05 Dataset & Dataframe

Telung Pan
telung@mac.com

Datascientist

介紹新工具

- Python IDLE
- repl.it
- Jupyter Notebook
  - conda
  - miniconda
  - brew cask install miniconda

Jupyter Notebook

- 網址 jupyter.org
- pip install - - upgrade pip
  pip install notebook
  pip install - - upgrade ipython jupyter
- conda install -c Conda-forge notebook

return 語法 I

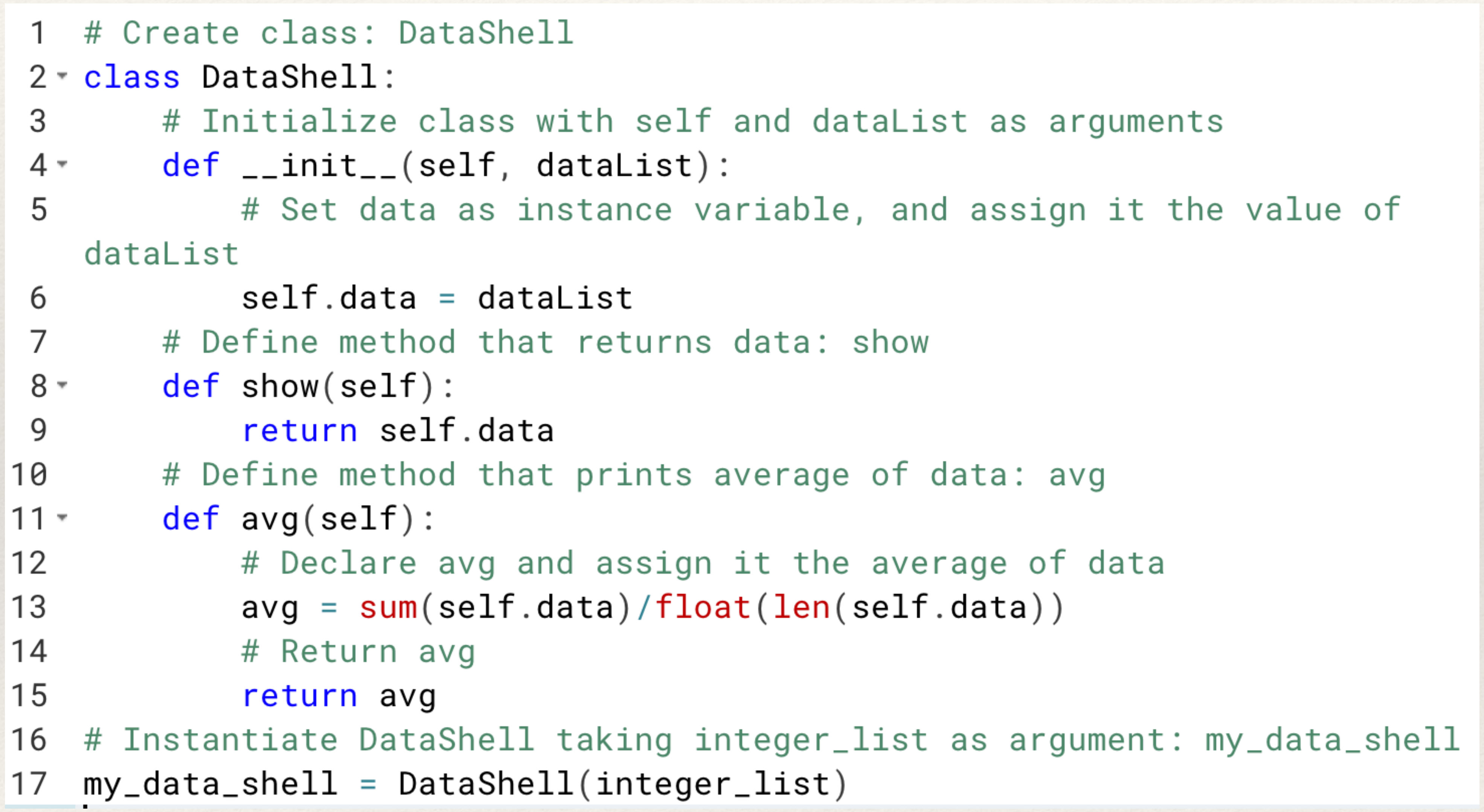
```
class DataShell:
    def __init__(self, x):
        return x
```

return int(self.value)

- 上面的語法我們以 return x 取代熟悉的 print()
- print() 直接輸出字串到 console，而 return 會離開所在的 function 或 method 然後將回傳值傳給呼叫他的人。

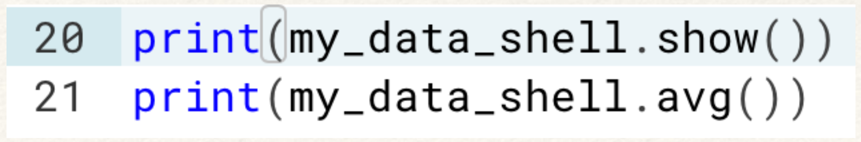
return 語法 II

- 先建立一個 DataShell 類別



```python
# Create class: DataShell
class DataShell:
    # Initialize class with self and dataList as arguments
    def __init__(self, dataList):
        # Set data as instance variable, and assign it the value of dataList
        self.data = dataList
    # Define method that returns data: show
    def show(self):
        return self.data
    # Define method that prints average of data: avg
    def avg(self):
        # Declare avg and assign it the average of data
        avg = sum(self.data)/float(len(self.data))
        # Return avg
        return avg
# Instantiate DataShell taking integer_list as argument: my_data_shell
my_data_shell = DataShell(integer_list)
```

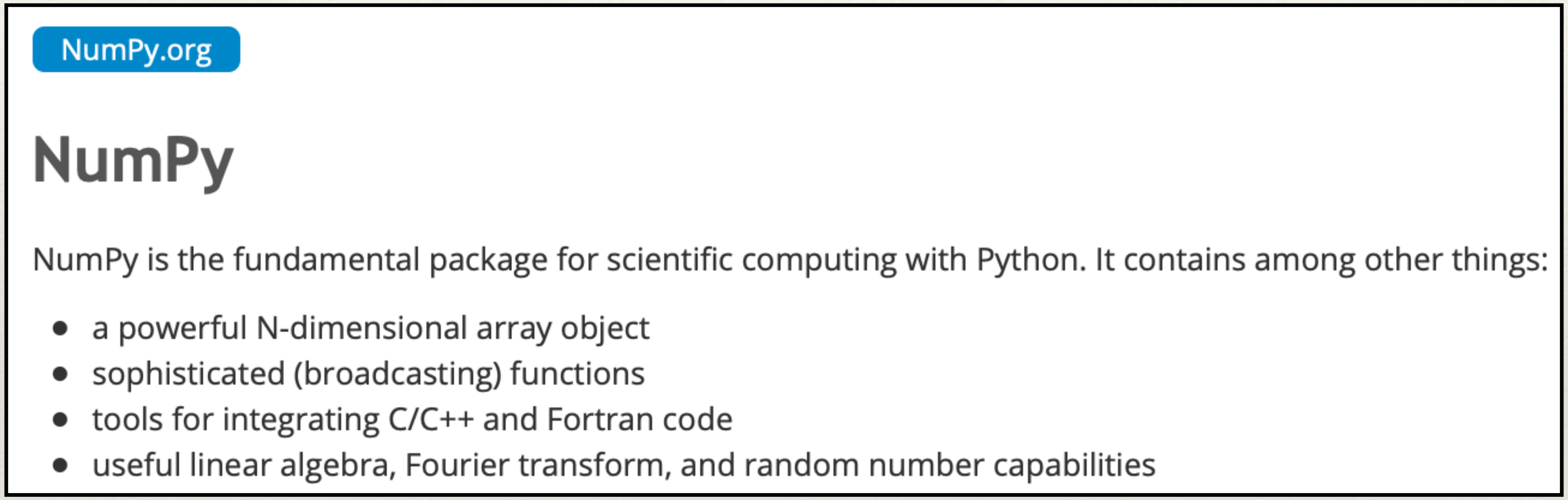
- 呼叫 show 和 avg 方法，列印出來。



```python
print(my_data_shell.show())
print(my_data_shell.avg())
```

return 語法 III: 擴充更加強大的 DataShell 類別

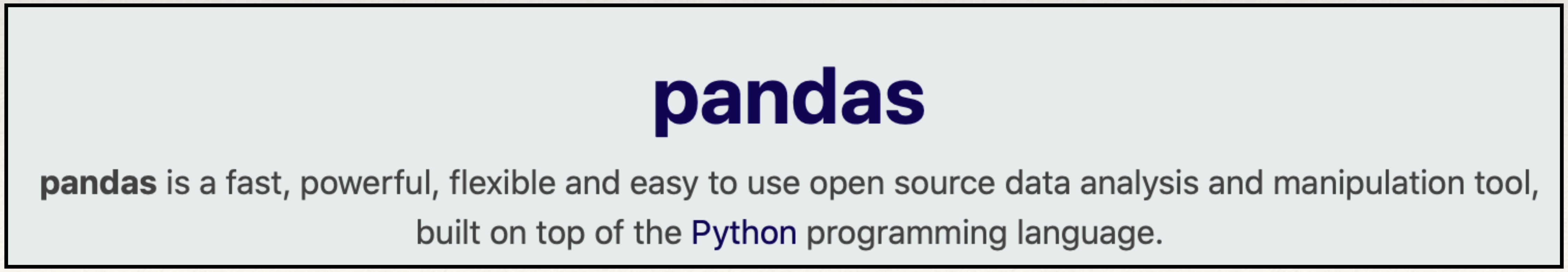
- 匯入 (import) bumpy 和 pandas 套件 (packages)


NumPy.org

NumPy

NumPy is the fundamental package for scientific computing with Python. It contains among other things:

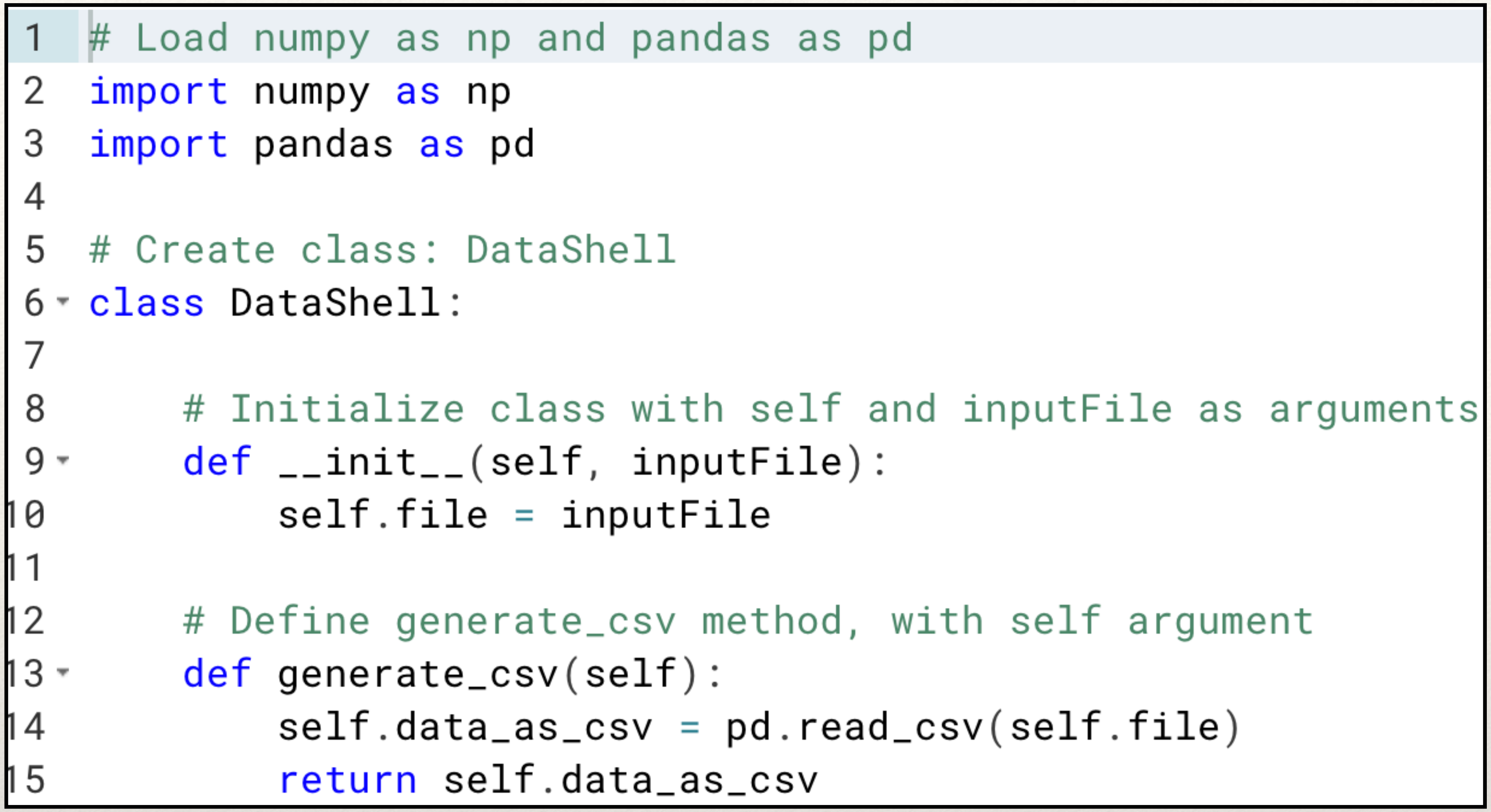
- a powerful N-dimensional array object
- sophisticated (broadcasting) functions
- tools for integrating C/C++ and Fortran code
- useful linear algebra, Fourier transform, and random number capabilities


pandas

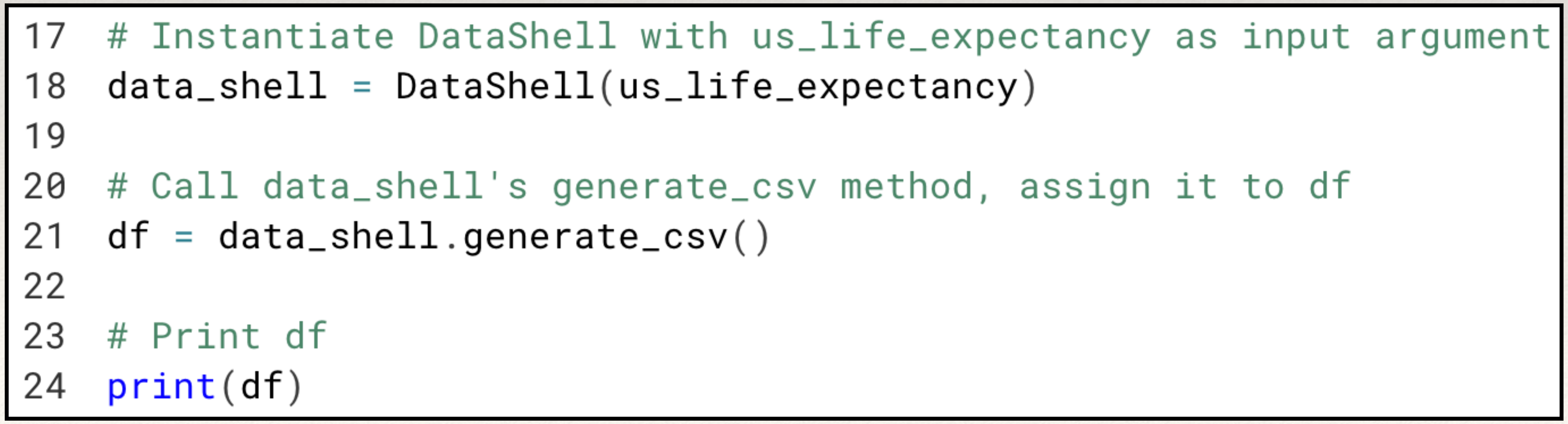
pandas is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool, built on top of the Python programming language.

Panda 官方網站文件

https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.read_table.html



```python
# Load numpy as np and pandas as pd
import numpy as np
import pandas as pd

# Create class: DataShell
class DataShell:

    # Initialize class with self and inputFile as arguments
    def __init__(self, inputFile):
        self.file = inputFile

    # Define generate_csv method, with self argument
    def generate_csv(self):
        self.data_as_csv = pd.read_csv(self.file)
        return self.data_as_csv
```

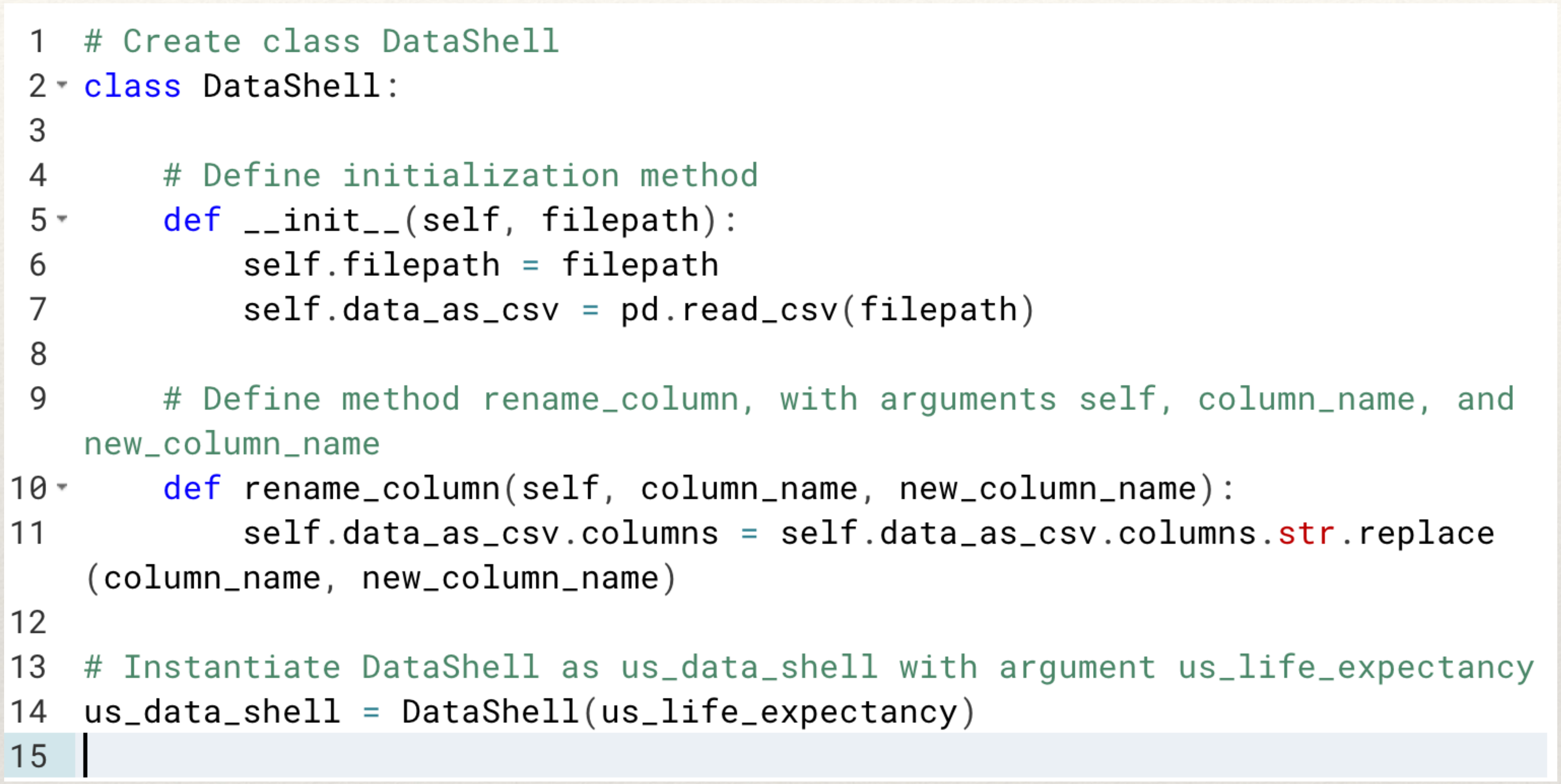


```python
# Instantiate DataShell with us_life_expectancy as input argument
data_shell = DataShell(us_life_expectancy)

# Call data_shell's generate_csv method, assign it to df
df = data_shell.generate_csv()

# Print df
print(df)
```

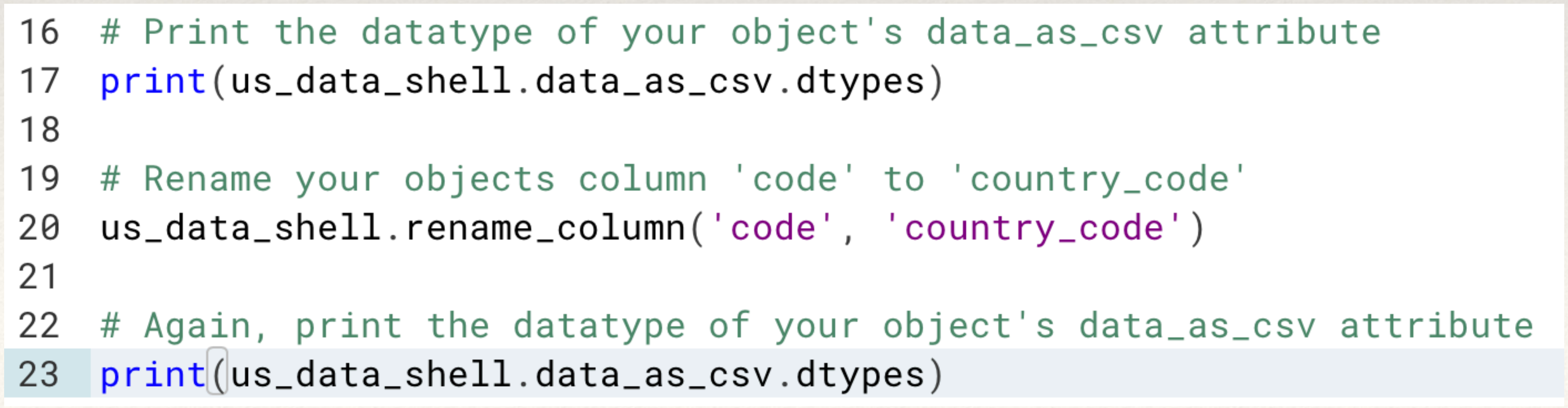
- **Source code**

```
import numpy as np
import pandas as pd

class DataShell:
 def __init__(self, inputFile):
        self.file = inputFile
   def generate_csv(self):
        self.data_as_csv = pd.read_csv(self.file)
        return self.data_as_csv

us_life_expectancy = '/Users/telung/Documents/us_life_expectancy.csv'
data_shell = DataShell(us_life_expectancy)
df = data_shell.generate_csv()

print(df)
```

修改欄位名稱的程式

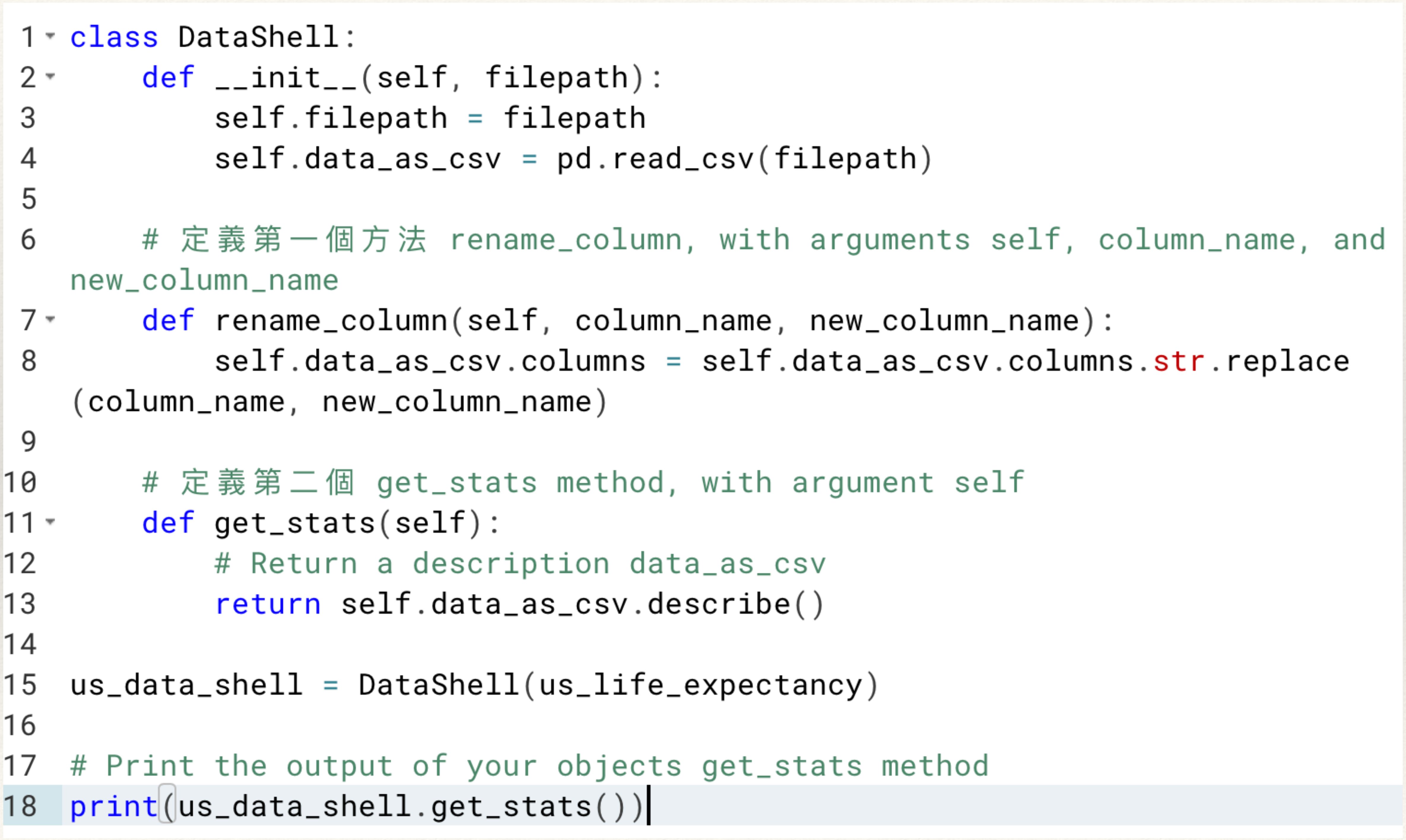


```python
# Create class DataShell
class DataShell:

    # Define initialization method
    def __init__(self, filepath):
        self.filepath = filepath
        self.data_as_csv = pd.read_csv(filepath)

    # Define method rename_column, with arguments self, column_name, and new_column_name
    def rename_column(self, column_name, new_column_name):
        self.data_as_csv.columns = self.data_as_csv.columns.str.replace(column_name, new_column_name)

# Instantiate DataShell as us_data_shell with argument us_life_expectancy
us_data_shell = DataShell(us_life_expectancy)

```



```python
# Print the datatype of your object's data_as_csv attribute
print(us_data_shell.data_as_csv.dtypes)

# Rename your objects column 'code' to 'country_code'
us_data_shell.rename_column('code', 'country_code')

# Again, print the datatype of your object's data_as_csv attribute
print(us_data_shell.data_as_csv.dtypes)
```

加入回傳自身的資訊



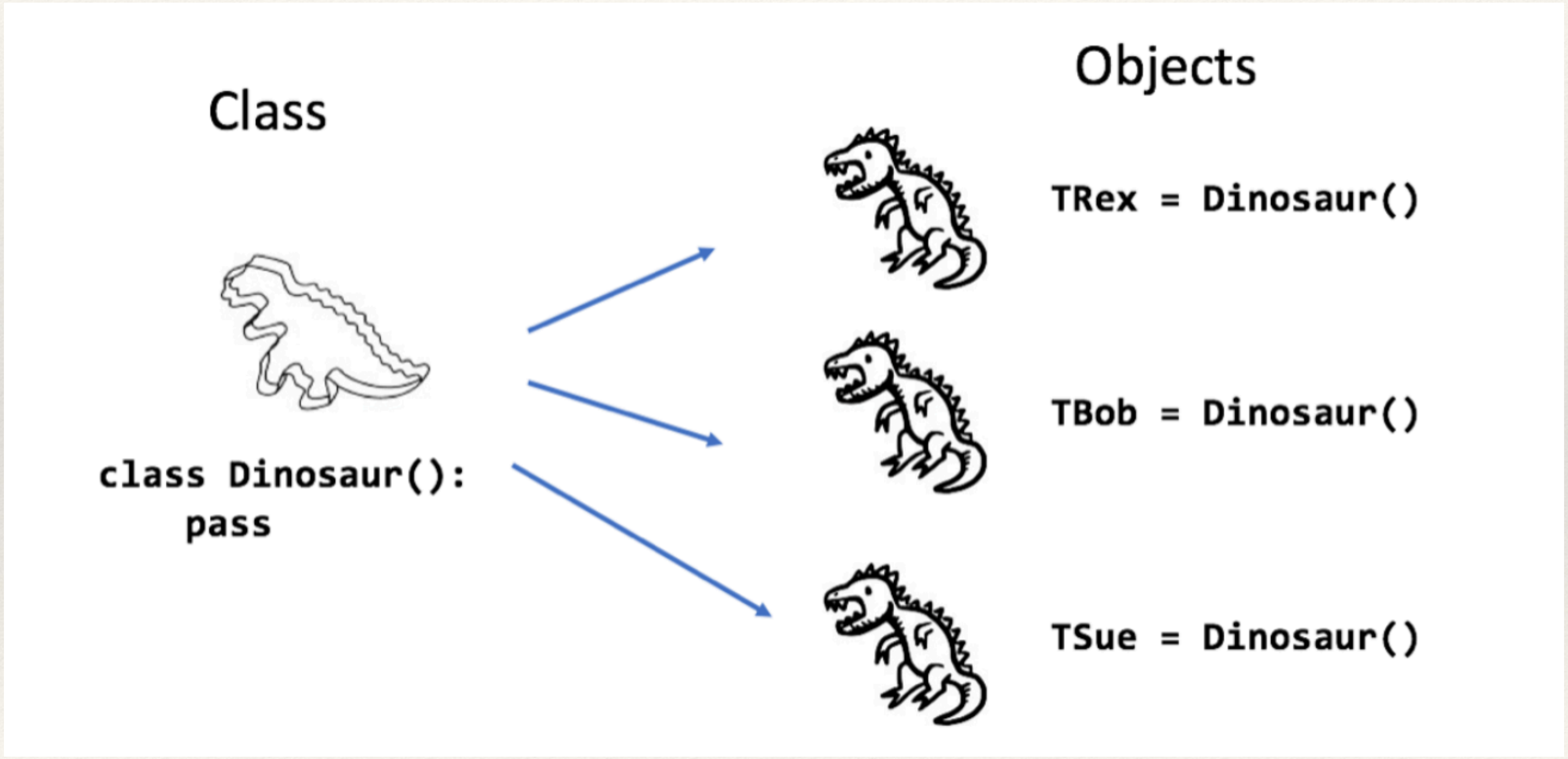
```python
class DataShell:
    def __init__(self, filepath):
        self.filepath = filepath
        self.data_as_csv = pd.read_csv(filepath)

    # 定義第一個方法 rename_column, with arguments self, column_name, and new_column_name
    def rename_column(self, column_name, new_column_name):
        self.data_as_csv.columns = self.data_as_csv.columns.str.replace(column_name, new_column_name)

    # 定義第二個 get_stats method, with argument self
    def get_stats(self):
        # Return a description data_as_csv
        return self.data_as_csv.describe()

us_data_shell = DataShell(us_life_expectancy)

# Print the output of your objects get_stats method
print(us_data_shell.get_stats())
```

有關 Python 物件導向程式設計的一些建議指引

參考其他人的寫法

- GitHubCode
- Python 官方程式文件
- 參考 codebase:
  - Pandas as Spark
  - PEP Style

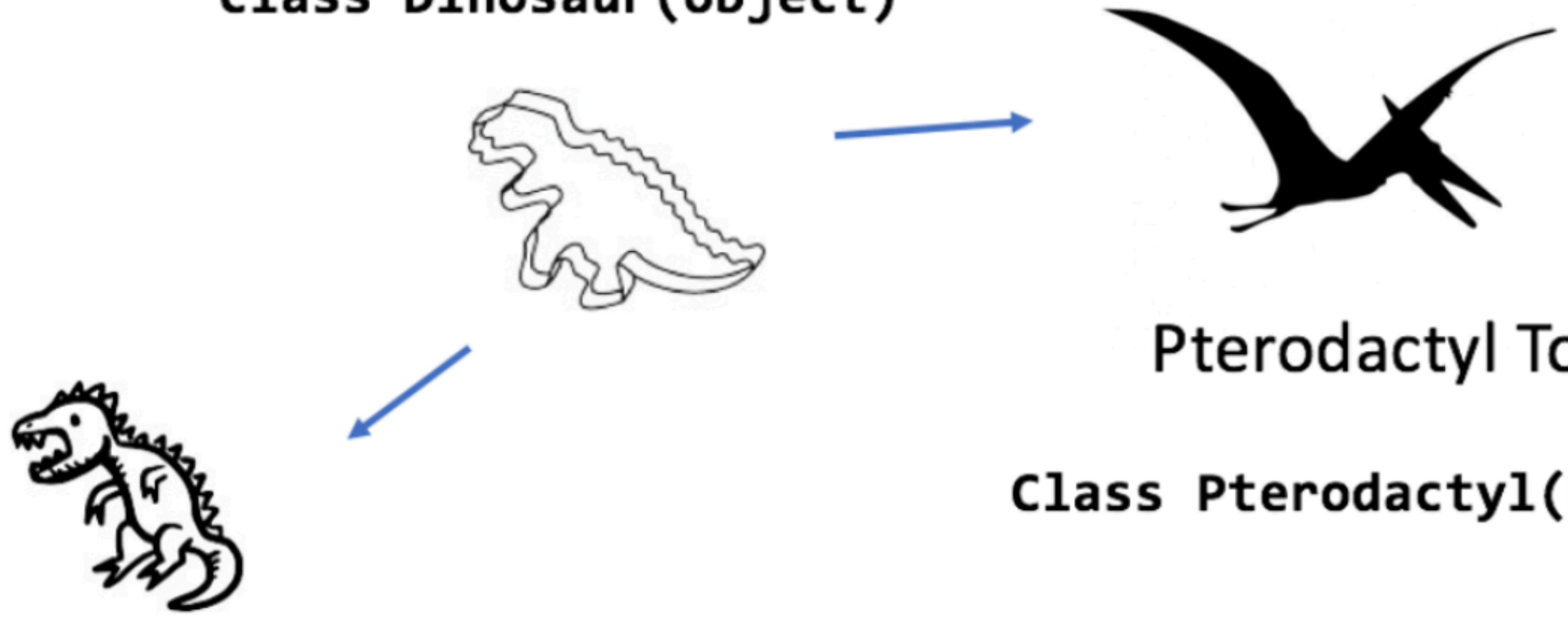
繼承 (Inheritance)



```
class Dinosaur():
    pass
```

```
TRex = Dinosaur()
TBob = Dinosaur()
TSue = Dinosaur()
```

Is a and Has a Relationship



```
class Dinosaur(object)
```

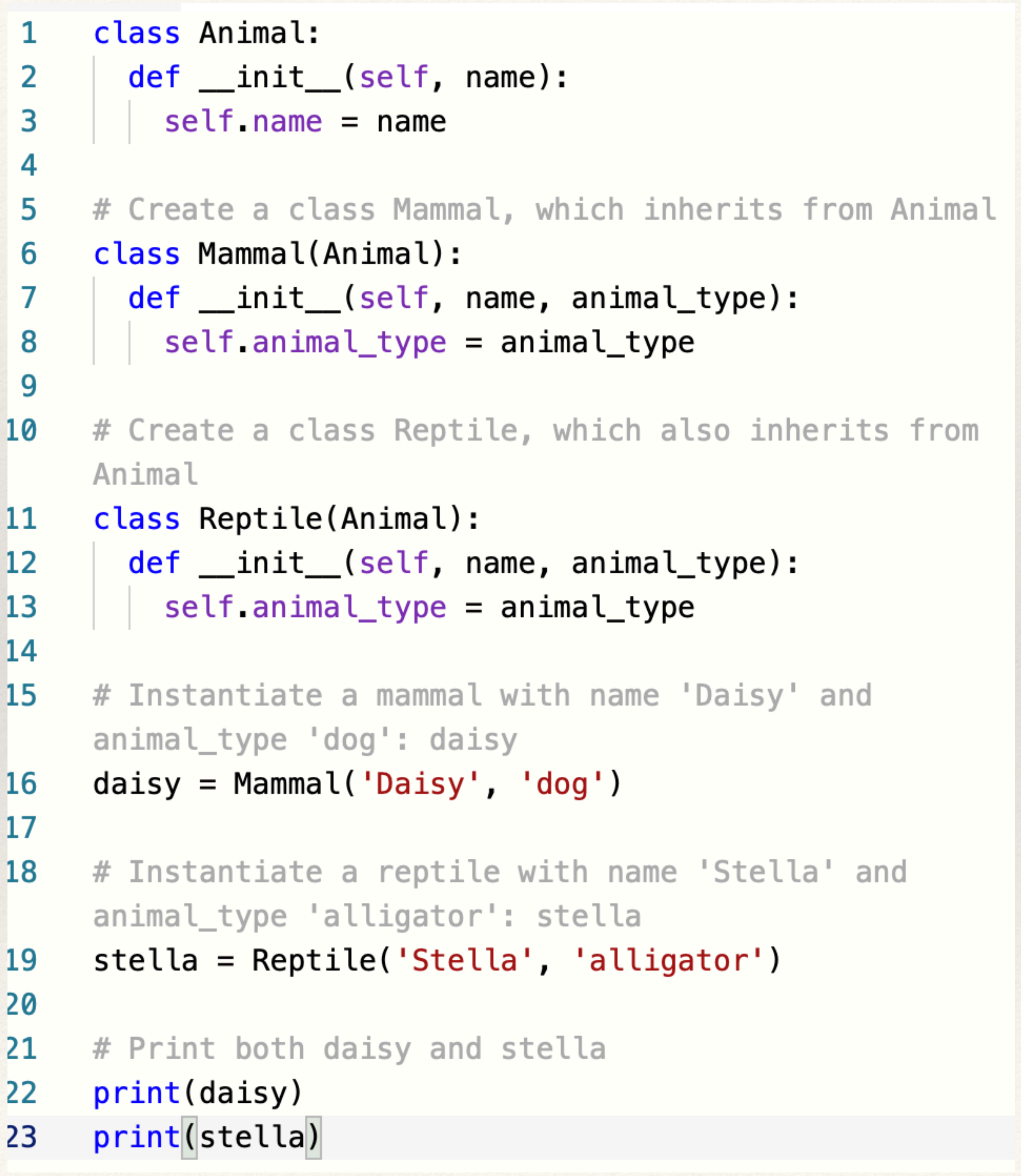
Pterodactyl Tom

```
Class Pterodactyl(Dinosaur)
```

Tyrannosaurus Sue

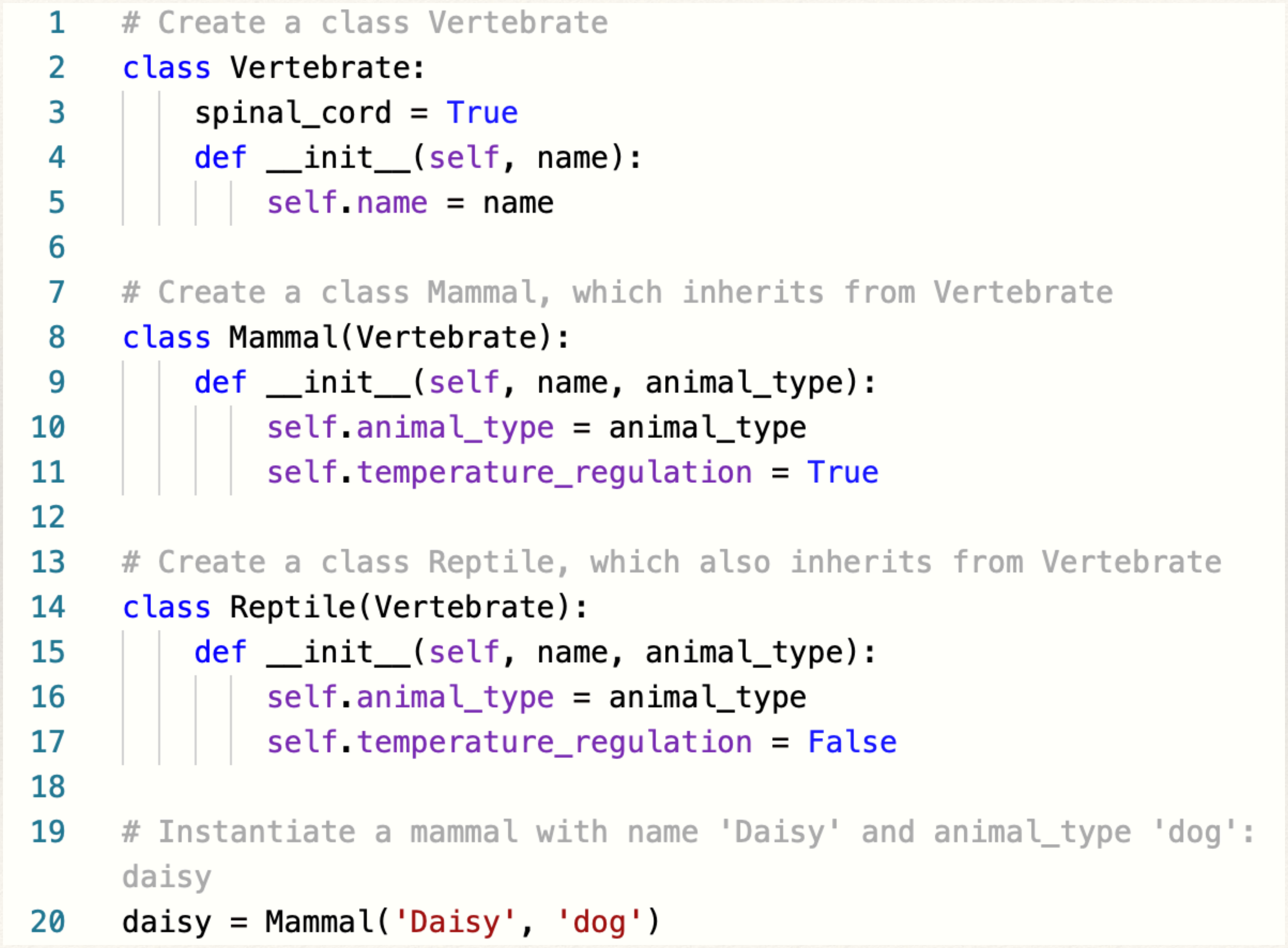
```
class Tyrannosaurus(Dinosaur)
```

Inheritance, simple one

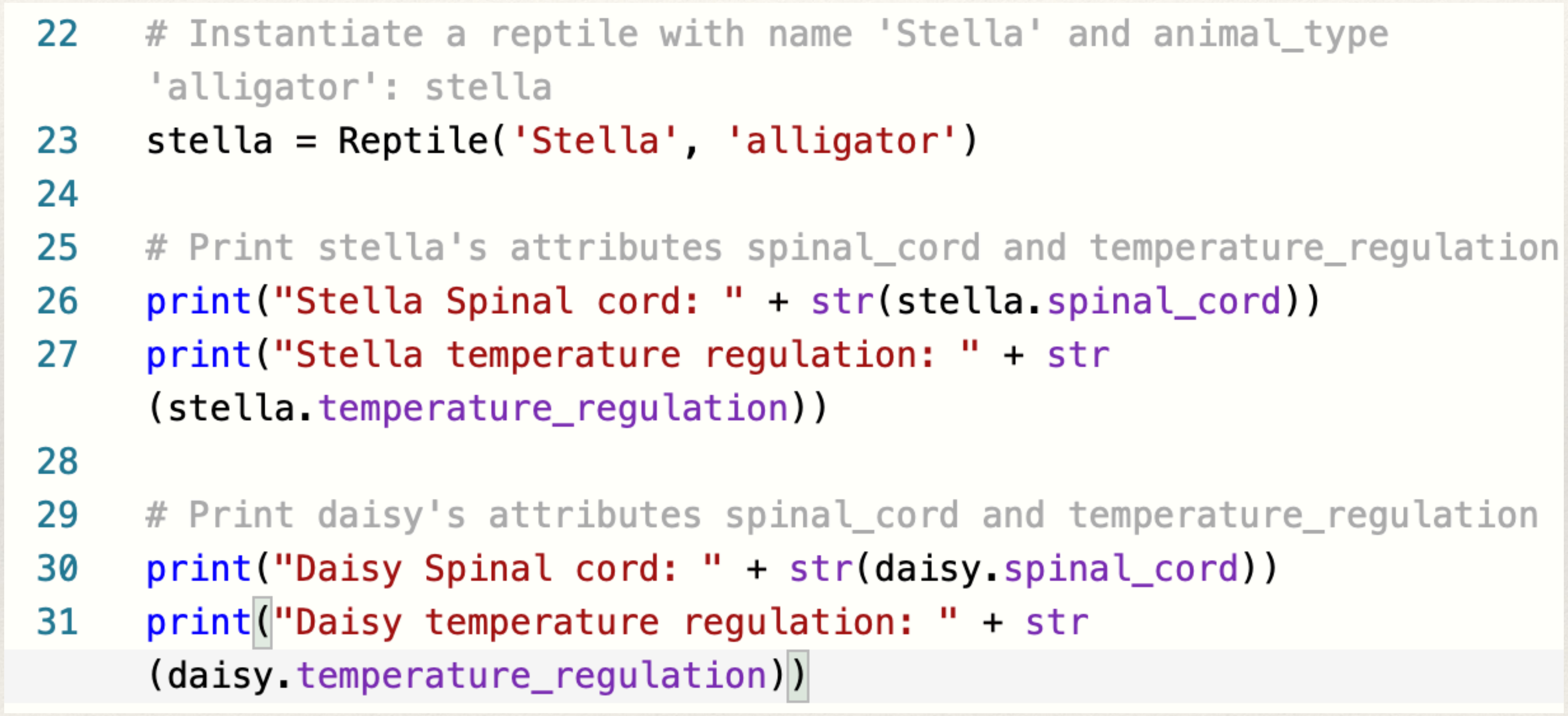


```python
class Animal:
    def __init__(self, name):
        self.name = name

# Create a class Mammal, which inherits from Animal
class Mammal(Animal):
    def __init__(self, name, animal_type):
        self.animal_type = animal_type

# Create a class Reptile, which also inherits from Animal
class Reptile(Animal):
    def __init__(self, name, animal_type):
        self.animal_type = animal_type

# Instantiate a mammal with name 'Daisy' and animal_type 'dog': daisy
daisy = Mammal('Daisy', 'dog')

# Instantiate a reptile with name 'Stella' and animal_type 'alligator': stella
stella = Reptile('Stella', 'alligator')

# Print both daisy and stella
print(daisy)
print(stella)
```

Inheritance, more complete

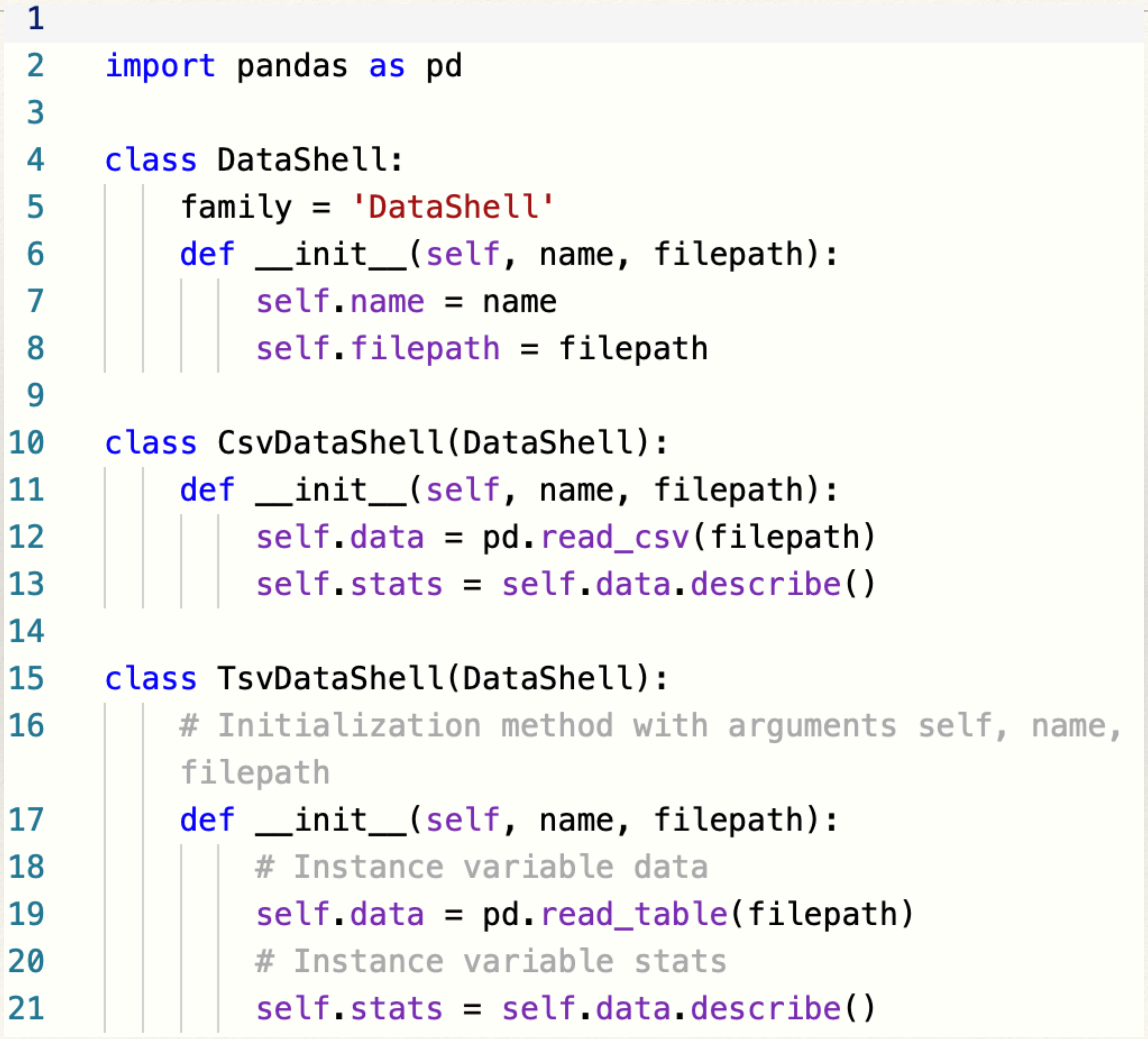


```python
# Create a class Vertebrate
class Vertebrate:
    spinal_cord = True
    def __init__(self, name):
        self.name = name

# Create a class Mammal, which inherits from Vertebrate
class Mammal(Vertebrate):
    def __init__(self, name, animal_type):
        self.animal_type = animal_type
        self.temperature_regulation = True

# Create a class Reptile, which also inherits from Vertebrate
class Reptile(Vertebrate):
    def __init__(self, name, animal_type):
        self.animal_type = animal_type
        self.temperature_regulation = False

# Instantiate a mammal with name 'Daisy' and animal_type 'dog': daisy
daisy = Mammal('Daisy', 'dog')
```

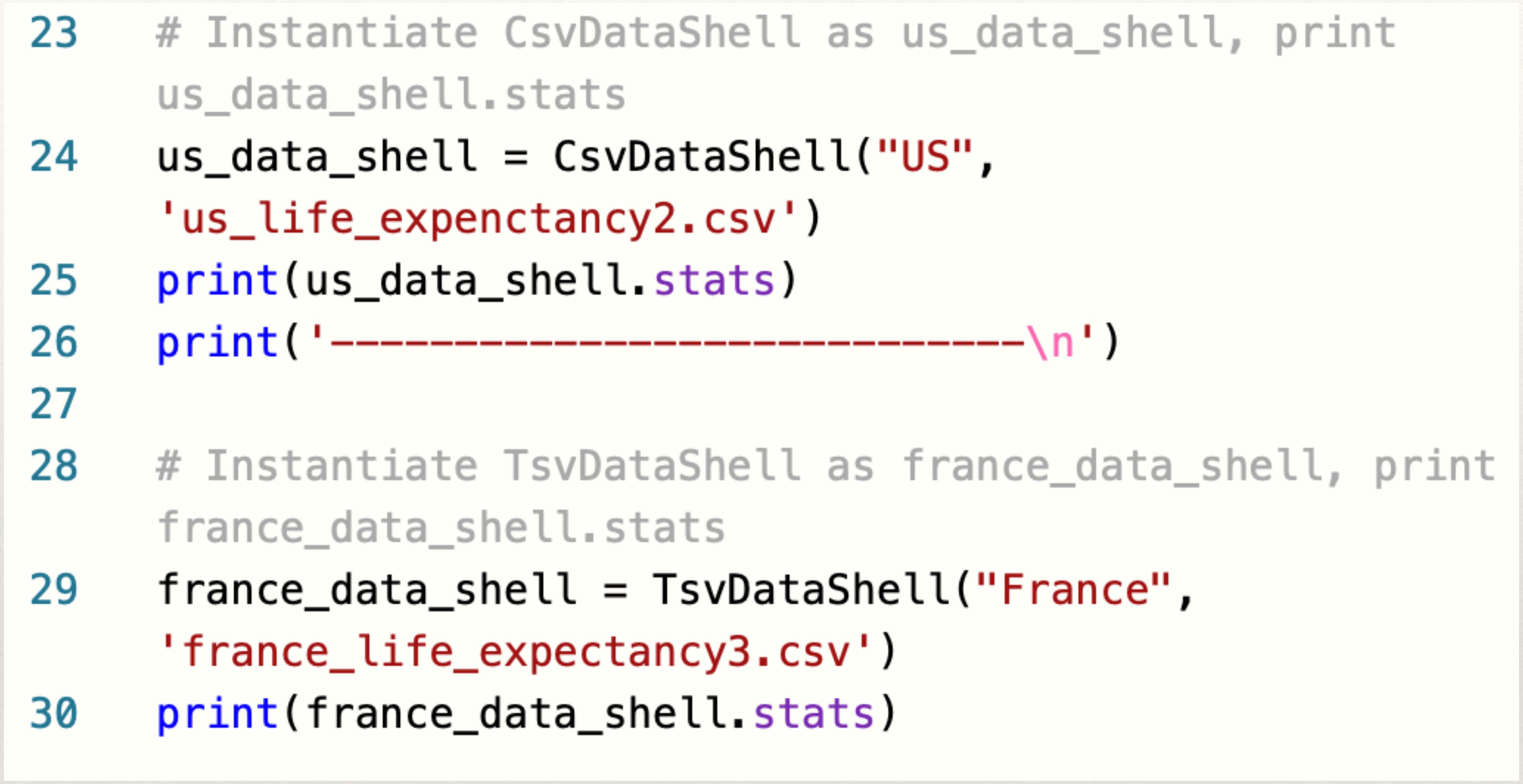


```python
# Instantiate a reptile with name 'Stella' and animal_type 'alligator': stella
stella = Reptile('Stella', 'alligator')

# Print stella's attributes spinal_cord and temperature_regulation
print("Stella Spinal cord: " + str(stella.spinal_cord))
print("Stella temperature regulation: " + str(stella.temperature_regulation))

# Print daisy's attributes spinal_cord and temperature_regulation
print("Daisy Spinal cord: " + str(daisy.spinal_cord))
print("Daisy temperature regulation: " + str(daisy.temperature_regulation))
```

實際案例練習：



```python

import pandas as pd

class DataShell:
    family = 'DataShell'
    def __init__(self, name, filepath):
        self.name = name
        self.filepath = filepath

class CsvDataShell(DataShell):
    def __init__(self, name, filepath):
        self.data = pd.read_csv(filepath)
        self.stats = self.data.describe()

class TsvDataShell(DataShell):
    # Initialization method with arguments self, name, filepath
    def __init__(self, name, filepath):
        # Instance variable data
        self.data = pd.read_table(filepath)
        # Instance variable stats
        self.stats = self.data.describe()
```



```python
# Instantiate CsvDataShell as us_data_shell, print us_data_shell.stats
us_data_shell = CsvDataShell("US", 'us_life_expenctancy2.csv')
print(us_data_shell.stats)
print('----------------------------\n')

# Instantiate TsvDataShell as france_data_shell, print france_data_shell.stats
france_data_shell = TsvDataShell("France", 'france_life_expectancy3.csv')
print(france_data_shell.stats)
```