

引き継ぎ資料 Vol.3

CにないPythonの世界

2016/07/??

コンセプト

Cには無いPythonの世界を堪能しよう!

目次

1. タプル・リスト・ディクショナリ
2. 関数
3. オブジェクト
4. 名前空間
5. numpy・matplotlib

目次

1. タプル・リスト・ディクショナリ

2. 関数

3. オブジェクト

4. 名前空間

5. numpy・matplotlib

Cの言語仕様にはないデータ構造

- ・ タプル
- ・ リスト
- ・ ディクショナリ

タプルとは

数が並んだもの

Cで近い機能は配列

タプル

簡単な例

```
>>> t = (1, 2, 3)
>>> print(t)
(1, 2, 3)
>>> print(t[0])
1
```

ここまではCの配列と同じ

タプル

あえてCで書くなら...

```
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char** argv) {
    int t[] = { 1, 2, 3 };
    printf("%d\n", t[0]);
    return 0;
}
```

簡単!

難しい例

```
>>> t = 1, 2, 3
>>> x, y, z = t
>>> def hoge():
...     return 4, 5, 6
...
>>> a1, a2, a3 = hoge()
>>> a = hoge()
>>> u, v, w = z, y, x
```

それぞれの変数の中身は？

```
>>> t = 1, 2, 3
>>> x, y, z = t
>>> def hoge():
...     return 4, 5, 6
...
>>> a1, a2, a3 = hoge()
>>> a = hoge()
>>> u, v, w = z, y, x
```

```
>>> print(t)
(1, 2, 3)
>>> print(x, y, z)
1 2 3
>>> print(a1, a2, a3)
4 5 6
>>> print(a)
(4, 5, 6)
>>> print(u, v, w)
3 2 1
```

タプルの要点

- ・ タプルに必要なのは“,”(カンマ)
- ・ 複数の値を返す関数はタプルを一つ返す関数
- ・ タプルは自動的に展開され複数の変数に代入
- ・ **タプルでは要素の変更は不可**

タプルでは要素の変更は認められない

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

```
>>> t[1] = 4
```

Traceback (most recent call last):

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

TypeError: 'tuple' object does not support item assignment

リストとは

タプルに近い数の並んだもの
後から変更が可能

リスト

簡単な例

```
>>> t = []  
>>> t.append(1)  
>>> t.append(2)  
>>> print(t)  
[1, 2]  
>>> t[1] = 3  
>>> print(t)  
[1, 3]
```

ディクショナリとは

添字に数字以外が使えるリスト

言語によっては連想配列・Map・HashMapなど

目次

1. タプル・リスト・ディクショナリ

2. 関数

3. オブジェクト

4. 名前空間

5. numpy・matplotlib

目次

1. タプル・リスト・ディクショナリ

2. 関数

3. オブジェクト

4. 名前空間

5. numpy・matplotlib

目次

1. タプル・リスト・ディクショナリ

2. 関数

3. オブジェクト

4. 名前空間

5. numpy・matplotlib

目次

1. タプル・リスト・ディクショナリ

2. 関数

3. オブジェクト

4. 名前空間

5. numpy・matplotlib