

# プログラミング言語論 #10

## Python, Argument Handling

2018-06-25



A close-up photograph of green, pinnate leaves against a dark, blurred background. The leaves are arranged in a cascading pattern from the top left towards the center.

2 番目のオブジェクト指向プログラミング言語として Python を紹介する。  
また、関数の呼び出しにおける引数の取り扱いに関して紹介する。

# Python

## Python 3

---

拡張子	py
実行	/usr/bin/python (インタプリタ)
型に関する特徴 1	弱い動的型付け言語
特徴 2	(途中から) OOPL
それ以外の特徴	version2 と 3 で大きな変化 <sup>1</sup> インデント重要, コロン重要, 括弧激減

---

repl.it で演習

---

<sup>1</sup>この講義では version 3 を使用するので print は関数である

# Some Python Syntax

## 関数レベル

### A function in C

```
1 // intを返す1引数関数fの定義
2 int f (int x) {
3     if (x < 0) {
4         x = -x;
5     }
6     x = x + 10;
7     return x;
8 }
```

### sample1.py

```
1 # 1引数関数fの定義
2 def f (x): # 型宣言なし
3     if x < 0: # :までが条件
4         x = -x # ; ( ) 不要
5     x = x + 10 # indent重要
6     return x
```

- 動的型付言語なので型に関する宣言がない。def は Javascript の var と同じく「定義する」という意味
- いろいろな括弧がコロンに置き換わる
- インデントが違くと構文エラー（オフサイドルール）
- セミコロンはほぼ省略、コメント開始は#

# syntax

## 定義系構文，関数適用

```
1  # 大文字はメタ変数
2  # []は省略可能
3
4  import PACKAGE [as ID]
5
6  def ID(ARGS) : BODY
7
8  class ID(SUPER) : DEFS
9
10 F(ARGS)      # 関数F
11 P.F(ARGS)    # パッケージPのF
12 O.M(ARGS)    # Oのメソッド
```

## 制御系構文

```
1  if COND      : BODY
2  [elif COND:  BODY]
3  [else :      BODY]
4
5  for ID in RANGE : BODY
6
7  while COND : BODY
8  [else: BODY]
9
10 # 例外処理=エラーイベント処理
11 try: BODY
12 [except: HANDLER]
```

- while, for は C と同様に break, continue を持つ
- 例外処理に関しては次回説明するかも
- λ 計算をやっていないのでリスト内包表記，ラムダ式は省略
- if `__name__ == '__main__':` への言及は省略

## sample2.py

```
1  #!/usr/bin/env python          # 必須：インタプリタ指定
2  #-*- coding: utf-8 -*-        # 文字コード指定
3
4  def countN (n, k):              # nの約数を数える下請け関数
5      if k == 1:                  # 2までを調べる
6          return 0
7      else:
8          if n % k == 0:
9              return 1 + countN(n, k - 1)
10         else:
11             return countN(n, k - 1)
12
13 def cat(x):                      # category of 'x'
14     n = countN(x, x - 1)
15     if n == 0: return "sosu"
16     elif n == 2: return "tasu"  # elif == else if
17     else : return -1
```

```
1  # 色々 (リスト内包表記, ラムダ式, 辞書型, メソッド) 使っていいなら1行:
2  def cat(x): return {0:"sosu", 2:"tasu"}.get(len([1 for n in range(2,x-1) if x % n == 0]),-1)
```

# OOPL in Python

sample3.py

## クラス C1 の定義

```
1  #!/usr/bin/env python
2  #-*- coding: utf-8 -*-
3
4  class C1:
5      # __init__は常にコンストラクタを表す
6      def __init__(self):
7          self.x = 1 # インスタンス変数x
8          self.y = 2 # インスタンス変数y
9      # メソッド
10     def result(self):
11         return self.x + self.y
12
13 c1 = C1() # コンストラクタは無引数?
14 print(c1.x)
15 print(c1.result())
16 # 前半終了
```


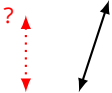

## C1 のサブクラスである C2 の定義

```
17 class C2(C1): # C2はC1に含まれる
18     def __init__(self, a):
19         self.x = 3
20         self.y = 4
21         self.z = a # インスタンス変数z
22     # def result(self): return self
23         .x * self.y
24     def result2(self):
25         return (self.x+self.y) *
26             self.z
27
28 c2 = C2(10) # コンストラクタは1引数
29 print(c2.x)
30 print(c2.z)
31 print(c2.result()) # no error
32 print(c2.result2())
```

7,8 行目でインスタンス変数 x, y に代入しているが、定義がない。

Python ではローカル変数、インスタンス変数は定義する必要がない。最初の代入文で自動的に定義される。

## 仮引数と実引数の対応

言語	C	C++	Python
定義（宣言）	<code>int f(int x, int y)</code>	<code>int f(int x)</code>	<code>def f(self, x)</code>
関数適用	<code>f(x1, 2);</code> 	<code>s.f(3)</code> 	<code>s.f(4)</code> 

- C では、 $n$  引数関数には  $n$  個の実引数
- C++ では、 $n$  引数関数には  $n$  個の実引数．先頭の  $s$  は  $f$  の選択に貢献．
- Python では、 $n$  引数関数には  $n - 1$  個の実引数．第 1 実引数は括弧の外（関数名の左）に出ている（正確には  $f$  は関数ではなくメソッドなので，そういうもの）



## Python パッケージ

これがあるから Python を使うという人がいるようなもの

- numpy – 数値計算パッケージ
- scipy – 科学技術計算パッケージ
- matplotlib – グラフ描画パッケージ
- TensorFlow – Google 製深層学習ライブラリ
- Chainer – Preferred Networks 製深層学習 (+GPU 計算)
- requests – http レイヤ通信パッケージ (演習室で動く @\_@)
- **pip – パッケージマネジャー**
- jupyter – jupyternotebook, jupyterlab

# パッケージの使用例

[http://weather.livedoor.com/weather\\_hacks/webservice](http://weather.livedoor.com/weather_hacks/webservice)

getWeather.py (コマンドラインで permission を設定すること)

```
1  #!/usr/bin/env python
2  #encoding:utf-8
3  import requests as r          # パッケージ読み込み
4  import json as j             # パッケージ読み込み
5
6  url = 'http://weather.livedoor.com/forecast/webservice/'
7  loc = '420010'               # 地域番号
8  res = r.get(url+'json/v1?city='+loc).text
9
10 # 読み込んだJSONフォーマットの文字列を辞書型に変換
11 d = j.loads(res) # パッケージ.関数 ~ オブジェクト.関数
12 print(d['title'])           # いろいろ表示
13 for f in d['forecasts']:    # f(orecast)
14     print (f['dateLabel']+'('+f['date']+') '+f['telop'])
```

- <https://repl.it/> は最近インターネット接続のブロックをやめたので実行できるようになった
- 地域番号一覧: [http://weather.livedoor.com/forecast/rss/primary\\_area.xml](http://weather.livedoor.com/forecast/rss/primary_area.xml)
- これは Python のプログラムだが pipe を使えば C からこのプログラムの出力を利用することができる

pipe()を使えばC言語  
のプログラムからPython  
のプログラムの出力を利用  
することができる。つ  
まりCのプログラムが現  
在の天気データを基に描  
画内容を変えることはと  
ても簡単なことである。

# 1. 可変長引数関数

C

```
1  printf("\n");
2
3  char *x = ...;
4  printf(x);
5
6  printf(x, y);
7
8  printf(x, y, z);
9
10 printf(x, y, z, g);
```

静的に正しい型を付けることは困難

可変長引数関数を除けば C の関数は非常に単純

## 2. Default Arguments, デフォルト引数

### C++

```
1 int f (int x, int y) {  
    ...  
}
```

```
1 f(1, 3);    // x = 1, y = 3  
2 f(1);       // error  
3 f();        // error
```

```
1 int f (int x, int y=0) {  
    ...  
}
```

```
1 f(1, 3);    // x = 1, y = 3  
2 f(1);       // x = 1, y = 0  
3 f();        // error
```

常に右寄せ

```
1 int f (int x=1, int y=2)  
    ...  
}
```

```
1 f(2, 3);    // x = 2, y = 3  
2 f(2);       // x = 2, y = 2  
3 f();        // x = 1, y = 2
```

利点：

### 3. Keyword Arguments, キーワード引数

Python ▶ 4.7.2@3.6.1

#### Python

```
1 def foo (a): return (a + 1)
2 # foo (3)
3 def foo (a, b): return (a + b)
4 # foo(3, 4)
5 def foo(a, b = 1): return (a + b)
6 # foo(3,b=2)
7 def foo(a = 10, b = 1): return (a + b)
8 # foo(b=3, a=1), foo()
```

#### C

```
1 foo(1, x = 10);
2 .....
3 x = 10;
4 foo(1, x);    // このxは実引数
```

利点：

# プログラム言語論専門用語

14

arity

▶ wikipedia

関数の**仮**引数の個数: nullary, unary, binary, ternary,  $n$ -ary

variable arity function, 可変長引数

default arguments, デフォルト引数

keyword arguments, キーワード引数