

PyQB

Monga

Iterators and generators



PyQB

Monga

Iterators an generators

Lecture XXI: Tabular data

123

Programming in Python¹

Mattia Monga

Dip. di Informatica Università degli Studi di Milano, Italia mattia.monga@unimi.it

Academic year 2023/24, I semester

¹⊚⊕ © 2023 M. Monga. Creative Commons Attribuzione — Condividi allo stesso modo 4.0 Internazionale. http://creativecommons.org/licenses/by-sa/4.0/deed.it

1

Tabular data

PyQB

Monga

Iterators and

Data are often given/collected as tables: matrices with rows for individual records and columns for the fields of the records. This is especially common in statistics, R has a built-in type for this: the dataframe.

pandas



PyQB Monga

pandas (Python for data analysis) brings the DataFrame type to Python. It is based on numpy.

- Series: a one-dimensional labeled array capable of holding any data type (integers, strings, floating point numbers, Python objects, etc.). The axis labels are collectively referred to as the index.
- DataFrame: a 2-dimensional labeled data structure with columns of potentially different types. You can think of it like a spreadsheet, or a dict of Series objects.

th of it

. 1

Series



Series

Monga

import pandas as pd s = pd.Series(np.random.randn(5), index=["a", "b", "c", "d", "e"])

s is a numpy array of floats, each one has a label.

 $d = \{"b": 1, "a": 0, "c": 2\}$

s = pd.Series(d)

The ordering depends on Python and pandas version... The current ones takes the insertion order, but you can provide explicitly the index.

```
d = \{"b": 1, "a": 0, "c": 2\}
s = pd.Series(d, index=['a', 'b', 'c'])
```

126

vectorized) but also a dict.

A Series is convenient because it is a ndarray (and can be

Dataframes



PyQB

Monga

d = { "one": pd.Series([1.0, 2.0, 3.0], index=["a", "b", "c"]), "two": pd.Series([1.0, 2.0, 3.0, 4.0], index=["a", "b", "c", "d"]),

df = pd.DataFrame(d)

A DataFrame has an index and a columns attribute. There are many ways of creating DataFrames, see the docs.

From csv or spreadsheets



Monga

generators

Monga

A famous example: Fisher's Iris flowers dataset. 150 records, "sepal length", "sepal width", "petal

length","petal width","class"

```
iris = pd.read_csv('iris.csv')
# with a url
iris = pd.read_csv('https://tinyurl.com/iris-data')
```

Two ways of indexing



PyQE

Monga

Iterators and generators

PyQB

Monga

Iterators an

Lecture XXII: More pandas

Zecture 70 tm. More pundus

131

• .loc[] "label based"

• .iloc[] "position based"

For both you can use: a single value, a list of values, a boolean array. Two notable things:

- If you use a slice notation with .loc ('a':'f') the last value is included! (different from plain python and from .iloc)
- ② Can be also a callable function with one argument (the calling Series or DataFrame) and that returns valid output for indexing (one of the above)

130

PyQB

Monga

Iterators and generators Iterators



PyQB

Iterators and

Monga

Object can be iterable. Python defines the iterator protocol as:

- iterator.__iter__() Return the iterator object itself. This is required to allow both containers and iterators to be used with the for and in statements.
- iterator.__next__() Return the next item from the container. If there are no further items, raise the StopIteration exception.

Group by

Data can be grouped with groupby, then *summary* function (sum, mean, ...) can be applied to **each** group at the same time.

iris = pd.read_csv('https://tinyurl.com/iris-data')

iris.groupby('variety').mean()

Groups are special **lazy** types which generate data only when needed for the summary operation.

13

Notable iterators



PyQB

Monga

Iterators and generators

Built-in lists, tuples, ranges, sets, dicts are iterators.

- Numpy arrays
- Pandas Series and DataFrames

134

Generators



PyQE

Monga

Iterators and

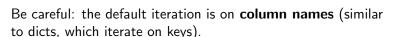
```
def mygenerator() -> int:
    for i in [1, 6, 70, 2]:
        yield i
    print('Ended') # Just to see when it reaches this
        -> point

g = mygenerator()

print(g) # not useful
print(next(g))
print(next(g))
print(next(g))
print(next(g))
print(next(g))
print(next(g))
```

135

Pandas DataFrame



- iterrows(): Iterate over the rows of a DataFrame as (index, Series) pairs. This converts the rows to Series objects, which can change the dtypes and has some performance implications.
- itertuples(): Iterate over the rows of a DataFrame as namedtuples of the values. This is a lot faster than iterrows(), and is in most cases preferable to use to iterate over the values of a DataFrame.

Iterating is **slow**: whenever possibile try to use vectorized operation or **function application**.



PyQB

Monga

Iterators and generators

Pandas function application



., , , , _

Monga

Iterators and generators

```
# apply the function to each column
df.apply(lambda col: col.mean() + 3)
# apply the function to each row
df.apply(lambda row: row + 3, axis=1)
```

Pandas query



```
PyQB
```

Monga

Iterators and generators

```
df[df['A A'] > 3]
# equivalent to this (backticks because of the space)
df.query('`A A` > 3')
# query can also refer to the index
df.query('index >= 15')
# same as
df[15:]
```

138