みんなのPython勉強会 #21 2017.2.3

Python3の進化について

辻 真吾

@tsjshg

自己紹介

- 1975年生まれ
- 都内のとある大学で研究やってることになってます
  - ただ、やっていることはPythonで主に機械学習を使ったデータ解析
  - というわけで、大学辞めて会社でも作ろうかと思ってます
  - お仕事手伝ってくれる方、募集してます
- http://www.tsjshg.info/

AIを使ってヒトが賢くなる

A Go player increased their global ranking by 300 places by playing Google DeepMind's computer

Sam Shead, Business Insider UK

May 13, 2016, 5:11 AM 1,788

FACEBOOK LINKEDIN TWITTER EMAIL PRINT

Fan Hui makes a living by playing Go, a Chinese board game that dates back over 2,500 years with more moves possible than there are atoms in the universe.

The objective of Go is simple: to surround the other player's stones with your own, forcing them to sacrifice their piece.

Fan is pretty good, having been crowned European champion every year for the last four years.

Professional Go player Fan Hui. Google

http://www.businessinsider.com/fan-hui-practices-go-in-deepminds-office-2016-5

今日はPython3のお話

その前に

- 「Python2と3はどっちがいいんですか？」
- Python3をおすすめしたい！
- が…
- yumなどLinuxのシステムツールにPython2のもの
- Google Cloud Platform SDKが2.7を要求

2と3の違い

（いろいろありますが・・・）

文字列

	Python2	Python3
`len('abc')`	3	3
`len('あいう')`	9	3
`len(u'あいう')`	3	3 (3.3から記法が復活)

遅延評価

Python2

```
>>> range(10)
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

Python3

```
In [26]: range(10)
Out[26]: range(0, 10)
```

必要になったときに具体的な値を返すことで、メモリ利用の効率化などを実現

ま、2は忘れて3へ

これまでの進化が一望できる

Previous topic
Python Documentation contents

Next topic
What's New In Python 3.6

This Page
Report a Bug
Show Source

What's New in Python

The "What's New in Python" series of essays takes tours through the most important changes between major Python versions. They are a "must read" for anyone wishing to stay up-to-date after a new release.

OrderedDict

- 3.1で導入された順番を保持するdict

```
from collections import OrderedDict
```

```
od = OrderedDict()
od['a'] = 1
od['b'] = 2
od['c'] = 3
```

```
od
```

```
OrderedDict([('a', 1), ('b', 2), ('c', 3)])
```

OrderedDictの進化

- 3.5で実装がCになる
  - 4〜100倍の高速化
- 3.6では、組み込み型のdictの実装が変更
  - 20〜25%のメモリ使用量の削減
  - 順番を保持するようになるも、これは副作用的なものなので、頼らないで（変更の可能性あり）

argparseの導入

- 3.2で追加
  - --helpでヘルプを自動生成など多機能
- 元々、よく使われていた外部パッケージが取り込まれた
- 同じ機能のoptparseはdeprecated（廃止予定）に

```
import argparse

parser = argparse.ArgumentParser(description='和の計算')

parser.add_argument('integers', metavar='N', type=int, nargs='+', help='和を計算する整数を入力してください。')
parser.add_argument('--sum', dest='accumulate', action='store_const', const=sum, default=max, help='和を計算（指定が無いときは最大値）')

args = parser.parse_args()

print(args.accumulate(args.integers))
```

python argparse_test.py 1 2 3 4 5 6 7 8 9 10 --sum → 55

https://docs.python.org/3/library/argparse.html

周辺環境の整備

- 3.3でvenvモジュールが追加
  - Pythonの仮想環境が作れる
    - これに関しては、condaも便利です
- 3.4でpipが標準搭載に
  - これに関しても、condaが便利

Python3の未来を占う

- range, map, filterなどがリストをそのまま返さなくなった
- 3になった時の大きな変化

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

```
<filter at 0x110b8d3c8>
```

```
list(evens)
```

```
[2, 4]
```

非同期処理

- 3.4でasyncioパッケージが導入
  - この時点ではprovisional（まだまだ準備中）
  - 3.6でstableに
- 重い処理があるときに後回しにできる仕組み
  - Webサーバなどで、限られたリソースで大量の処理をさばけるようになる

並列処理

- 2.6で追加されたmultiprocessingモジュールを使いやすくするconcurrent.futuresが3.2で導入された

```
import time

def i_sleep
    time.sle

s = time.ti
i_sleep()
i_sleep()
i_sleep()
print('{:0.2
```

```
import concurrent.futures

s = time.time()
with concurrent.futures.ProcessPoolExecutor() as executor:
    executor.submit(i_sleep)
    executor.submit(i_sleep)
    executor.submit(i_sleep)
print('{:0.2f}'.format(time.time() -s ))
```

3.01

1.05

型ヒント

- 3.5からtypingモジュール、まだprovisional

```
def greeting(name: str) -> str:
    return 'Hello {}'.format(name)
```

```
greeting('みんな')
```

```
'Hello みんな'
```

でも、怒られることはない！

```
greeting(2)
```

```
'Hello 2'
```

PEP（Python Enhancement Proposal）484

> It should also be emphasized that **Python will remain a dynamically typed language, and the authors have no desire to ever make type hints mandatory, even by convention.**

まとめ

- Python2.7のサポートは、2020年まで
  - Python3に移行しましょう！
- Python3は非同期処理や型ヒントなど、大規模なシステムでも使える堅牢性を意識している（？）
- まだまだ進化は続くと思うので、これからも楽しみ