Programovanie v jazyku Python

Vedecké výpočty v Pythone
prednáška 10

Katedra kybernetiky a umelej inteligencie
Technická univerzita v Košiciach
Ing. Ján Magyar, PhD.

Pole (Array)

- (viacrozmerné) usporiadanie hodnôt rovnakého typu
- k údajom pristupujeme pomocou indexov - na základe ukladania v pamäti
- v Pythone dve možnosti:
  - zoznam zoznamov (zoznamov zoznamov...)
  - pomocou knižnice `numpy`
    - C-čková implementácia polí
    - vyšší výkon a rýchlejšie výpočty ako pomocou zoznamov

Numpy

- implementácia polí v Pythone (v skutočnosti sú to matice)
- podobné, ako zoznamy, iba nemeniteľné (hodnoty však vieme aktualizovať)
- vytvorenie polí:

```
import numpy as np
my_array = np.array([2, 3, 4])
```

- konvertovanie vstupu na pole:

```
numpy.asarray(a, dtype=None, order=None)
```

Typ polí

- každé pole má typ `ndarray`, jednotlivé prvky musia byť rovnakého typu `my_array.dtype`

- typy prvkov nie sú pythonovské primitívne typy, sú to vlastné implementácie knižnice `numpy`

Tvar polí

- celkový počet prvkov: `my_array.size`
- každé pole má jednu alebo viac dimenzií
- tvar poľa je definovaný rozmermi
  `my_array.shape`

- vracia n-ticu s rozmermi, napr.: `(2, 3)`, kde prvá hodnota je vonkajší rozmer, a postupujeme smerom dnu
- ak je pole dvojdimenzionálne, prvý rozmer je počet riadkov, druhý je počet stĺpcov*

Úprava rozmerov poľa

- `numpy.reshape(array, newshape, order='C')`
  - upraví tvar poľa na požadované rozmery
  - počet prvkov v poli a počet prvkov v novom tvare musí byť rovnaký
  - order určí poradie pridávania prvkov (`C` alebo `F`)
  - vracia nové pole
  - -1 na automatické výpočty rozmeru

- `numpy.flatten(order='C')`
  - vracia nové pole - vektorová reprezentácia (jeden riadok, resp. stĺpec)

zeros(), ones() a full()

- metódy slúžia na inicializáciu matice s hodnotami 0, 1 alebo vlastnou hodnotou

- `numpy.zeros(shape, dtype=float, order='C')`
- `numpy.ones(shape, dtype=float, order='C')`
- `numpy.full(shape, fill_value, dtype=None, order='C')`
- `numpy.empty(shape, dtype=float, order='C, *, like=None)`

Generovanie rozsahov

- `numpy.arange(start, stop, step)`
  - funguje rovnako ako `range()`
  - možnosť práce s floatmi – skôr použiť `linspace`
- `numpy.linspace(start, stop, num=50, endpoint=True)`
  - vygeneruje `num` hodnôt v rovnakej vzdialenosti od seba v intervale `[start, stop]`
  - `endpoint` určí, či interval má obsahovať hodnotu `stop`

Pridávanie prvkov

- implementuje konkatenáciu polí

- `numpy.hstack(tup)`
  `numpy.hstack((array1, array2))`
- `numpy.vstack(tup)`
  `numpy.vstack((array1, array2))`

- pre vytvorenie polí dynamicky je lepšie použiť zoznamy, a následne vygenerovať pole

Indexovanie a slicing v numpy

- zásady sú rovnaké ako v Pythone, numpy ale umožňuje skrátený zápis

```
np.array([(1,2,3), (4,5,6)])
print('First row:', e[0])
print('Second column:', e[:,1])
print('Second row, first two values:', e[1, :2])
```

Indexovanie pomocou polí

- ako index vieme použiť aj druhé pole – načítajú sa hodnoty pod indexom podľa druhého poľa
- druhé pole môže mať rôzne tvary – určí tvar výsledku
- vieme použiť aj viac polí, kde hodnoty udávajú index po jednotlivé dimenzie

Podmienené indexovanie

- výsledky podmienok vieme použiť aj na podmienené indexovanie, teda na generovanie masiek

```
a = np.arange(12).reshape(3, 4)
b = a > 4
a[b]
a[b] = 0
```

vyberie hodnoty väčšie ako 4

vynulujeme dané čísla

Aritmetické operácie nad maticami

- primitívne operácie fungujú element-wise

```
test = np.array([1, 2, 3])
test += 1

test [2, 3, 4]
```

- ak je potrebné pretypovanie, tak sa to robí na všeobecnejší typ

Štatistické metódy

- `np.min(array) / np.argmin(array)`
- `np.max(array) / np.argmax(array)`
- `np.mean(array)`
- `np.median(array)`
- `np.std(array)`

- všetky podporujú parameter `axis`, ktorý určí, po ktorej osi sa má vypočítať štatistická metrika
- numpy ponúka aj matematické funkcie

Maticové operácie

- `np.transpose(array, axes=None)`
- `np.dot(array1, array2)`
- `np.matmul(array1, array2)`
  - ak polia sú dvojrozmerné - štandardné násobenie
  - ak polia majú jeden rozmer - vygeneruje sa z nich dvojrozmerné pole
  - ak polia majú viac rozmerov, ako dva - považujú sa za zásobník dvojrozmerných matíc
- násobenie matíc je možné aj cez operátor `@` alebo `array1.dot(array2)`

Kópie a viewy

- pomocou metódy `view()` vieme vytvoriť pohľad na pôvodné pole

```
c = a.view()
c is a              False
c.base is a         True
```

- používajte metódy `copy()` pre hlboké kópie

```
d = a.copy()
```

Pandas

- knižnica pre dátovú analytiku
- nadstavba nad `numpy`
- pre tabuľkové údaje a časové rady
- umožňuje všeobecné indexovanie (cez názvy stĺpcov)
- zvyčajne sa používa importom

```
import pandas as pd
```

Základné objekty

- postupnosť prvkov – `Series`

```
s = pd.Series([1, 3, 5, np.nan, 6, 8])
```

- údajový rámec – `DataFrame`

```
my_ids = pd.date_range("20220214", periods=4)

df = pd.DataFrame(
    np.random.randn(4, 4),
    index=my_ids,
    columns=['A', 'B', 'C', ,D'])
```

Generovanie údajových rámcov

- z dictionary

```
df2 = pd.DataFrame({
    "A": 1.0,
    "B": "2022-02-14",
    "C": ["test", "train", "eval", "train"]
})
```

- načítanie zo súboru, napr.

```
df3 = pd.read_csv(path, sep=',')
```

Indexovanie v dataframeoch

- funguje všeobecné Pythonovské indexovanie
- ďalšie pomocné možnosti prístupu: `at`, `iat`, `loc`, `iloc`
- výber stĺpca (vráti postupnosť `Series`)
  `df["A"]`
- výber riadkov
  `df[0:3]`
  podľa indexov
  `df["20220214":"20220216"]`
- výber riadkov a stĺpcov
  `df.loc["20220214":"20220216", ["A", "B"]]`
- `iat` a `iloc` fungujú podobne ale iba s číselnými indexmi

Podmienený výber

- pomocou podmienok (podmieňovacie operátory)
  `df[df["A"] > 0]`

- pre filtrovanie môžeme použiť metódu `isin()`
  `df[df["C"].isin(["test", "train"])]`

Nastavenie hodnôt

- rozšírenie o stĺpec:
  `df["F"] = "new"`

- nastavenie hodnoty podľa (číselného) indexu
  `df.at[0, "A"] = 0 / df.iat[0, "A"] = 0`

- nastavenie niekoľkých hodnôt pomocou polí
  `df.loc[:, "D"] = np.array([5] * len(df))`

- podmienené nastavenie (napr. nastavenie na absolútne hodnoty)
  `df[df < 0] = -df`

Základná práca s dataframeami

- ukážkové riadky zo začiatku/konca dataframeu
  `df.head(n=g) / df.tail(n=5)`
- získanie zoznamu indexov a stĺpcov
  `df.index / df.columns`
- konvertovanie na `numpy` polia (iba pri rovnakých údajových typoch)
  `df.to_numpy()`
- prehľad štatistických metrík
  `df.describe()`
- transpozícia
  `df.T`

Triedenie údajov v dataframeoch

- triedenie indexov
  ```
  df.sort_index(axis=1, ascending=False)
  ```

- triedenie podľa hodnôt
  ```
  df.sort_values(by="B")
  ```

Spracovanie chýbajúcich hodnôt

- vymazanie riadkov s chýbajúcimi hodnotami
  `df.dropna(how="any")`

- doplnenie chýbajúcich hodnôt
  `df.fillna(value=0)`

- získanie prehľadu o prítomnosti chýbajúcich hodnôt
  `pd.isna(df)`

Štatistické metódy

- priemer podľa stĺpcov / riadkov
```
df.mean() / df.mean(1)
```

- aplikácia ľubovoľnej funkcie
```
df.apply(np.cumsum)  # suma
df.apply(lambda x: x.max() - x.min())
```

- početnosť hodnôt
```
df["C"].value_counts()
```

Ďalšia funkcionalita

- konkatenácia – `concat()`
- spojenie ako v SQL – `join()`
- zgrupovanie – `groupby()`
  `df.groupby("A") / df.groupby(["A", "B"])`
- zhustenie / rozbalenie dimenzií
  `df.stack() / df.unstack()`

Zhrnutie

- polia ako abstraktný dátový typ
- `numpy` polia
- `pandas` dataframy
- základné operácie pri práci s dátami