Lesson 10 - Pythonic objects 2

July 2, 2020

1 Agenda

- Object Representations
- Making our own collections
- Mixins
- Properties

2 Object Representations

[1]:
```
# %load repr.py
class Student(object):

    def __init__(self, first_name, last_name, age):
        self.first_name = first_name
        self.last_name = last_name
        self.age = age

    def user_description(self):
        return f'The student {self.first_name} {self.last_name} is {self.age}'


anthony = Student('Anthony', 'Hopkins', 21)
print(anthony)

print(anthony.user_description())
```

```
<__main__.Student object at 0x7f76e45ed450>
The student Anthony Hopkins is 21
```

repr() - return a string representing the object as the developer wants to see it

str() - return a string representing the object as the user wants to see it

[2]:
```
# %load repr_2.py
class Student(object):
    def __init__(self, first_name, last_name, age):
        self.first_name = first_name
        self.last_name = last_name
        self.age = age

    def __repr__(self):
        class_name = type(self).__name__
        return '{} ({} {} - {}) [{}]'.format(
            class_name, self.first_name,
            self.last_name, self.age, id(self)
        )

    def __str__(self):
        return '{} {}, {}'.format(
            self.first_name, self.last_name, self.age)


anthony = Student('Anthony', 'Hopkins', 21)
print(repr(anthony))
print(str(anthony))
```

```
Student (Anthony Hopkins - 21) [140148614348880]
Anthony Hopkins, 21
```

[3]:
```
# %load school_bus.py
from repr_2 import Student


class SchoolBus():

    def __init__(self, students_list=None):
        self.students = list(students_list) if students_list else []


students = [
    Student('John', 'Doe', 19),
    Student('Jack', 'Fluffy', 18),
    Student('Matthew', 'Wu', 19),
    Student('Heather', 'Rafferty', 19),
    Student('Randall', 'Blackdall', 20),
    Student('Marissa', 'Raynaud', 19),
    Student('Marlo', 'Ranbot', 19)
]

bus = SchoolBus(students)
```

```
for student in bus.students:
    print(student)
```

```
John Doe, 19
Jack Fluffy, 18
Matthew Wu, 19
Heather Rafferty, 19
Randall Blackdall, 20
Marissa Raynaud, 19
Marlo Ranbot, 19
```

3 Making our own collections

[4]:
```
# %load school_bus2.py
from repr_2 import Student


class SchoolBus():

    def __init__(self, students_list=None):
        self._students = list(students_list) if students_list else []

    def __iter__(self):
        return iter(self._students)


students = [
    Student('John', 'Doe', 19),
    Student('Jack', 'Fluffy', 18),
    Student('Matthew', 'Wu', 19),
    Student('Heather', 'Rafferty', 19),
    Student('Randall', 'Blackdall', 20),
    Student('Marissa', 'Raynaud', 19),
    Student('Marlo', 'Ranbot', 19)
]

bus = SchoolBus(students)

for student in bus:
    print(student)

first_names = [stud.first_name for stud in bus]
print(first_names)
```

```
John Doe, 19
Jack Fluffy, 18
```

```
Matthew Wu, 19
Heather Rafferty, 19
Randall Blackdall, 20
Marissa Raynaud, 19
Marlo Ranbot, 19
['John', 'Jack', 'Matthew', 'Heather', 'Randall', 'Marissa', 'Marlo']
```

[5]:
```
# %load school_bus3.py
from repr_2 import Student


class SchoolBus():

    def __init__(self, students_list=None):
        self._students = list(students_list) if students_list else []

    def __iter__(self):
        return iter(self._students)

    def __str__(self):
        students = [
            '[{}] {} {}, {}'.format(
                index, stud.first_name, stud.last_name, stud.age)
            for index, stud in
            enumerate(self._students)
        ]

        return ' \n'.join(students)


students = [
    Student('John', 'Doe', 19),
    Student('Jack', 'Fluffy', 18),
    Student('Matthew', 'Wu', 19),
    Student('Heather', 'Rafferty', 19),
    Student('Randall', 'Blackdall', 20),
    Student('Marissa', 'Raynaud', 19),
    Student('Marlo', 'Ranbot', 19)
]

bus = SchoolBus(students)
print(bus)
```

```
[0] John Doe, 19
[1] Jack Fluffy, 18
[2] Matthew Wu, 19
[3] Heather Rafferty, 19
[4] Randall Blackdall, 20
```

```
[5] Marissa Raynaud, 19
[6] Marlo Ranbot, 19
```

[6]:
```
bus[1]
```

```
                                                                              ␣
↪---------------------------------------------------------------------------

    TypeError                                 Traceback (most recent call␣
↪last)

    <ipython-input-6-b6a5a4094801> in <module>
----> 1 bus[1]


    TypeError: 'SchoolBus' object is not subscriptable
```

[ ]:
```
# %load school_bus4.py
from repr_2 import Student


class SchoolBus():

    def __init__(self, students_list=None):
        self._students = list(students_list) if students_list else []

    def __iter__(self):
        return iter(self._students)

    def __getitem__(self, index):
        return self._students[index]

    def __str__(self):
        students = [
            '[{}] {} {}, {}'.format(
                index, stud.first_name, stud.last_name, stud.age)
            for index, stud in
            enumerate(self._students)
        ]

        return ' \n'.join(students)


students = [
    Student('John', 'Doe', 19),
    Student('Jack', 'Fluffy', 18),
    Student('Matthew', 'Wu', 19),
    Student('Heather', 'Rafferty', 19),
    Student('Randall', 'Blackdall', 20),
    Student('Marissa', 'Raynaud', 19),
    Student('Marlo', 'Ranbot', 19)
]

bus = SchoolBus(students)
print(bus)
```

[ ]:
```
bus[2]
```

[ ]:
```
print(bus[2])
```

4 Mixins

[ ]:
```
# %load mixin.py
class LoggingMixin:

    def __init__(self, *args, **kwargs):
        class_name = type(self).__name__
        print('Initializing object of type {}'.format(class_name))

        return super().__init__(*args, **kwargs)

    def __getitem__(self, index):
        print('Getting key ' + str(index))

        return super().__getitem__(index)
```

[ ]:
```
# %load school_bus5.py
from student import Student
from mixin import LoggingMixin


class SchoolBus():

    def __init__(self, students_list=None):
        self._students = list(students_list) if students_list else []

    def __iter__(self):
        return iter(self._students)

    def __getitem__(self, index):
        return self._students[index]

    def __str__(self):
        students = [
            '[{}] {} {}, {}'.format(
                index, stud.first_name, stud.last_name, stud.age)
            for index, stud in
            enumerate(self._students)
        ]

        return ' \n'.join(students)


class LoggingSchoolBus(LoggingMixin, SchoolBus):
    pass


students = [
    Student('John', 'Doe', 19),
    Student('Jack', 'Fluffy', 18),
    Student('Matthew', 'Wu', 19),
    Student('Heather', 'Rafferty', 19),
    Student('Randall', 'Blackdall', 20),
    Student('Marissa', 'Raynaud', 19),
    Student('Marlo', 'Ranbot', 19)
]

bus = LoggingSchoolBus(students)
print(bus)
print(bus[1])
```

[ ]:

```
# %load mixin2.py
class LoggingMixin:

    def __init__(self, *args, **kwargs):
        class_name = type(self).__name__
        print('Initializing object of type {}'.format(class_name))

        return super().__init__(*args, **kwargs)

    def __getitem__(self, index):
        print('Getting key ' + str(index))

        return super().__getitem__(index)


class StudentsCollection:
```

```python

    def __init__(self, students_list=None):
        self._students = list(students_list) if students_list else []

    def __iter__(self):
        return iter(self._students)

    def __getitem__(self, index):
        return self._students[index]
```

[ ]:
```python
# %load student.py
class Student(object):

    def __init__(self, first_name, last_name, age):
        self.first_name = first_name
        self.last_name = last_name
        self.age = age

    def __repr__(self):
        class_name = type(self).__name__
        return '{} ({} {} - {}) [{}]'.format(
            class_name, self.first_name,
            self.last_name, self.age, id(self)
        )

    def __str__(self):
        return '{} {}, {}'.format(
            self.first_name, self.last_name, self.age)
```

[ ]:
```python
# %load school_bus6.py
from student import Student
from mixin2 import LoggingMixin, StudentsCollection


class SchoolBusPrinter():

    def __str__(self):
        students = [
            '[{}] {} {}, {}'.format(
                index, stud.first_name, stud.last_name, stud.age)
            for index, stud in
            enumerate(self._students)
        ]

        return ' \n'.join(students)


```

```python
class SchoolBus(StudentsCollection, SchoolBusPrinter):
    pass


class LoggingSchoolBus(LoggingMixin, SchoolBus):
    pass


students = [
    Student('John', 'Doe', 19),
    Student('Jack', 'Fluffy', 18),
    Student('Matthew', 'Wu', 19),
    Student('Heather', 'Rafferty', 19),
    Student('Randall', 'Blackdall', 20),
    Student('Marissa', 'Raynaud', 19),
    Student('Marlo', 'Ranbot', 19)
]

simple_bus = SchoolBus(students)
print(simple_bus)
print(simple_bus[3])

logging_bus = LoggingSchoolBus(students)
print(logging_bus)
print(logging_bus[4])
```

5 Properties

[ ]:
```python
# %load student2.py
class Student(object):

    def __init__(self, first_name, last_name, age):
        self.first_name = first_name
        self.last_name = last_name
        self._age = age

    @property
    def age(self):
        return self._age

    @age.setter
    def age(self, age):
        if type(age) is not int:
            raise TypeError('Age must be integer')
        if age < 18:
            raise ValueError('Minimal age is 18')

        self._age = age

    def __repr__(self):
        class_name = type(self).__name__
        return '{} ({} {} - {}) [{}]'.format(
            class_name, self.first_name,
            self.last_name, self.age, id(self)
        )

    def __str__(self):
        return '{} {}, {}'.format(
            self.first_name, self.last_name, self.age)
```

[ ]:
```
john = Student('Elton', 'John', 21)
john.age
```

[ ]:
```
john.age = 16
```

[ ]:
```
joe = Student('Joe', 'Dow', 16)
```

[ ]:
```
joe.age
```

6 Challenge

Update the **Student** class so that the validations are applied also at **init** not allowing an object to be created if age is not valid.