ООП.
Объекты, аттрибуты, слоты, цепочки методов, абстрактные классы и методы, множественное наследование

Преподаватель: Тенигин Альберт Андреевич

Сириус
IT-Колледж

Наследование от object

```
(class) object

The base class of the class hierarchy.

When called, it accepts no arguments and returns a new featureless instance that has
no instance attributes and cannot be given any.
```

```
class Example(object):
    def __init__(self):
        self.a = 5
        self.b = 6
```

Сириус
IT-Колледж

```
class Example:
    def __init__(self):
        self.a = 5
        self.b = 6
```

Сириус
IT-Колледж

```
class Example:
    def __init__(self):
        self.a = 5
        self.b = 6

obj = Example()
setattr(obj, 'c', 7)
print('obj.c =', obj.c)
```

Сириус
IT-Колледж

```
class Example:
    def __init__(self):
        self.a = 5
        self.b = 6

obj = Example()
setattr(obj, 'c', 7)
print('obj.c =', obj.c)
```

```
obj.c равен 7
```

Сириус
IT-Колледж

```
class Example:
    def __init__(self):
        self.a = 5
        self.b = 6

obj = Example()
setattr(obj, 'c', 7)
obj.d = 8
print('obj.d =', obj.d)
```

Сириус
IT-Колледж

```
class Example:
    def __init__(self):
        self.a = 5
        self.b = 6

obj = Example()
setattr(obj, 'c', 7)
obj.d = 8
print('obj.d =', obj.d)
```

```
obj.d равен 8
```

Сириус
IT-Колледж

```
class Example:
    __slots__ = ('a', 'b')

    def __init__(self):
        self.a = 5
        self.b = 6

obj = Example()
setattr(obj, 'c', 7)
```

```
    setattr(obj, 'c', 7)
AttributeError: 'Example' object has no attribute 'c'
```

```python
class Example:
    __slots__ = ('a', 'b')

    def __init__(self):
        self.a = 5
        self.b = 6

obj = Example()
obj.d = 8
```

```
    obj.d = 8
    ^^^^^
AttributeError: 'Example' object has no attribute 'd'
```

Сириус
IT-Колледж

```python
class Example:
    __slots__ = ('a',)

    def __init__(self):
        self.a = 5
        self.b = 6

obj = Example()
```

```
    self.b = 6
    ^^^^^^
AttributeError: 'Example' object has no attribute 'b'
```

Сириус
IT-Колледж

```python
class Example:
    __slots__ = ('a', 'b')

    def __init__(self):
        self.a = 5
        self.b = 6

obj = Example()
obj.__dict__ = {'a': 8, 'b': 9, 'c': 10}
```

```
    obj.__dict__ = {'a': 8, 'b': 9, 'c': 10}
    ^^^^^^^^^^^^
AttributeError: 'Example' object has no attribute '__dict__'
```

```python
from sys import getsizeof

class ExampleSlots:
    __slots__ = ('a','b')

class ExampleNoSlots:
    pass

slots = [ExampleSlots() for _ in range(100)]
no_slots = [ExampleNoSlots() for _ in range(100)]

print(getsizeof(slots)+sum([getsizeof(item) for item in slots]))
print(getsizeof(no_slots)+sum([getsizeof(item) for item in no_slots]))
```

```
5720
6520
```

Сириус
IT-Колледж

```python
class ChessPiece:
    def __init__(self, x=0, y=0, x_lim=7, y_lim=7):
        if x < 0 | y < 0 | x > x_lim | y > y_lim:
            return ValueError('x={0}, y={1}'.format(x, y))
        self.x, self.y = x, y
        self.x_lim, self.y_lim = x_lim, y_lim

    def up(self):
        self.y = self.y_lim if self.y == 0 else self.y - 1

    def down(self):
        self.y = 0 if self.y == self.y_lim else self.y + 1

    def left(self):
        self.x = self.x_lim if self.x == 0 else self.x - 1

    def right(self):
        self.x = 0 if self.x == self.x_lim else self.x + 1
```

```
queen = ChessPiece()
queen.right()
queen.down()
queen.left()
queen.up()
queen.down()
queen.down()
queen.down()
queen.right()
queen.right()
print(queen.x, queen.y)
```

```
class ChessPiece:
    def __init__(self, x=0, y=0, x_lim=7, y_lim=7):
        if x < 0 | y < 0 | x > x_lim | y > y_lim:
            return ValueError('x={0}, y={1}'.format(x, y))
        self.x, self.y = x, y
        self.x_lim, self.y_lim = x_lim, y_lim

    def up(self):
        self.y = self.y_lim if self.y == 0 else self.y - 1
        return self

    def down(self):
        self.y = 0 if self.y == self.y_lim else self.y + 1
        return self

    def left(self):
        self.x = self.x_lim if self.x == 0 else self.x - 1
        return self

    def right(self):
        self.x = 0 if self.x == self.x_lim else self.x + 1
        return self
```

Chaining methods (цепочки методов)

```
queen = ChessPiece()
queen.right().down().left().up().down().down().right().right()
print(queen.x, queen.y)
```

Chaining methods (цепочки методов)

```
queen = ChessPiece()
queen.right()\
     .down()\
     .left()\
     .up()\
     .down()\
     .down()\
     .right()\
     .right()
print(queen.x, queen.y)
```

Chaining methods (цепочки методов)

```
queen = ChessPiece()
(
    queen.right()
        .down()
        .left()
        .up()
        .down()
        .down()
        .right()
        .right()
)
print(queen.x, queen.y)
```

```python
class Animal:

    def __init__(self, age: int):
        self.age = age

    def run(self):
        pass

class Bear(Animal):
    type_ = 'Bear'

    def __init__(self, age: int, color: str):
        super().__init__(age)
        self.color = color

    def run(self):
        print('The bear is running')
```

```python
from abc import ABC, abstractmethod

class Animal(ABC):

    @abstractmethod
    def __init__(self, age: int):
        self.age = age

    @abstractmethod
    def run(self):
        pass

class Bear(Animal):
    type_ = 'Bear'

    def __init__(self, age: int, color: str):
        super().__init__(age)
        self.color = color

    def run(self):
        print('The bear is running')
```

```
new_animal = Animal(10)
new_animal.run()
```

```
    new_animal = Animal(10)
                 ^^^^^^^^^^
TypeError: Can't instantiate abstract class Animal with abstract methods __init__, run
```

Сириус
IT-Колледж

```python
class Person:
    ending = '18:00'
    def __init__(self, name: str, age: int):
        self.name, self.age = name, age

    @classmethod
    def go_home(cls):
        print('{0}: it\'s {1} already, time to go'.format(cls.__name__, cls.ending))

class Teacher(Person):
    ending = 'night'

    def __init__(self, name: str, age: int, dept: str):
        super().__init__(name, age)
        self.dept = dept

class Student(Person):
    ending = '16:30'

    def __init__(self, name: str, age: int, group: str):
        super().__init__(name, age)
        self.group = group
```

```python
teacher = Teacher('TAA', 50, 'Dev')
teacher.go_home()
student = Student('X', 20, '7')
student.go_home()
```

```
Teacher: it's night already, time to go
Student: it's 16:30 already, time to go
```

Сириус
IT-Колледж

```python
class Teacher(Person):
    ending = 'night'

    def __init__(self, name: str, age: int, dept: str):
        super().__init__(name, age)
        self.dept = dept
        self.work = 0

    def teach(self, students: list):
        for student in students:
            volume = randint(0, 2)
            student.knowledge += volume
            self.work += volume * 2
```

```python
class Student(Person):
    ending = '16:30'

    def __init__(self, name: str, age: int, group: str, knowledge: int = 0):
        super().__init__(name, age)
        self.group, self.knowledge = group, knowledge

class Assistant(Teacher, Student):
    ending = '18:00'

    def __init__(self, name: str, age: int, group: str, dept: str, knowledge: int = 100):
        self.name, self.age, self.group, self.dept, self.knowledge = \
            name, age, group, dept, knowledge
        self.work = 0
```

Сириус
IT-Колледж

```python
teacher = Teacher('TAA', 50, 'Dev')
teacher.go_home()
student = Student('X', 20, '7')
student.go_home()
assistant = Assistant('Y', 25, '7', 'Dev')
assistant.go_home()
assistant.teach([student])
print(student.knowledge)
```

```
Teacher: it's night already, time to go
Student: it's 16:30 already, time to go
Assistant: it's 18:00 already, time to go
1
```