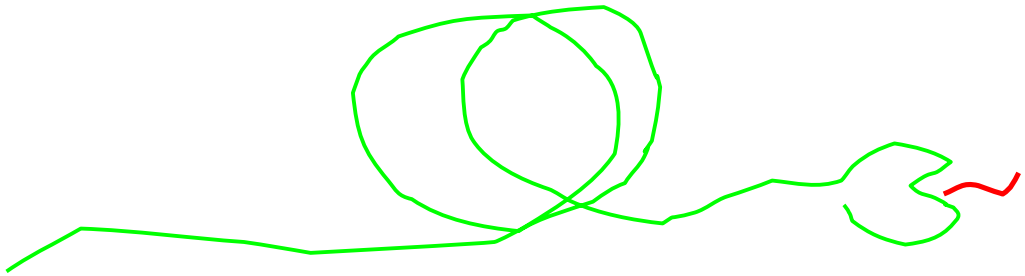
永和システムマネジメント
東京支社勉強会
Python

安井 力

Pythonひとめぐり

2005/12/28

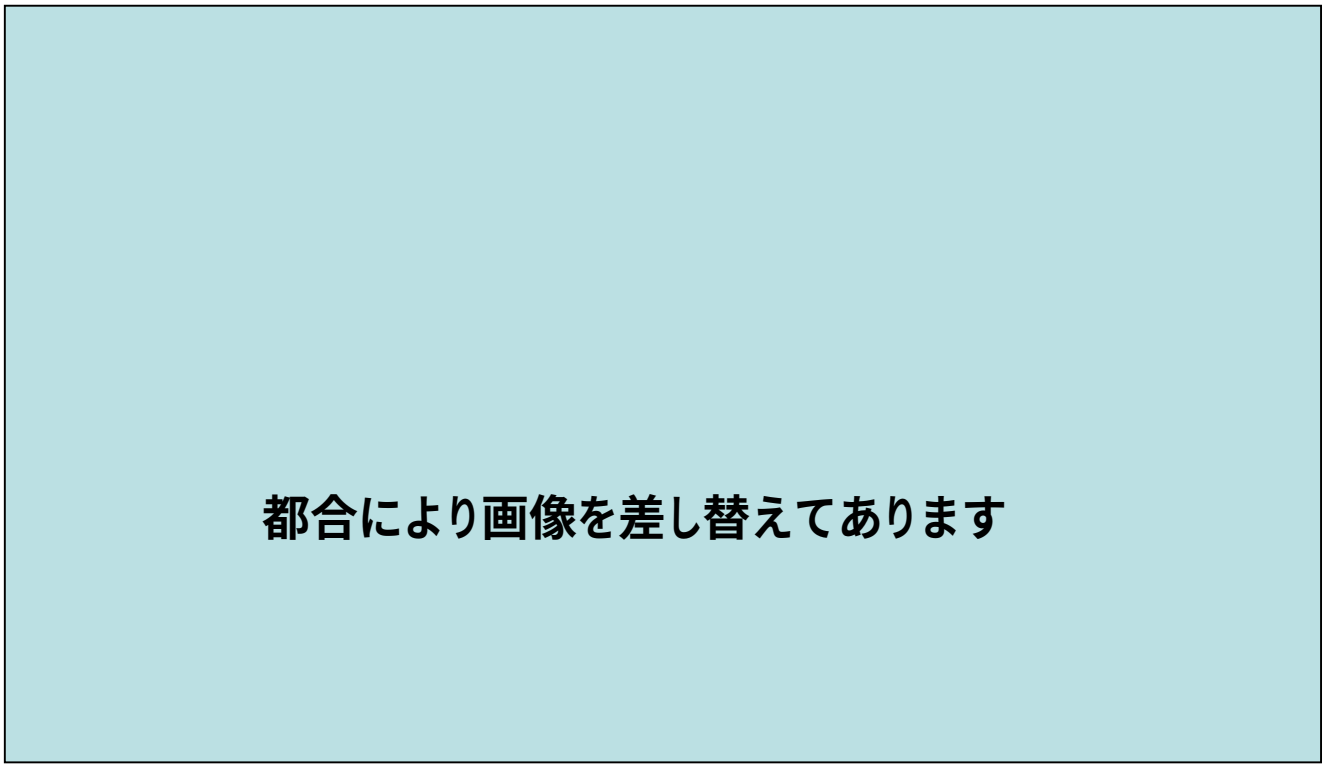
安井 力



都合により画像を差し替えてあります

- Ball Python (*Python regius*)
- **分布:西はセネガルからウガンダ辺りまでのアフリカ中部**
- **小型のニシキヘビで、最大全長でも2.5m以下で、通常では1mを少し超える程度の大きさである。性質は温和な個体が多く、外敵と遭遇した 場合でも攻撃体勢をとることはほとんどない。たいていの個体は、すぐさま独特な防御体勢をとる。その防御方とは体をボール状に丸めるといった方法であり、この種の名前の由来にもなっている。**

安井 力



Monty Python’s Flying Circus
モンティパイソンの空飛ぶサーカス

2005/12/28

安井 力

buzzwords

- **スクリプト言語**
- **オブジェクト指向**
- **欧米で大人気**
- **コンパイル可能**
- **お仕事に使える！**
- CやJavaとも連携
- Martin Fowlerも**オススメ**

(だった)

- Googleで**仕事が**
  できる！
- **扱いやすい**
- **インタラクティブ**
- Lisp**的な機能**
- **コードがキレイ**
- **周辺ツールとかも豊富**
- TOOWTDI

安井 力

データ構造

- **数字** `1  0.5  9999999999L  (1+2j)`
- **文字列** `"abc" 'def' "I'm a Pythonist"`
- **タプル** `(10, 20) (2, 3, 5, 7) (100,)`
- **リスト** `[10,20]  [1,3,4,6,8,10,12]`
- **ディクショナリ**

```
{"C":1972, "Java":1991,
   "Python":(1991, 1995) }
```


安井 力

タプルが曲者

- **不変**(immutable)

```
x = (10, 20)        x = [10, 20]
x[0] = 30 ×         x[0] = 30 ○
```

- **一発で代入可能**

```
(x,y,z)=(8,1,3)  [x,y,z]=[8,1,3]
```

- **でも実はこうも書ける**

```
x, y, z = 8, 1, 3
```


安井 力

こんなこともできちゃう
～複数の戻り値～

```
def two_values():
    return 10, 20

x, y = two_values()
```

こんなこともできちゃう
～スワップ～

x, y = y, x

安井 力

ブロックの書き方

- インデントでブロックが決まる

```
def euclid(a, b):
    if b==0:
        return a
    return euclid(b, a%b)
print euclid(42, 35)
```

ブロックの幅は自由

- 縦にそろってればブロック
- 普通は等幅が見やすいので注意

```
if f1():
  if f2():
        if f3() and f4():
            if f5():
              do_something()
              do_something2()
  else
     do_something_else()
```


安井 力

シーケンスを作る

- 1〜10の数字列
```
range(10)
```
- 2,4,8,16,32,...
```
[pow(2,x) for x
      in range(10)]
```
- 文字を1つずらす
```
[chr(ord(c)+1) for c
      in "abcde"]
```
- ファイルの中身
```
[ln for ln in fileinput.
      input("spam.txt")]
```

generator

- 読むごとに生成されるシーケンスもどき
- 「次」を読むまで、「次」のデータはない
  - ファイルの読み込み
  - 膨大な計算結果
- 1から∞までの数字のリスト

```
def f():
    i = 1
    while True:
        yield i
        i = i + 1
```

添え字とスライス

- シーケンスからひとつ取り出す

```
s="abcde"
s[0] … "a"
s[-1] … "e"
```

- 部分シーケンスを切り取る – スライス

```
s[3:5] … "de"
s[2:] … "cde"
s[1:-1] … "bcd"
s[:] … "abcde"
```

lambda式

- Lispからの遺産
- わりとふつー

```
>>>fn = lambda x: x * x
>>>fn(10)
100
>>>[fn(i) for i in range(5)]
[0, 1, 4, 9, 16]
```


安井 力

map / filter / reduce

- Lisp**からの遺産**
- **ふつー**

```
>>> map(lambda x,y: x+y, [1,2,3], [10,20,30])
[11, 22, 33]

>>> filter(lambda x: x%2==0, [1,2,3,4,5])
[2, 4]

>>> reduce(lambda x,y: x+y, [1,3,5,7,9])
25
```

関数

```
def deviation(values):
  sum = reduce(lambda x,y: x+y, values)
  avg = sum / len(values)
  dev2 = 0
  for v in values:
    dev2 += pow(v - avg, 2)
  return math.sqrt(dev2 / (len(values) - 1))
```

クラス

```
class Person:
  def __init__(self):
      self.name = ''
      self.birthday = None

  def get_age(self, at=datetime.date.today()):
      bd = self.birthday
      age = at.year - bd.year
      if (at.month, at.day) <= (bd.month, bd.day):
            age -= 1
      return age
```

ユニットテスト

```
class PersonTest(unittest.TestCase):
  def setUp(self):
      pass

  def test_get_age(self):
      p = Person()
      p.birthday = datetime.date(1973, 4, 20)
      now = datetime.date(2005, 12, 7)
      self.assertEqual(p.get_age(now), 32)
```

インスタンス変数を後付け

```
class Foo:
  pass

f = Foo()
f.name = "J.R.Luser"
f.address = ["Tokyo", "Minatoku",
             "Konan 2-4-12"]
```

メソッドだって後付け

```
class Bar:
    pass

def printname(self):
    print 'My name is ' + self.name

Bar.dump = printname
b = Bar()
b.name = 'Foobar'
b.dump()
```

こんなふうに書けば

```
class Interceptor:
  def __init__(self, method):
    self.method = method

  def __call__(self, x):
    print "before calling"
    self.method(x)
    print "after calling"
```

アスペクトっぽいことも簡単

```
class Bazz:
  def f(self, x):
      print "f(" + str(x) + ")"

>>> b = Bazz()
>>> b.f(10)
f(10)

>>> b.f = Interceptor(b.f)
>>> b.f(10)
before calling
f(10)
after calling
>>>
```

メタプログラミング

```
>>> eval("10+20*30")
610

>>> x=0
>>> exec("if x==0:¥n"
...      "  print 'NG'¥n"
...      "else:¥n"
...      "  print -x¥n")
NG
```


安井 力

大きな数の計算

```
>>> 2**10000

19950631168807583848837421626835850838234968318861924548520089498529
43883022194663191996168403619459789933112942320912427155649134941378
11175937859320963239578557300467937945267652465512660598955205500869
18193311542508608460618104685509074866089624888090489894838009253941
63325785062156830947390255691238806522509664387444104675987162698545
32228685381616943157756296407628368807607322285350916414761839563814
58969463899410840960536267821064621427333394036525565649530603142680
23496940033593431665145929773279665775606172582031407994198179607378
8245683762280037302885487251900834464581454650557 (以下2000桁以上続く)
```

- **ちょっと**Lisp**っぽい**

安井 力

組み込み型

- Unicode対応

  u"ゆにこーどもじれつだよ"

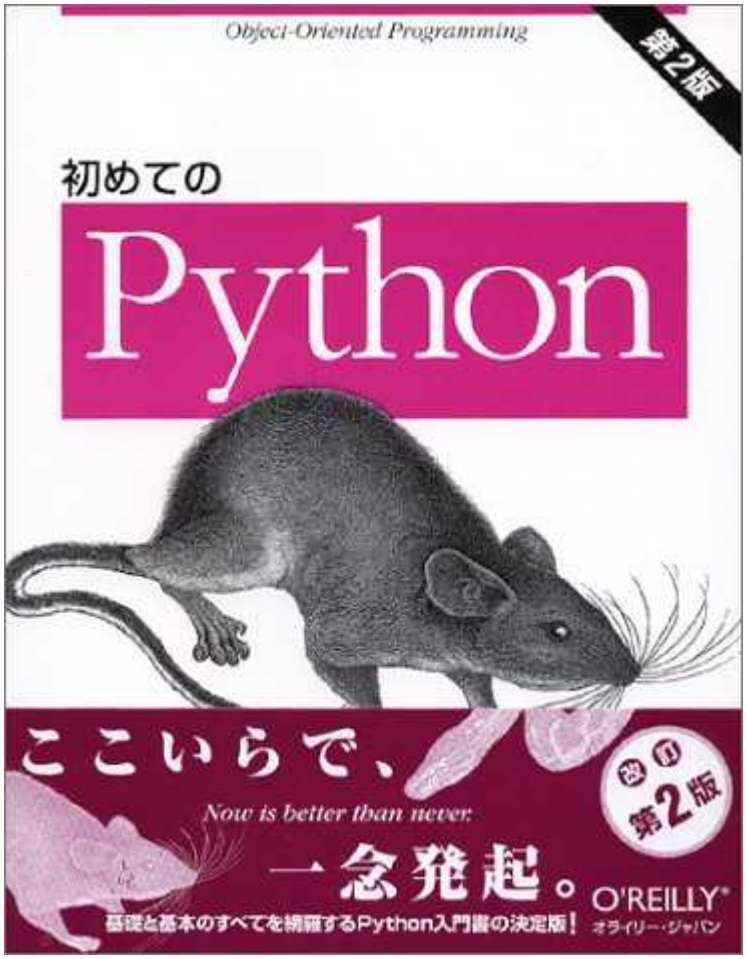
  unicode(元の文字列, 元のエンコーディング)
- 浮動小数点数
- 複素数

```
>>> (1+2j)*(1-2j)
(5+0j)
```

ライブラリ・ツール・プロダクト

- **標準モジュール群**
  - re, xml, urllib, pickle…
  - 300**以上**
- NumPy
- PyQt
- PyGtk
- wxPython
- Contract
- IDLE
- PyDev
- Bicycle Repair Man
- PyLint
- Zope
- Plone
- gadfly
- Trac

Pythonを学ぶ



「初めてのPython 第2版」

Mark Lutz, David Ascher

オライリージャパン

5040円

全715ページ

2005/12/28

安井 力

Pythonを学ぶ

- 起動と実行
- 簡単な使い方
- プログラムを書いてみる
- ヘルプ、ドキュメント
- 便利な標準モジュール

安井 力