Python Programming (Basic-Intermediate)

Module 6 - Advanced Topics

Object Oriented Programming (OOP)

Creating the class

```
class Dog:
    """A simple class of dogs"""

    def __init__(self, name, age):
        """Initialize name and age attrbutes."""
        self.name = name
        self.age = age

    def sit(self):
        """Simulate a dog sitting in response to the command."""
        print(f"{self.name} is now sitting.")

    def roll_over(self):
        """SImulate rolling over in response to the command."""
        print(f"{self.name} rolled over.")
```

```
dir()
```

```
['Dog',
 'In',
 'Out',
 '_',
 '__',
 '___',
 '__builtin__',
 '__builtins__',
 '__doc__',
 '__loader__',
 '__name__',
 '__package__',
 '__spec__',
```

```
 '_dh',
 '_i',
 '_i1',
 '_i2',
 '_ih',
 '_ii',
 '_iii'
```

```
my_dog = Dog('Will', 6)

print(f"My dog's name is {my_dog.name}.")
print(f"My dog is {my_dog.age} years old.")
```

```
My dog's name is Will.
My dog is 6 years old.
```

Calling methods

```
my_dog.sit()
```

```
Will is now sitting.
```

```
my_dog.roll_over()
```

```
Will rolled over.
```

Creating multiple instances

```
my_dog = Dog('Will', 6)
your_dog = Dog('Black', 3)

print(f"My dog's name is {my_dog.name}.")
print(f"My dog is {my_dog.age} years old.")

print(f"My dog's name is {your_dog.name}.")
print(f"My dog is {your_dog.age} years old.")
your_dog.sit()
```

```
My dog's name is Will.
My dog is 6 years old.
My dog's name is Black.
```

```
My dog is 3 years old.
Black is now sitting.
```

Car class

```
class Car:
    """A simple attempt to represent a car."""

    def __init__(self, make, model, year):
        """Initialize attributes to describe a car."""
        self.make = make
        self.model = model
        self.year = year

    def get_description_name(self):
        """Return a neatly formatted descriptive name."""
        long_name = f"{self.year} {self.make} {self.model}"
        return long_name.title()
```

```
my_old_car = Car('Honda','Accord', 2018)
print(my_old_car.get_description_name())
```

```
2018 Honda Accord
```

Default value for an attribute

```
class Car:
    """A simple attempt to represent a car."""

    def __init__(self, make, model, year):
        """Initialize attributes to describe a car."""
        self.make = make
        self.model = model
        self.year = year
        self.odometer_reading = 0

    def get_description_name(self):
        """Return a neatly formatted descriptive name."""
        long_name = f"{self.year} {self.make} {self.model}"
        return long_name.title()

    def read_odometer(self):
        """Print a statement showing the car's mileage."""
        print(f"This car has {self.odometer_reading} miles on it.")
```

```
my_old_car = Car('Honda','Accord', 2018)
print(my_old_car.get_description_name())
my_old_car.read_odometer()
```

```
2018 Honda Accord
This car has 0 miles on it.
```

```
my_old_car.odometer_reading = 97
my_old_car.read_odometer()
```

```
This car has 97 miles on it.
```

Modifying attribute values through a method

```
class Car:
    """A simple attempt to represent a car."""

    def __init__(self, make, model, year):
        """Initialize attributes to describe a car."""
        self.make = make
        self.model = model
        self.year = year
        self.odometer_reading = 0

    def get_description_name(self):
        """Return a neatly formatted descriptive name."""
        long_name = f"{self.year} {self.make} {self.model}"
        return long_name.title()

    def read_odometer(self):
        """Print a statement showing the car's mileage."""
        print(f"This car has {self.odometer_reading} miles on it.")

    def update_odometer(self, mileage):
        """Set the odometer reading to the given value."""
        self.odometer_reading = mileage
```

```
my_old_car = Car('Honda','Accord', 2018)
print(my_old_car.get_description_name())
my_old_car.update_odometer(80)
my_old_car.read_odometer()
```

```
2018 Honda Accord
This car has 80 miles on it.
```

```
class Car:
    """A simple attempt to represent a car."""

    def __init__(self, make, model, year):
        """Initialize attributes to describe a car."""
        self.make = make
        self.model = model
        self.year = year
        self.odometer_reading = 0

    def get_description_name(self):
        """Return a neatly formatted descriptive name."""
        long_name = f"{self.year} {self.make} {self.model}"
        return long_name.title()

    def read_odometer(self):
        """Print a statement showing the car's mileage."""
        print(f"This car has {self.odometer_reading} miles on it.")

    def update_odometer(self, mileage):
        """Set the odometer reading to the given value."""
        self.odometer_reading = mileage

    def increment_odometer(self, miles):
        self.odometer_reading += miles
```

```
my_old_car = Car('Honda','Accord', 2018)
print(my_old_car.get_description_name())
my_old_car.update_odometer(1500)
my_old_car.read_odometer()
my_old_car.increment_odometer(100)
my_old_car.read_odometer()
```

```
2018 Honda Accord
This car has 1500 miles on it.
This car has 1600 miles on it.
```

Default printing method

```
print(my_old_car)
```

```
<__main__.Car object at 0x7898f6904e80>
```

```
class Car:
    """A simple attempt to represent a car."""

    def __init__(self, make, model, year):
        """Initialize attributes to describe a car."""
        self.make = make
        self.model = model
        self.year = year
        self.odometer_reading = 0

    def get_description_name(self):
        """Return a neatly formatted descriptive name."""
        long_name = f"{self.year} {self.make} {self.model}"
        return long_name.title()

    def __str__(self):
        return self.get_description_name()

    def read_odometer(self):
        """Print a statement showing the car's mileage."""
        print(f"This car has {self.odometer_reading} miles on it.")

    def update_odometer(self, mileage):
        """Set the odometer reading to the given value."""
        self.odometer_reading = mileage

    def increment_odometer(self, miles):
        self.odometer_reading += miles
```

```
my_old_car = Car('Honda', 'Accord', 2018)
print(my_old_car)
```

```
2018 Honda Accord
```

Default object display method

```
my_old_car
```

```
<__main__.Car at 0x7898f6906110>
```

```
class Car:
    """A simple attempt to represent a car."""

    def __init__(self, make, model, year):
        """Initialize attributes to describe a car."""
        self.make = make
        self.model = model
        self.year = year
        self.odometer_reading = 0

    def get_description_name(self):
        """Return a neatly formatted descriptive name."""
        long_name = f"{self.year} {self.make} {self.model}"
        return long_name.title()

    def __str__(self):
        return self.get_description_name()

    def __repr__(self):
        return "Car Instance: " +  self.get_description_name()

    def read_odometer(self):
        """Print a statement showing the car's mileage."""
        print(f"This car has {self.odometer_reading} miles on it.")

    def update_odometer(self, mileage):
        """Set the odometer reading to the given value."""
        self.odometer_reading = mileage

    def increment_odometer(self, miles):
        self.odometer_reading += miles
```

```
Car('Honda', 'Accord', 2018)
```

```
Car Instance: 2018 Honda Accord
```

```
import pandas as pd
df = pd.read_csv('/content/sample_data/california_housing_test.csv')
```

```
print(df)
```

```
      longitude  latitude  housing_median_age  total_rooms  total_bed
0       -122.05     37.37                27.0       3885.0
1       -118.30     34.26                43.0       1510.0
2       -117.81     33.78                27.0       3589.0
3       -118.36     33.82                28.0         67.0
4       -119.67     36.33                19.0       1241.0
```

```
...         ...       ...                 ...          ...
2995    -119.86     34.42                23.0       1450.0
2996    -118.14     34.06                27.0       5257.0          1
2997    -119.70     36.30                10.0        956.0
2998    -117.12     34.10                40.0         96.0
2999    -119.63     34.42                42.0       1765.0

      population  households  median_income  median_house_value
0         1537.0       606.0         6.6085            344700.0
1          809.0       277.0         3.5990            176500.0
2         1484.0       495.0         5.7934            270500.0
3           49.0        11.0         6.1359            330000.0
4          850.0       237.0         2.9375             81700.0
```

```
df
```

```
      longitude  latitude  housing_median_age  total_rooms  total_bed
0       -122.05     37.37                27.0       3885.0
1       -118.30     34.26                43.0       1510.0
2       -117.81     33.78                27.0       3589.0
3       -118.36     33.82                28.0         67.0
4       -119.67     36.33                19.0       1241.0
...         ...       ...                 ...          ...
2995    -119.86     34.42                23.0       1450.0
2996    -118.14     34.06                27.0       5257.0          1
2997    -119.70     36.30                10.0        956.0
2998    -117.12     34.10                40.0         96.0
2999    -119.63     34.42                42.0       1765.0

      population  households  median_income  median_house_value
0         1537.0       606.0         6.6085            344700.0
1          809.0       277.0         3.5990            176500.0
2         1484.0       495.0         5.7934            270500.0
3           49.0        11.0         6.1359            330000.0
4          850.0       237.0         2.9375             81700.0
```

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	media
0	-122.05	37.37	27.0	3885.0	661.0	1537.0	606.0	6.6085
1	-118.30	34.26	43.0	1510.0	310.0	809.0	277.0	3.5990
2	-117.81	33.78	27.0	3589.0	507.0	1484.0	495.0	5.7934

Inheritance

```
class ElectricCar(Car):
    """Represent aspect of a car, specific to electric vehicles."""

    def __init__(self, make, model, year):
        """Initialize attributes of the parent class."""
        """Then initialize attributes specific to an electric car."""
        super().__init__(make, model, year)
        self.battery_size = 75

    def describe_battery(self):
        """Print a statement describing the battery size."""
        print(f"This car has a {self.battery_size}-kWh battery.")
```

```
my_tesla = ElectricCar('tesla','model s', 2020)
print(my_tesla)
my_tesla.describe_battery()
```

```
2020 Tesla Model S
This car has a 75-kWh battery.
```

Override parent class method

```
class ElectricCar(Car):
    """Represent aspect of a car, specific to electric vehicles."""

    def __init__(self, make, model, year):
        """Initialize attributes of the parent class."""
        """Then initialize attributes specific to an electric car."""
        super().__init__(make, model, year)
        self.battery_size = 75

    def describe_battery(self):
        """Print a statement describing the battery size."""
        print(f"This car has a {self.battery_size}-kWh battery.")

    def read_odometer(self):
        """Override the parent read_odometer method"""
        print(f"Overrided: This car has {self.odometer_reading} miles on
```

```
my_tesla = ElectricCar('tesla','model s', 2020)
print(my_tesla)
my_tesla.read_odometer()
```

```
2020 Tesla Model S
Overrided: This car has 0 miles on it.
```

Storing and importing class from modules

```
del Car, ElectricCar
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
import sys
sys.path.append('/content/drive/MyDrive/AIS_DG/lib')
```

```
from car import Car, ElectricCar
```

```
x = ElectricCar('tesla', 'model y', 2022)
print(x)
x.describe_battery()
```

```
2022 Tesla Model Y
This car has a 75-kWh battery.
```

Exception

Using try-except block

```
print(5/0)
```

```
division by zero
```

```
try:
    print(5/0)
except:
    print("You cannot divide by zero.")
```

```
You cannot divide by zero.
```

try..except..else block

```
x = 0
try:
    y = 5/x
except:
    print("You cannot divide by 0.")
else:
    print(f"The answer of 5/{x} is {y}.")
```

```
You cannot divide by 0.
```

```
x = 2
try:
    y = 5/x
except:
    print("You cannot divide by 0.")
else:
    print(f"The answer of 5/{x} is {y}.")
```

```
The answer of 5/2 is 2.5.
```

Handling a specific exception

```
filename = 'data.txt'

with open(filename) as f:
    contents = f.read()
```

```
[Errno 2] No such file or directory: 'data.txt'
```

```
filename = 'data.txt'

try:
    with open(filename) as f:
        contents = f.read()
except FileNotFoundError:
    print(f"Sorry, the file {filename} does not exist.")
```

```
Sorry, the file data.txt does not exist.
```

finally

```
x = 2
try:
    y = 5/x
except:
    print("You cannot divide by 0.")
else:
    print(f"The answer of 5/{x} is {y}.")
finally:
    print("----------------------------")
```

```
The answer of 5/2 is 2.5.
----------------------------
```

```
x = 0
try:
    y = 5/x
except:
    print("You cannot divide by 0.")
else:
    print(f"The answer of 5/{x} is {y}.")
finally:
    print("----------------------------")
```

```
You cannot divide by 0.
----------------------------
```

```
!pip install pyspark
```

```
Collecting pyspark
  Downloading pyspark-3.5.0.tar.gz (316.9 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 316.9/316.9 MB 4.3 MB/s
  Preparing metadata (setup.py) ... done
Requirement already satisfied: py4j==0.10.9.7 in /usr/local/lib/python
Building wheels for collected packages: pyspark
  Building wheel for pyspark (setup.py) ... done
  Created wheel for pyspark: filename=pyspark-3.5.0-py2.py3-none-any.w
  Stored in directory: /root/.cache/pip/wheels/41/4e/10/c2cf2467f71c67
Successfully built pyspark
Installing collected packages: pyspark
Successfully installed pyspark-3.5.0
```

```
import pyspark.pandas as ps
```

```
/usr/local/lib/python3.10/dist-packages/pyspark/pandas/__init__.py:50:
  warnings.warn(
```

```
df = ps.read_csv('/content/drive/MyDrive/AIS_DG/Flight_flights.csv')
```

```
/usr/local/lib/python3.10/dist-packages/pyspark/pandas/utils.py:1016:
  warnings.warn(message, PandasAPIOnSparkAdviceWarning)
```

```
type(df)
```

```
pyspark.pandas.frame.DataFrame
```