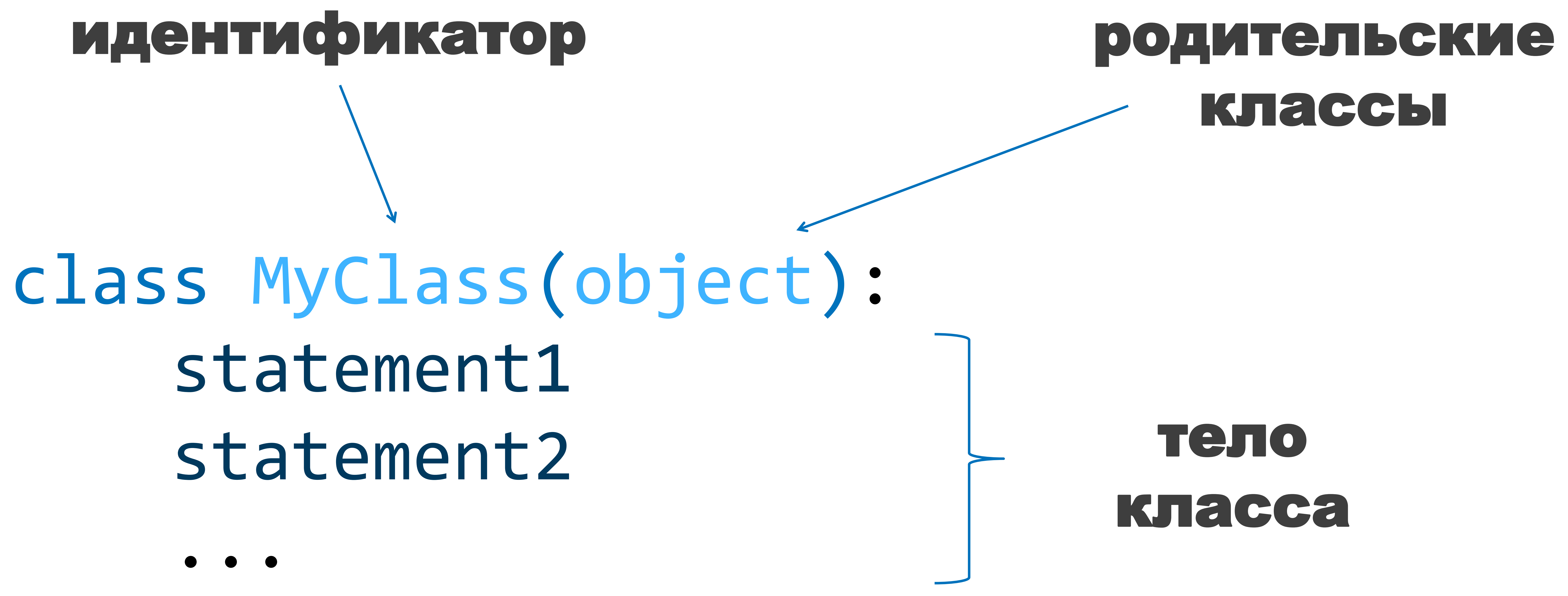
Классы

Докладчик: Евграфов Михаил

Объекты классов

Пользовательские классы



```
class MyClass(object):
    statement1
    statement2
    ...
```

Объект класса

```
class MyClass:
    pass


>>> print(
...     MyClass,
...     f"MyClass type: {type(MyClass).__name__}",
...     sep="\n",
... )
<class '__main__.MyClass'>
MyClass type: type
```

Тело класса

```
class AbsurdClass:
    num1: int = 2
    num2: int = num1 ** 2

    for i in range(num2):
        print(f"{i = };")
```

```
i = 0;
i = 1;
i = 2;
i = 3;
```

Операции с объектами класса

```
class Point2D:
    abscissa: float = .0
    ordinate: float = .0



>>> point2d_instance = Point2D()
>>> Point2D.abscissa = 42
>>> print(f"Point2D abscissa value: {Point2D.abscissa}")
>>> print(point2d_instance)
Point2D abscissa value: 42
<__main__.Point2D object at 0x0000017B59046F90>
```

Объекты класса и переменные

```
class Point2D:
    abscissa: float = .0
    ordinate: float = .0

>>> point = Point2D
>>> point_instance = point()
>>> print(point_instance)
>>> print(f"point ordinate: {point.ordinate}")
<__main__.Point2D object at 0x0000017B596B3210>
point ordinate: 0.0
```

Объекты класса и аргументы функций

```
def print_object_info(obj: object) -> None:
    print(f"object is: {id(obj)}")
    print(f"object: {obj}")
```

```
>>> print_object_info(Point2D)
object is: 1629268705952
object: <class '__main__.Point2D'>
```

Объекты класса и возвращаемые значения

```
def create_point_type() -> type:
    class Point2D:
        abscissa: float
        ordinate: float

    return Point2D

>>> point_type = create_point_type()
>>> print(point_type)
<class '__main__.create_point_type.<locals>.Point2D'>
```

__doc__

```python
class Point2D:
    """
    Точка двумерного пространства.

    Attrs:
        abscissa: абсцисса точки двумерного пространства.
        ordinate: ордината точки двумерного пространства.
    """

    abscissa: float
    ordinate: float


Point2D.__doc__
```

Экземпляры классов

__init__

```
class Point2D:
    abscissa: float
    ordinate: float

    def __init__(self, abscissa: float, ordinate: float) -> None:
        self.abscissa = abscissa
        self.ordinate = ordinate
```

```
>>> point = Point2D(3.14, -2.72)
>>> print(point.abscissa, point.ordinate)
3.14 -2.72
```

__init__

```
class Point2D:
    abscissa: float
    ordinate: float

    def __init__(self, abscissa: float, ordinate: float) -> None:
        self.abscissa = abscissa
        self.ordinate = ordinate
        return (self.abscissa, self.ordinate)


>>> point = Point2D(3.14, -2.72)
TypeError: __init__() should return None, not 'tuple'
```

Атрибуты

```
class Point2D:
    name: str = "A"
    abscissa: float =.0
    ordinate: float =.0

    def __init__(
        self,
        abscissa: float,
        ordinate: float,
    ) -> None:
        self.abscissa = abscissa
        self.ordinate = ordinate
```

Поиск атрибутов

```
>>> point1 = Point2D(3.14, -2.72)
>>> print(f"class attribute: {Point2D.abscissa}")
>>> print(f"instance attribute: {point1.abscissa}")
>>> print(f"class attribute via instance: {point1.name}")
class attribute: 0.0
instance attribute: 3.14
class attribute via instance: A
```

Атрибуты объектов класса

```
class Point2D:
    name: str = "A"
    abscissa: float =.0
    ordinate: float =.0

    def __init__(self, abscissa: float, ordinate: float) -> None:
        self.abscissa = abscissa
        self.ordinate = ordinate


>>> point1 = Point2D(3.14, -2.72)
>>> point2 = Point2D(-3.14, 2.72)
>>> point1.name, point2.name
('A', 'A')
```

Изменяемые атрибуты объектов класса

```
from typing import Any

class MyClass:
    list_: list[Any] = []

    def __init__(self, obj: Any) -> None:
        self.list_.append(obj)

>>> my_class1 = MyClass(3.14)
>>> print(my_class1.list_)
>>> my_class2 = MyClass(42)
>>> print(my_class2.list_)
[3.14]
[3.14, 42]
```

Динамическое создание атрибутов

```
class Point2D:
    abscissa: float =.0
    ordinate: float =.0

    def __init__(self, abscissa: float, ordinate: float) -> None:
        self.abscissa = abscissa
        self.ordinate = ordinate

>>> point1 = Point2D(3.14, -2.72)
>>> Point2D.applique = 42
>>> print(f"instance: {point1.applique}")
>>> print(f"class: {Point2D.applique}")
instance: 42
class: 42
```

Динамическое создание атрибутов

```
class Point2D:
    abscissa: float =.0
    ordinate: float =.0

    def __init__(self, abscissa: float, ordinate: float) -> None:
        self.abscissa = abscissa
        self.ordinate = ordinate

>>> point1 = Point2D(3.14, -2.72)
>>> point1.applique = 42
>>> print(f"instance: {point1.applique}")
>>> print(f"class: {Point2D.applique}")
AttributeError: type object 'Point2D' has no attribute 'applique'
```

Методы

```
class MyClass:
    def print_message(self) -> None:
        print("Hello world!")

>>> my_class = MyClass()
>>> my_class.print_message()
>>> MyClass.print_message(my_class)
>>> MyClass.print_message()
Hello world!
Hello world!
TypeError: ...
```

Методы

```
class MyClass:
    def print_message(self) -> None:
        print("Hello world!")


>>> my_class = MyClass()
>>> print(type(MyClass.print_message).__name__)
>>> print(type(my_class.print_message).__name__)
function
method
```

Статичные методы

```
class MyClass:
    @staticmethod
    def print_message() -> None:
        print("Hello world!")



>>> my_class = MyClass()
>>> my_class.print_message()
>>> MyClass.print_message()
Hello world!
Hello world!
```

Обращения к атрибутам в методах

```
from typing import Any

class Bag:
    bag: list[Any]

    def __init__(self) -> None:
        self.bag = []

    def add(self, obj: Any) -> None:
        self.bag.append(obj)

    def add_twice(self, obj: Any) -> None:
        self.add(obj)
        self.add(obj)
```

Обращения к атрибутам в методах

```
>>> bag = Bag()
>>> bag.add(3.14)
>>> print(bag.bag)
>>> bag.add_twice(42)
>>> print(bag.bag)
[3.14]
[3.14, 42, 42]
```

Публичные и служебные атрибуты

```
class MyClass:
    name: str
    _name: str
    __name: str

    def __init__(self, name: str) -> None:
        self.name = self._name = self.__name = name

>>> my_class = MyClass(name="name")
>>> print(my_class.name)
>>> print(my_class._name)
>>> print(my_class.__name)
name
name
AttributeError: 'MyClass' object has no attribute '__name'
```

Публичные и служебные атрибуты

```
class MyClass:
    name: str
    _name: str
    __name: str

    def __init__(self, name: str) -> None:
        self.name = self._name = self.__name = name

>>> my_class = MyClass(name="name")
>>> print(my_class.name)
>>> print(my_class._name)
>>> print(my_class._MyClass__name)
name
name
name
```

Датаклассы

dataclass

```
import dataclasses

@dataclasses.dataclass
class Point2D:
    abscissa: float = .0
    ordinate: float = .0

>>> point = Point2D(abscissa=3.14)
>>> print(point)
>>> print(Point2D(ordinate=3.14) == point)
Point2D(abscissa=3.14, ordinate=0.0)
False
```

field

```python
import dataclasses
from typing import Any

@dataclasses.dataclass
class MyClass:
    list_: list[Any] = dataclasses.field(
        default_factory=list
    )

my_class1 = MyClass()
my_class2 = MyClass()
print(my_class1.list_ is my_class2.list_)
```

asdict

```
import dataclasses

@dataclasses.dataclass
class Point2D:
    abscissa: float = .0
    ordinate: float = .0

>>> point = Point2D()
>>> print(dataclasses.asdict(point))
{'abscissa': 0.0, 'ordinate': 0.0}
```

astuple

```
import dataclasses

@dataclasses.dataclass
class Point2D:
    abscissa: float = .0
    ordinate: float = .0

>>> point = Point2D()
>>> print(dataclasses.astuple(point))
(0.0, 0.0)
```

Семинар