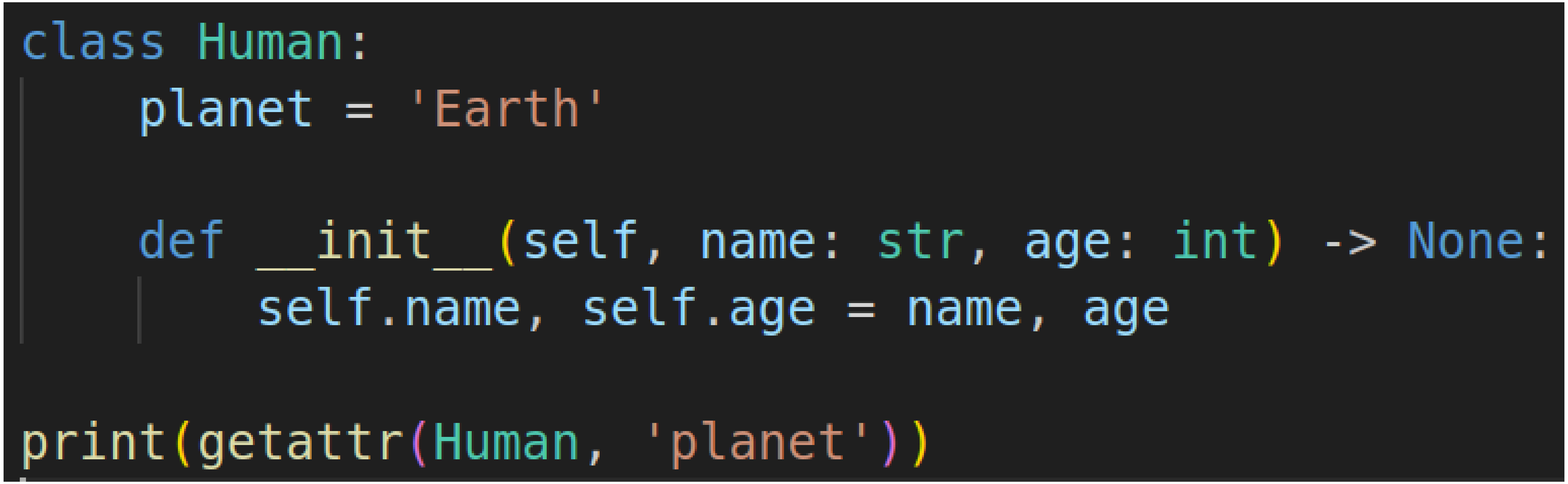
ООП 5. Type. Аттрибуты. Метаклассы.

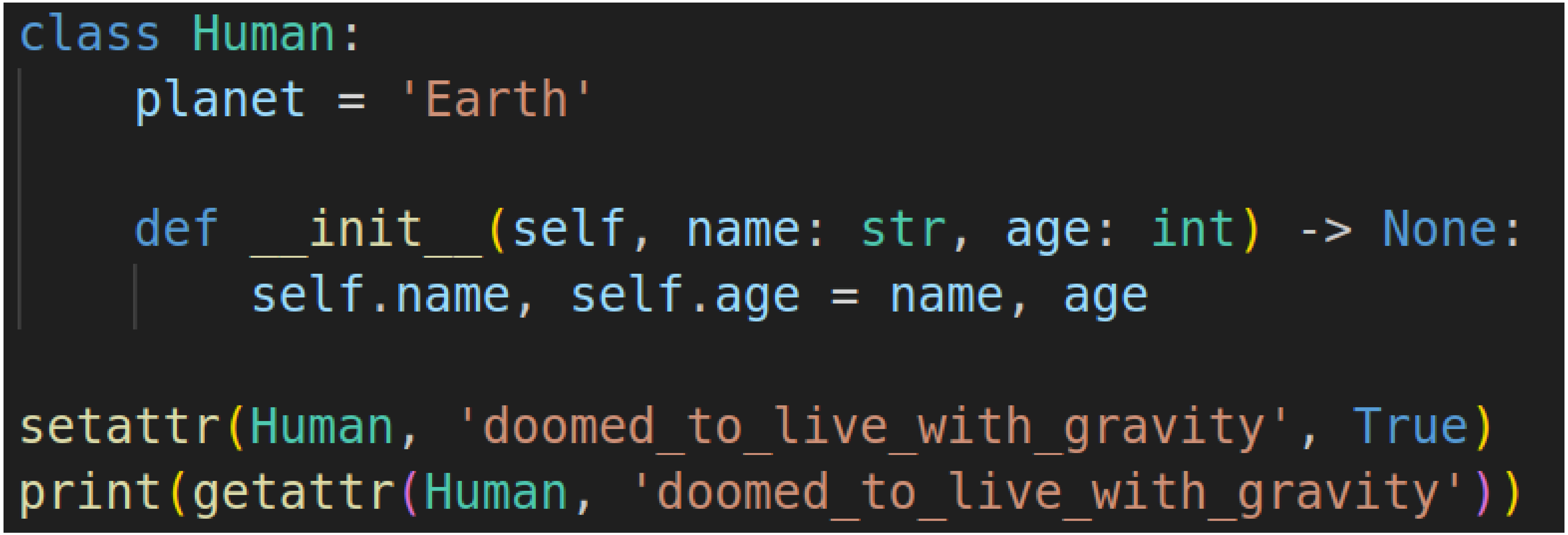
Преподаватель: Тенигин Альберт Андреевич

Сириус
IT-Колледж



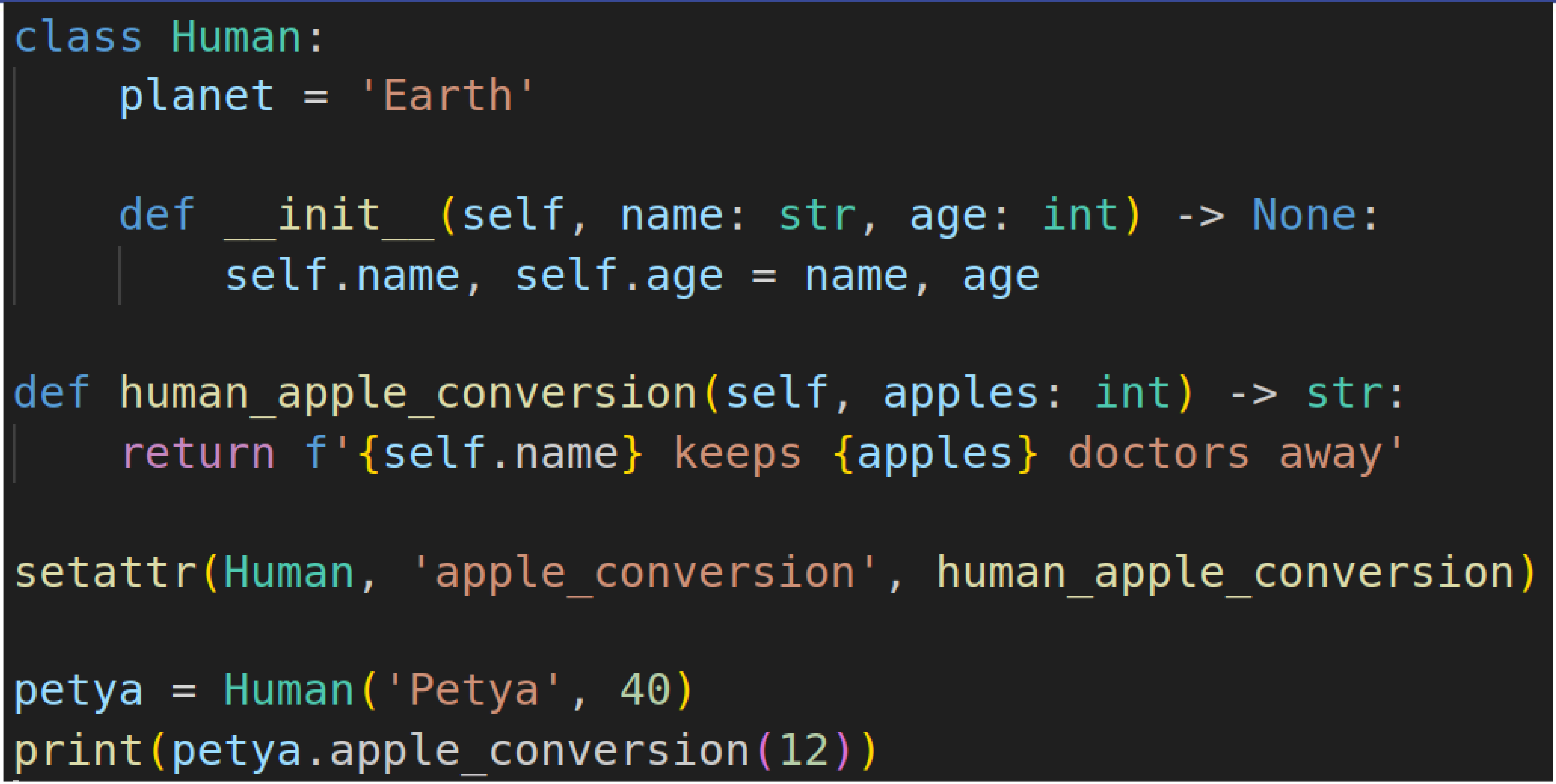
```python
class Human:
    planet = 'Earth'

    def __init__(self, name: str, age: int) -> None:
        self.name, self.age = name, age

print(getattr(Human, 'planet'))
```

Earth

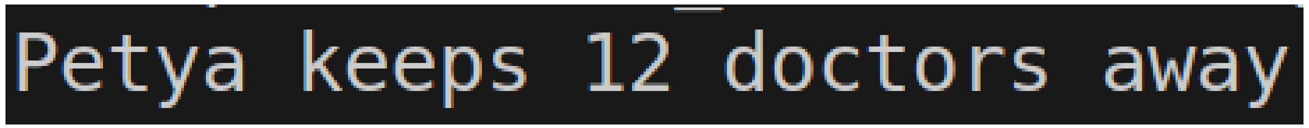

```python
class Human:
    planet = 'Earth'

    def __init__(self, name: str, age: int) -> None:
        self.name, self.age = name, age

setattr(Human, 'doomed_to_live_with_gravity', True)
print(getattr(Human, 'doomed_to_live_with_gravity'))
```

```
True
```

Сириус
IT-Колледж

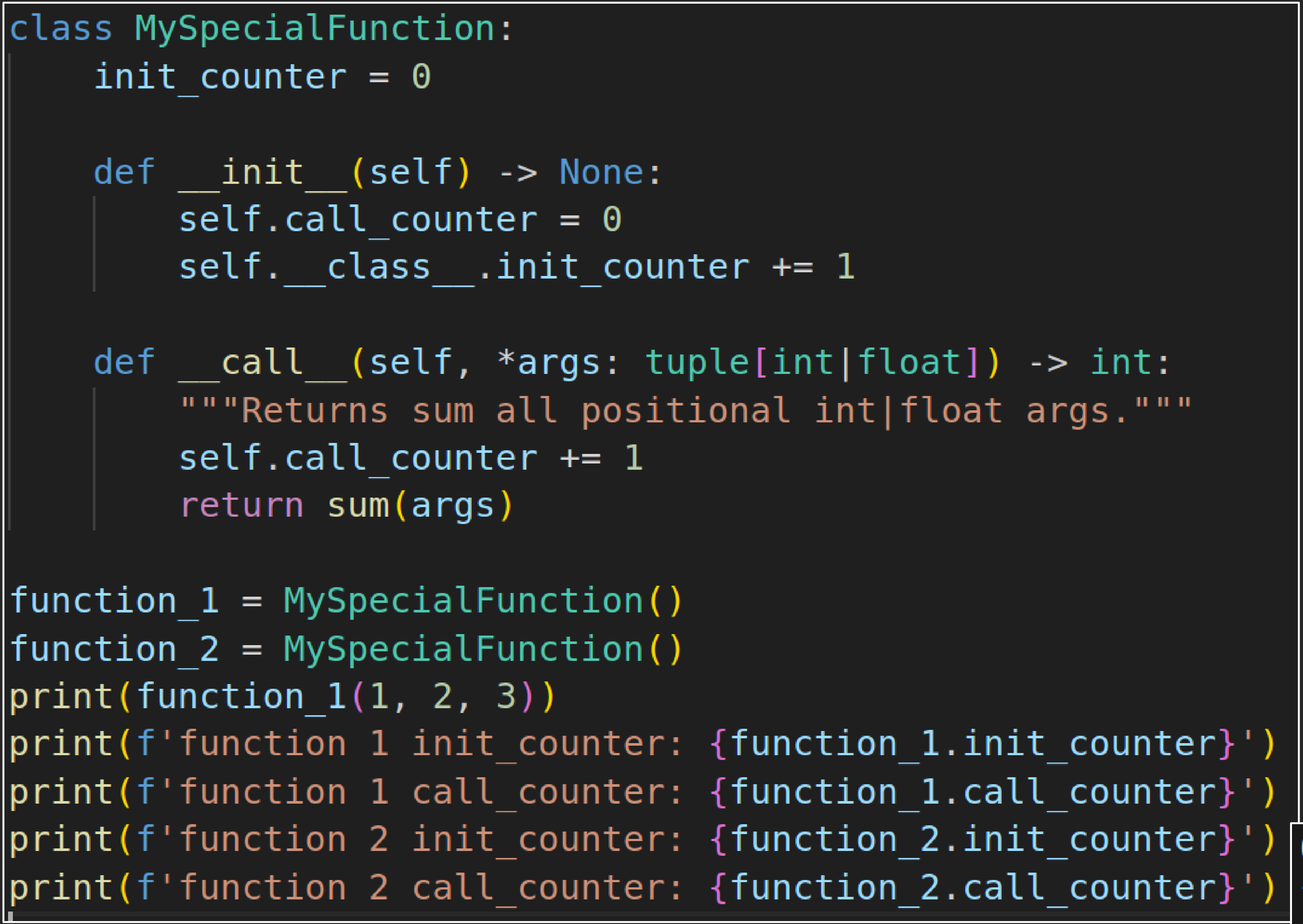


```python
class Human:
    planet = 'Earth'

    def __init__(self, name: str, age: int) -> None:
        self.name, self.age = name, age

def human_apple_conversion(self, apples: int) -> str:
    return f'{self.name} keeps {apples} doctors away'

setattr(Human, 'apple_conversion', human_apple_conversion)

petya = Human('Petya', 40)
print(petya.apple_conversion(12))
```



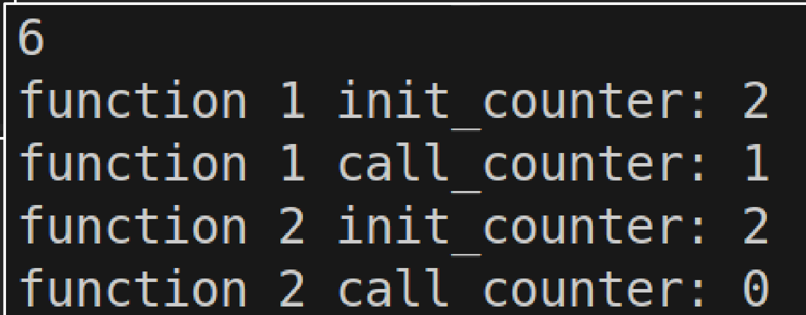
```
Petya keeps 12 doctors away
```

Сириус
IT-Колледж

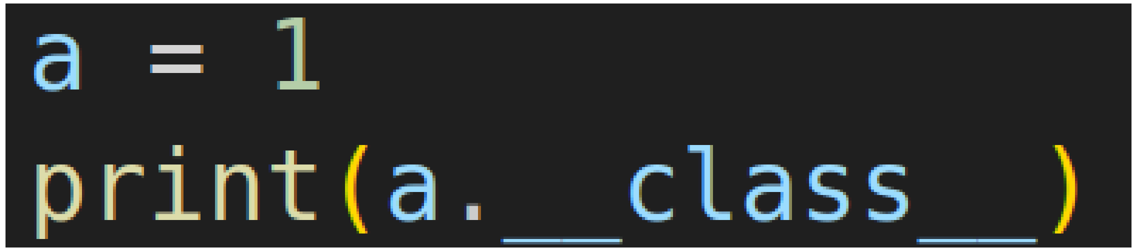


```python
class MySpecialFunction:
    init_counter = 0

    def __init__(self) -> None:
        self.call_counter = 0
        self.__class__.init_counter += 1

    def __call__(self, *args: tuple[int|float]) -> int:
        """Returns sum all positional int|float args."""
        self.call_counter += 1
        return sum(args)

function_1 = MySpecialFunction()
function_2 = MySpecialFunction()
print(function_1(1, 2, 3))
print(f'function 1 init_counter: {function_1.init_counter}')
print(f'function 1 call_counter: {function_1.call_counter}')
print(f'function 2 init_counter: {function_2.init_counter}')
print(f'function 2 call_counter: {function_2.call_counter}')
```

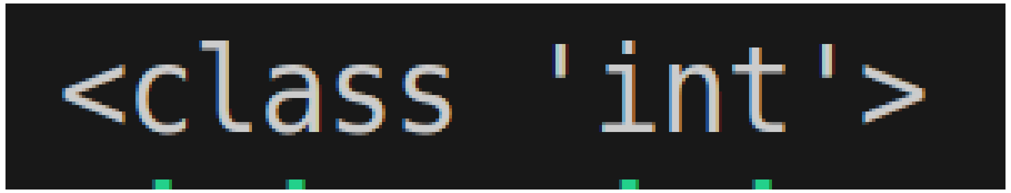
Вывод:



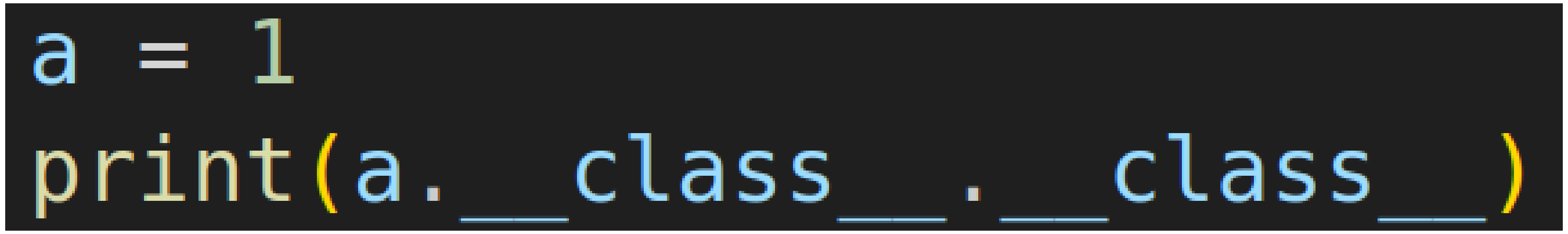
```
6
function 1 init_counter: 2
function 1 call_counter: 1
function 2 init_counter: 2
function 2 call_counter: 0
```



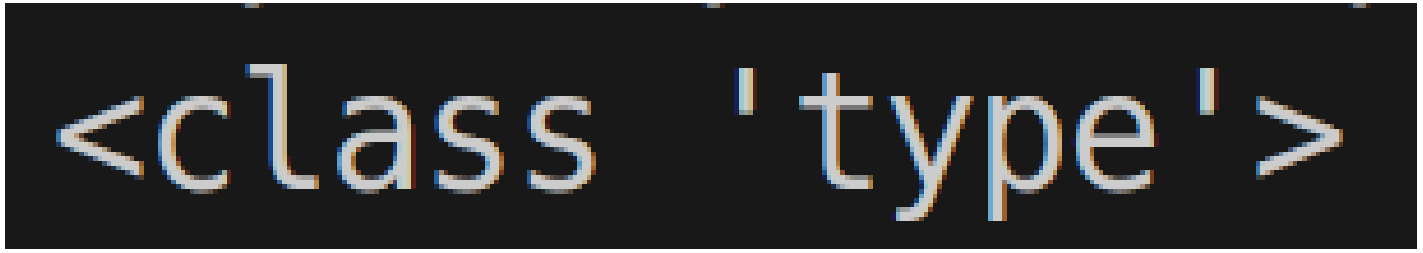
```python
a = 1
print(a.__class__)
```



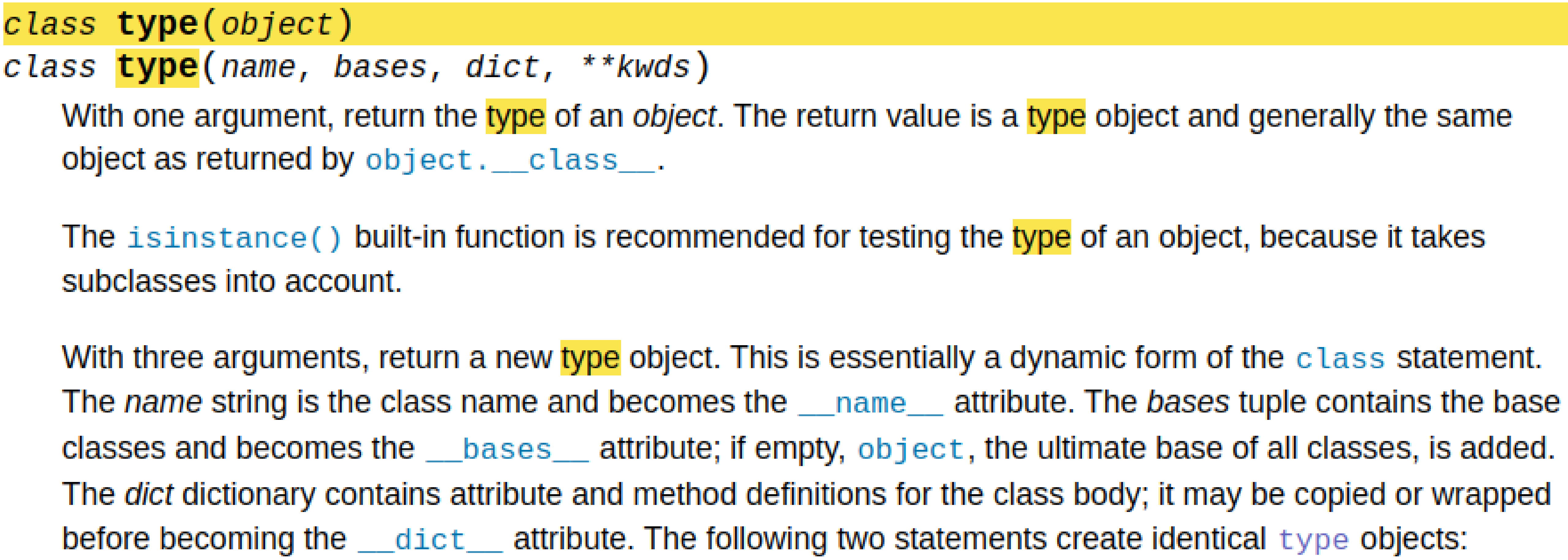
```
<class 'int'>
```



```
a = 1
print(a.__class__.__class__)
```



```
<class 'type'>
```



class **type**(*object*)

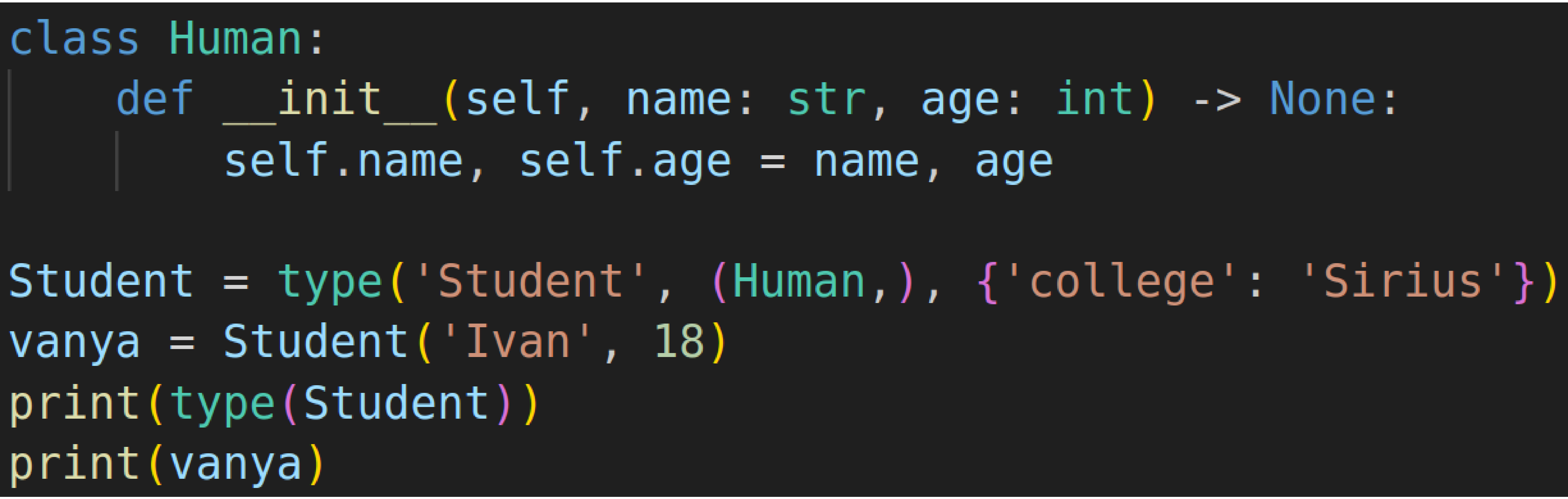
class **type**(*name, bases, dict, **kwds*)

With one argument, return the type of an *object*. The return value is a type object and generally the same object as returned by `object.__class__`.

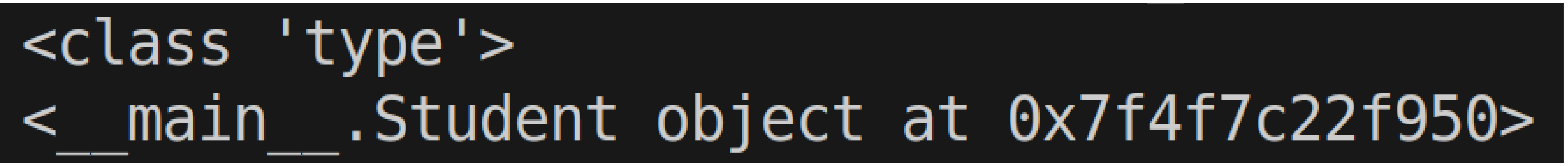
The `isinstance()` built-in function is recommended for testing the type of an object, because it takes subclasses into account.

With three arguments, return a new type object. This is essentially a dynamic form of the `class` statement. The *name* string is the class name and becomes the `__name__` attribute. The *bases* tuple contains the base classes and becomes the `__bases__` attribute; if empty, `object`, the ultimate base of all classes, is added. The *dict* dictionary contains attribute and method definitions for the class body; it may be copied or wrapped before becoming the `__dict__` attribute. The following two statements create identical `type` objects:

Сириус
IT-Колледж

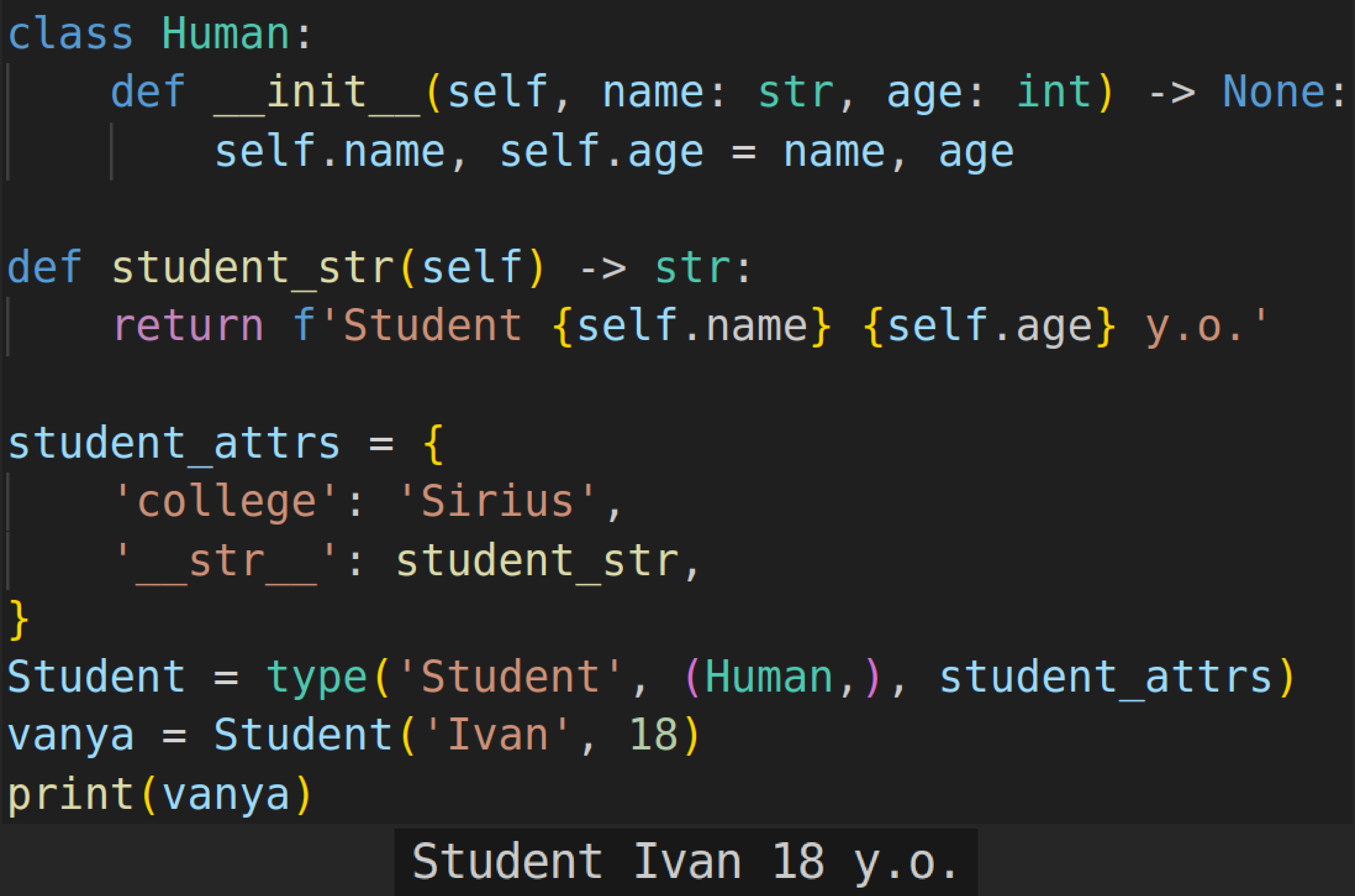


```python
class Human:
    def __init__(self, name: str, age: int) -> None:
        self.name, self.age = name, age

Student = type('Student', (Human,), {'college': 'Sirius'})
vanya = Student('Ivan', 18)
print(type(Student))
print(vanya)
```



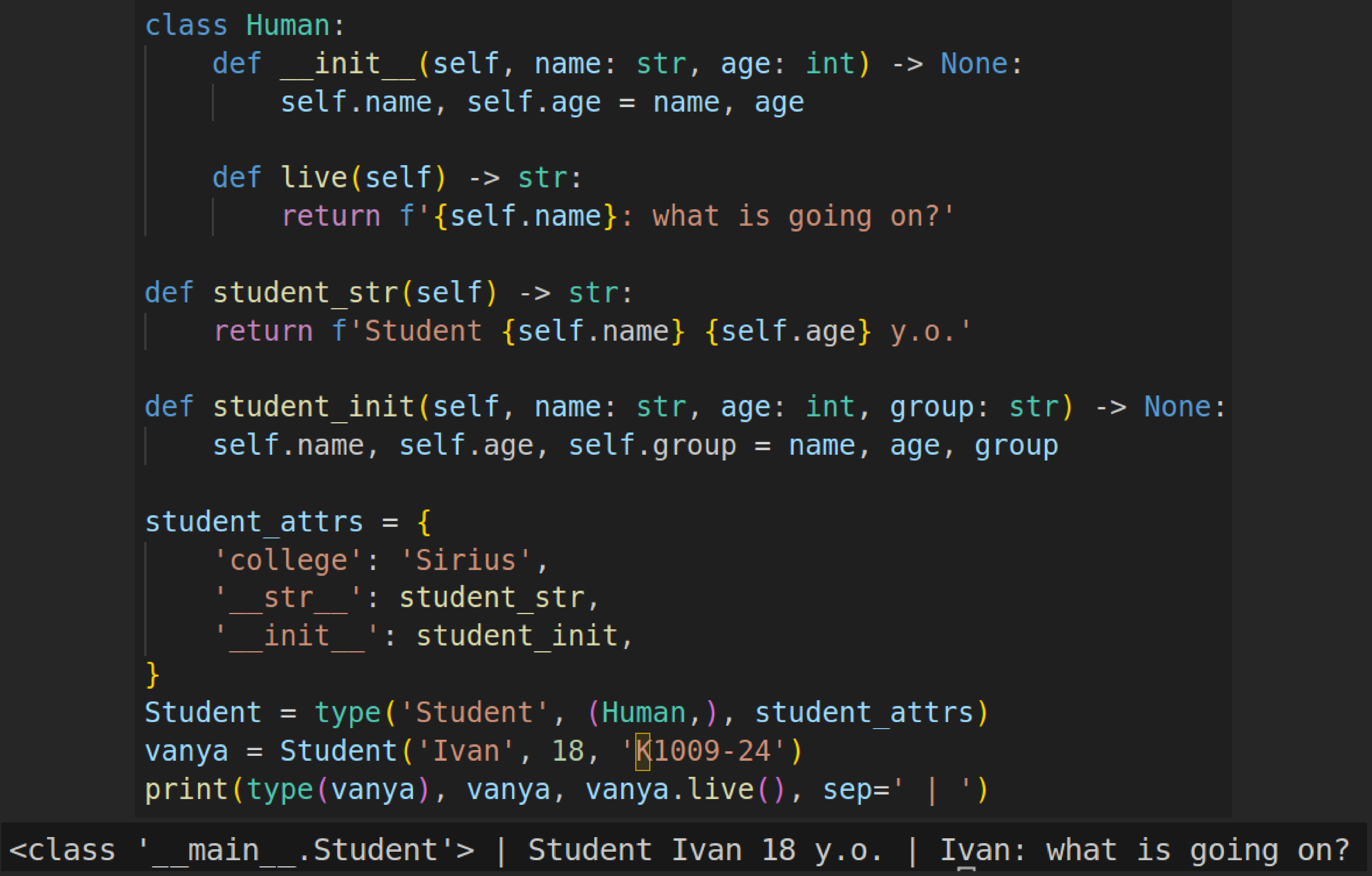
```
<class 'type'>
<__main__.Student object at 0x7f4f7c22f950>
```

Сириус
IT-Колледж



```python
class Human:
    def __init__(self, name: str, age: int) -> None:
        self.name, self.age = name, age

def student_str(self) -> str:
    return f'Student {self.name} {self.age} y.o.'

student_attrs = {
    'college': 'Sirius',
    '__str__': student_str,
}
Student = type('Student', (Human,), student_attrs)
vanya = Student('Ivan', 18)
print(vanya)
```

```
Student Ivan 18 y.o.
```



```python
class Human:
    def __init__(self, name: str, age: int) -> None:
        self.name, self.age = name, age

    def live(self) -> str:
        return f'{self.name}: what is going on?'

def student_str(self) -> str:
    return f'Student {self.name} {self.age} y.o.'

def student_init(self, name: str, age: int, group: str) -> None:
    self.name, self.age, self.group = name, age, group

student_attrs = {
    'college': 'Sirius',
    '__str__': student_str,
    '__init__': student_init,
}
Student = type('Student', (Human,), student_attrs)
vanya = Student('Ivan', 18, 'K1009-24')
print(type(vanya), vanya, vanya.live(), sep=' | ')
```

```
<class '__main__.Student'> | Student Ivan 18 y.o. | Ivan: what is going on?
```