第 4 回 Python 講座

2018 年 7 月 27 日

1　前回の回答

2

```
import random
s=int(input(”数字を入力”))
alpha=”abcdefghijklmnopqrstuvwxyz”
for l in range(s):
  i = random.randint(0,len(alpha)-1)
  print(alpha[i])
```

4

```
import re
mozi=”adbcacbdexxyyzzabchixyzopqssabexypzxyz”
m1=re.finditer(r”abc+”,mozi)
for match1 in m1:
  print(match1.span(),”abc”)
m2=re.finditer(r”xyz+”,mozi)
for match2 in m2:
  print(match2.span(),”xyz”)
```

2　モジュール

pyhton において機能ごとに分割してスクリプトを作成したい時があります。その場合にそれらのファイル (.py ファイル) をモジュールと呼びます。モジュールにある関数やクラスはモジュールを import することで使用することができます。python には標準モジュールが用意されています。前回の演習時に自分が用いたのも標準モジュールの一つの random になります。下によく使うモジュールや機械学習に用いるモジュール, ライブラリを紹介しますこの中のいくつかは標準モジュールではなく追加でインストールしなくてはなりませんが以上にあげたものはすべてインストールしても損することはないかと思います。特に numpy は機械学習だけでなく python を使う上で非常に便利な計算ができるようになるためこれだけは絶対に入れたほうが良いかと思います。またプログラム内でモジュール名はすきに変更することができます。その場合 import モジュー

ル名 as 変更したい名前、とプログラミングすることでできます。numpy なんかはほぼすべての参考書、web サイトで np と略されていますし長いモジュール名はよく略されています。

モジュール名	解説
math	数学関数を使用できる
random	乱数発生ができる
datetime	時間を調べられる
csv	csv ファイルを操作できる
Path	ファイルパスを操作できる
glob	ファイル名を取得できる
numpy	numpy 変数及びリストを使用できる
Image	画像の情報を入手できる
matplotlib	グラフ作成ができる
re	正規表現を扱うことができる
os	コマンドラインのコマンドを使用できる
scipy	科学技術計算用のライブラリ
chainer	日本製の機械学習ライブラリ
tensorflow	Google 社製の機械学習ライブラリ
scikitlearn	アンサンブル学習などを行うライブラリ

3　クラス

python はオブジェクト指向言語でもあります。そのためクラスが用意されています。機械学習を行う際には主に学習用のクラスを用意してそのクラスをモデル化して学習を行います。なのでクラスの管理および理解が重要になってきます。しかしオブジェクト指向言語がよくわからない、クラス、継承、それ何？って人も中にはいると思います。python は Java や C++ よりもオブジェクト指向を気にしなくてもよろしいですが最低限の知識を持っておいたほうがいいでしょう。理屈が理解できない人はそういうものだと思っておくと後で理解が深まるかもしれません。今回の講座では詳しいオブジェクト指向の説明はしません。python では他の言語とどう違うのかを中心に説明します。

3.1　データ属性

C++ ではデータメンバ、Java ではフィールドにあたるものがデータ属性です。正確に言えば違うのですがそう思ってくださって支障ありません。python ではローカル変数同様洗顔する必要がありません。ですのでいきなり使うことができます。データ属性は self. 名前で使用することができます。また__init__メソッドはメソッドが定義されているクラスのインスタンス生成時に呼び出されます。そのほかは__call__メソッドが主で call は関数のように呼び出したら呼び出されるメソッドになります。

3.2　クラス変数

python にもクラス変数があります。データ属性 (インスタンス変数) がインスタンス固有のデータであるのに対して、クラス全体で共有するデータがクラス変数です。

3.3　プライベートアクセス

Java などの場合は private,public などのアクセス修飾子がありアクセスを制限していましたが python にはありません. つまり python では基本的に外部からのアクセスが可能なのです. しかし慣例として_変数名とすることで public ではないとすることができますでは簡単なオブジェクト指向のプログラムを見てみましょう. 次のプログラムは買い物時の計算プログラムをクラスを用いてプログラムしたものです

オブジェクト指向

```
import math
class Calcfee:
    def __init__(self):
        self.shopping_fee=1000
        self.tax_rate=0.08
        self.value=0
    def addItem(self,price):
        self.value+=price
    def calc(self):
        total=self.value+self.shopping_fee
        tax=math.ceil(total*self.tax_rate)
        v=math.ceil(total+tax)
        return v
fee1=Calcfee()
fee1.addItem(500)
print(fee1.calc(),"円")
fee2=Calcfee()
fee2.shopping_fee=1500
fee2.addItem(800)
fee2.addItem(500)
print(fee2.calc(),"円")
```

4　機械学習に向けて

なぜ今までオブジェクト指向の説明を簡単にしたかというと機械学習で用いる様々なライブラリはすべてオブジェクトで生成されており、深い知識は無くとも少しは理解していなければライブラリを用いた機械学習が行えないからです. ならライブラリを用いなければいいという考えを持つかもしれませんがライブラリは絶対

使ったほうが良いです.s それはコードの長さや設定の数が明らかに違うからです. 自分が chainer を用いているため下記のプログラムは chainer を用いたものになりますが実際に機械学習のプログラムは下記のようなものになります

機械学習

```
class LSTM(Chain):
    def __init__(self,in_size,hidden_size,out_size):
        super(LSTM, self).__init__(
            xh = L.Linear(in_size, 100),
            hh = L.LSTM(100, hidden_size),
            hh2 = L.Linear(hidden_size,hidden_size),
            hh3 = L.Linear(hidden_size,100),
            hy = L.Linear(100, out_size)
        )
    def __call__(self, x, t=None, train=False):
        x = Variable(x)
        if train:
            t = Variable(t)
        h = self.xh(x)
        h = self.hh(h)
        h = F.relu(self.hh2(h))
        h = F.relu(self.hh3(h))
        y = self.hy(h)
        if train:
            return F.mean_squared_error(y, t)
        else:
            return y.data
    def reset(self):
        self.zerograds()
        self.hh.reset_state()
```

これは機械学習の設定を行ったものになります. これでも長いかと思われがちですが, なんのライブラリも用いずにこの class を作ろうとするならばおそらく 200 から 300 行程度必要になるかと思います. また何か機械学習について調べようとするとだいたい何らかのライブラリを用いてコードをプログラムしているはずなのでやはり最低限のオブジェクト指向の知識は必要かと思います.1 年生は今は焦らずとも 2 年生でオブジェクト指向言語の講義があるためしっかりとやるのはそこでいいと思いますが, 今のうちにやっておくといいかもしれません.2 年生以降はこの講座の最低限の知識だけは持ってから機械学習を行うことをお勧めします.(自分は講義のときあまり理解せずに研究やってるときに理解したため大変でした)

自分に余裕があれば機械学習講座も開催しますが学会等ありますのでもしかしたらできないかもしれません.

4.1 numpy

機械学習を行う時ほぼ確実に必要になるのが numpy です.numpy は anaconda3 をインストールした人は最初から導入されていますが,python だけをインストールした場合は pip コマンド等でインストールしなければ使えません.numpy とは python で行列計算等を行うのに便利なモジュールでどのパッケージでも numpy が求められます.i 次元はもちろん多次元も簡単に実装できます. また numpy 上で使える様々な関数が使用可能になります.numpy 配列は pyhton のリストにさらに拡張機能を増やしたものと考えるとわかりやすいでしょう. ファイル操作も savetxt や loadtxt で簡単に行えるようになりますし配列の分割や追加もより直感的に行えるようになります. 機械学習を行う際はまずこれをうまく扱えるようになるところから始まります.

4.2 Chainer

機械学習を行う際, パッケージは主に TensorfFlow と Chainer のどちらかを使うことになるでしょう. そのうちの Chainer について軽く触れていきます.Chanier は日本製の機械学習パッケージです.Chainer は非常に発展が速く, どんどんアップデートされていきます.2015 年の 6 月にリリース開始され,20171 年 6 月に v2.00,2017 年の 12 月に安定版の chaienrv3 のアップデートがあったと思えば,2018 年の 4 月には chainerv4.0.0 がリリースされ,7 月には chainerv4.2.0 がリリースされています. もうすぐ chaine4v4.4.0 がリリースされる噂もあるぐらいです. ですので chainer を扱う場合はバージョン管理に気を付けなければなりません.

chainer 一番の特徴は「複雑なネットワークに対するプログラムが簡単に組める」ことと言えるでしょう. 枠組みさえ理解してしまえば独自のネットワークを構築するには一番かと思います. 機械学習では基本のプログラムは同じものが多いですが計算部分は独自で組まなければデータセットに対していい学習ができるとは言えません. そういった点では chianer はおすすめできます.

機械学習と一言で言っても画像認識, 自然言語処理, ビックデータ解析, 最適化など様々なものがあります. そのすべてをサポートしてくれるフレームワークをもつのはあまりありません, ですのでパッケージを選ぶ際には注意したほうがよいでしょう.