関数

```
def 関数名(引数リスト):
    ...
    ...
```

2.1.8　クラス

```
class クラス名:
    def __init__(self,引数,...): #コンストラクタ
        ...
        ...
```

```
    def メソッド名(self,引数,...):
        ...
        ...
```

メソッドの第一引数は、必ずメソッドが属するオブジェクトを受け取る

```
class Spam:
    def ham(self,egg,bacon):
        ...
        ...

obj = Spam()
obej.ham(arg1,arg2) # self<-obj,egg<-arg1,bacon<-arg2
```

インスタンスの属性値は、インスタンス. 属性名で指定して代入するだけで作成される=> 事前に定義する必要性はない

2.1.9 モジュール

組み込み関数、組み込み型 => いつでも使える モジュール=> 一般のライブラリ

```
import math #math モジュールをインポート
```

2.1.10 入出力

```
>>> text = input()
spam
>>> text
'spam'

>>> input('好きな食べ物は?:')
好きな食べ物は？: Spam
'Spam'
```

2.1.11 ステートメント

pass 文

何もしない文、文法上なにか必要だが何も書くことがないときに使ったりする

```
>>> if spam:
>>>     if ham:
# これだとエラー

>>> if spam:
>>>     pass # 何もしない
>>> if ham: pass
```

while **文**

```
while 条件式:
    ...
    ...
    ...
else:   # while ループを抜けたあとに実行される
    ...
```

range(stop)、range(start,stop[,step])

初期値 start に、増分値 step を加算した数列のイテレータを返す for 文のループで使う * start 最初の数値を指定する、省略時は 0 * stop 増分値 step が正の値なら数列の上限、負の値なら下限値を指定する * step 数値の増分を指定する、省略時は 1 となる

```
>>> for i in range(10): # 0 to 9 のループ
>>>     ...

>>> number = lsit(range(100)) # 0〜99 のリストを作る
```

例外処理

```
try:
    ...
except [式 as 変数名]:
    ...
else:       # エラーが発生しなかったときに最後に実行される
    ...
finally:    # エラー発生の有無にかかわらず実行される
    ...
```

例外を発生させるのは raise(throw ではない) raise ValueError(‘値が不正です’)

assert **文**

assert 文は実行時にデータや条件の整合性をチェックし、問題がないことを確認するときに使用する

assert 式 [, 式]

assert 文に指定した式が偽となった場合、AssertError 例外を送出して、処理を中断する

with **文**

ファイルオブジェクトを処理するとき、

```
fileobj = open(filename,'w')
try:
    do_something(fileobj)
```

```
finally:
    fileobj.close()
```

ファイルオブジェクトへの書き込みを確実にファイルに反映するためには、必ず try〜finally ブロックを使って、open() したファイルを file.close() する処理を書かなければならない with 文はこのようなパターンを簡単に実装するためのブロック

```
with open(filename,'w') as fileobj:
    do_something(fileobj)
```

上のコードと同義

with 文から初期化処理と終了処理を呼び出すインターフェースを持ったオブジェクトを、コンテキストマネージャーと呼ぶコンテキストマネージャーには、file オブジェクト、socket オブジェクト、Lock オブジェクトなどがある