

ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ
УНИВЕРСИТЕТ

КАФЕДРА АВТОМАТИЗАЦИЯ ПРОИЗВОДСТВЕННЫХ ПРОЦЕССОВ

ЛЕКЦИЯ №5

Объектно-ориентированное
программирование в Python.
Наследование и полиморфизм

СОСТАВИТЕЛЬ: КАНД. ТЕХН. НАУК БЫКАДОР В.С.

Общее представление об наследовании в ООП

Наследование в ООП представляет собой механизм расширения возможностей исходного класса. При использовании механизма наследования исходный класс, то есть класс от которого будет производиться наследование называется как правило **«родительским»**, а тот класс, который будет создан в результате наследования называется **«дочерним»**.

Принципы механизма наследования

Механизм наследования позволяет расширить или изменить функциональность родительского класса придерживаясь двух важных принципов:

1. Не вносить изменения в исходный, то есть в родительский класс.
2. В дочернем классе должно быть минимум кода, это код должен касаться только новой функциональности вся та функциональность, которая должна повторять то, что есть в родительском классе автоматически будет взята из родительского класса.

Простой пример «Элементарное число»

Пусть имеется класс «Элементарное число», которое только может суммироваться с таким же «Элементарным числом» и больше ничего не может делать.

```
1
2 class ElementaryDigital:
3     def __init__(self, value):
4         self._value = value
5
6     @property
7     def value(self):
8         return self._value
9
10    @value.setter
11    def value(self, val):
12        self._value = val
13
14    def add(self, elementary_digit):
15        return ElementaryDigital(self._value + elementary_digit.value)
16
```

Одно подчёркивание!

свойство

Метод

Простой пример «Элементарное число»

```
1  from clsElementaryDigital import ElementaryDigital
2
3  a = ElementaryDigital(5)
4  b = ElementaryDigital(3)
5
6  c = a.add(b)
7
8  print(f'a = {a.value}')
9  print(f'b = {b.value}')
10 print(f'c = {c.value}')
11
```

Объявление экземпляров класса и их инициализация

Переменная «с» не объявлена как экземпляр класса ElementaryDigital, но она тоже является экземпляром данного класса.

PROBLEMS

OUTPUT

TERMINAL

DEBUG CONSOLE

```
PS C:\Users\bykvi\YandexDisk\В работе\Лекции по Python\tests> & C:/Users/bykvi/AppData/Local/Microsoft/WindowsApps/python3.10.exe "c:/Users/bykvi/YandexDisk/В работе/Лекции по Python/tests/main.py"
```

```
a = 5
```

```
b = 3
```

```
c = 8
```

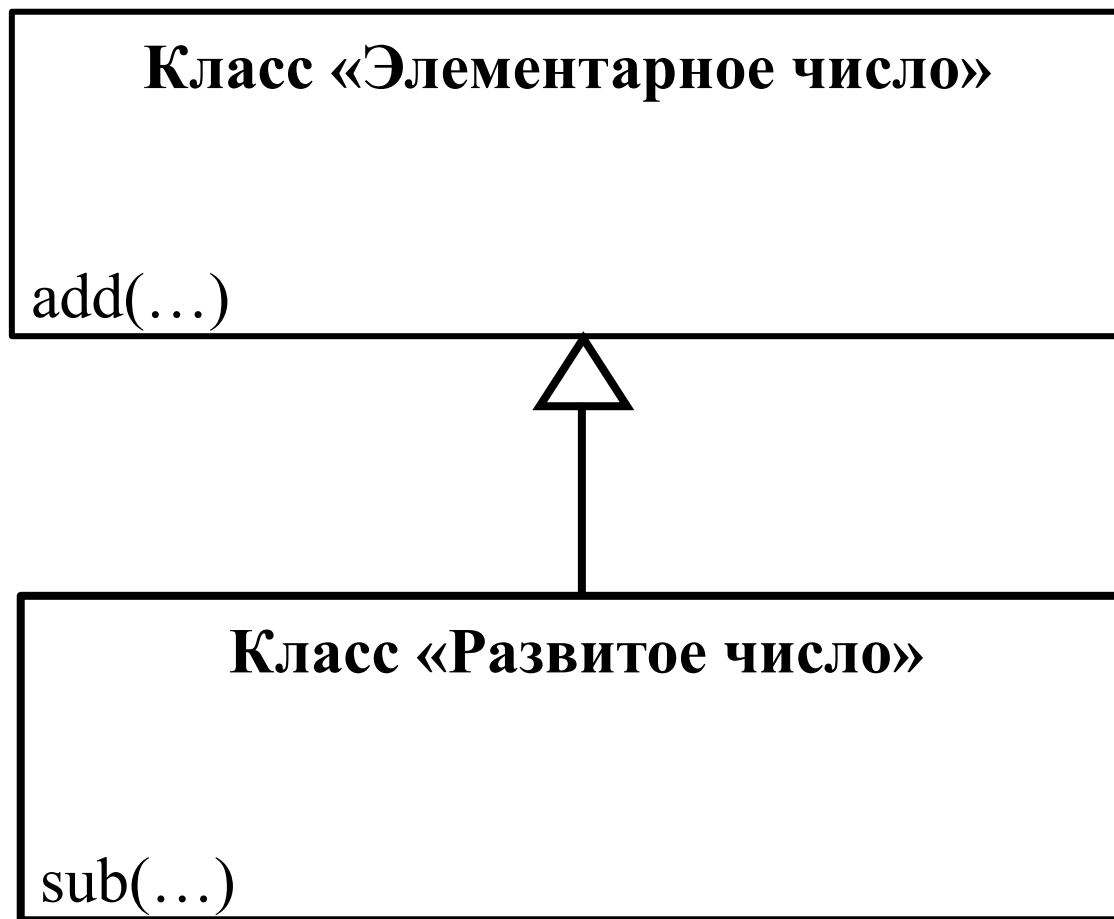
«Развитое число»

Затем нам потребовалось число более развитое, которое может не только суммироваться с себе подобными, но может ещё и вычитаться, назовем такой класс как **«Развитое число»**.

Для решения данной задачи мы можем поступить двумя способами:

- 1) просто скопировать существующий код из класса **«Элементарное число»** в класс **«Развитое число»** и добавить в класс **«Развитое число»** код с недостающим методом, но это не путь ООП.
- 2) А путь ООП это взять и унаследовать класс **«Развитое число»** от класса **«Элементарное число»**, тогда классу **«Развитое число»** будет доступны методы и свойства класса **«Элементарное число»** автоматически, дописать нужно будет только метод для вычитания.

Схема наследования



Программный код для класса «Развитое число»

```
1  from clsElementaryDigital import ElementaryDigital
2
3  class AdvancedDigital(ElementaryDigital):
4      def sub(self, advanced_digit):
5          return AdvancedDigital(self._value - advanced_digit.value)
6
```

← НАСЛЕДОВАНИЕ

Программный код для класса «Развитое число»

```
1  from clsAdvancedDigital import AdvancedDigital
2
3  a = AdvancedDigital(5)
4  b = AdvancedDigital(3)
5
6  d = a.sub(b)
7  c = a.add(b)
8
9  print(f'a = {a.value}')
10 print(f'b = {b.value}')
11 print(f'd = {d.value} sub(...)')
12 print(f'c = {c.value} add(...)')
13
```

PROBLEMS

OUTPUT

TERMINAL

DEBUG CONSOLE

```
PS C:\Users\bykvi\YandexDisk\В работе\Лекции по Python\tests>
osoft/WindowsApps/python3.10.exe "c:/Users/bykvi/YandexDisk/B
py"
```

```
a = 5
```

```
b = 3
```

```
d = 2 sub(...)
```

```
c = 8 add(...)
```

Что значит поле с одним подчёркиванием?

```
self._value = value
```

Обратите внимание на то, что поле **`_value`** записывается с одним символом подчёркивания, а не с двумя как ранее. Что это означает? Это означает, что поле является **защищённым** и такое определение поля является не что средним между частным и публичным. К защищённому полю нельзя обратиться из вне класса, но к этому полю могут обращаться как сам класс, в котором это поле определено, так и потомки этого класса.

Тогда можно свести наши знания о модификаторах доступа атрибутов в следующую таблицу:

Модификатор атрибута	Обозначение в Python	Значение атрибута	Пример атрибута
Частный (private)	Два подчёркивания	Обращаться можно только внутри того класса, в котором данный атрибут был определён.	<pre>self.__value = 5 def __add(self): pass</pre>
Защищённый (protected)	Одно подчёркивание	Обращаться можно внутри того класса, в котором данный атрибут был определён, а также к атрибуту могут обращаться наследники данного класса.	<pre>self._value = 5 def _add(self): pass</pre>
Публичный, общедоступный (public)	Нет каких-либо специальных обозначений	<p>Обращаться можно в любой части кода:</p> <p>в классе, где атрибут определён,</p> <p>в потомках класса, из внешнего кода, где есть ссылка на класс.</p>	<pre>self.value = 5</pre> <p>(для соблюдения принципа инкапсуляции лучше публичными делать не поля, а свойства)</p> <pre>def add(self): pass</pre>

Расширение конструктора родительского класса

```
1
2 class Square:
3     def __init__(self, width, height):
4         self._lst_geometry = list()
5         self._lst_geometry.append(width)
6         self._lst_geometry.append(height)
7
8     def calculate(self):
9         res = 1
10        for item in self._lst_geometry:
11            res *= item
12        return res
13
14
15
16 s = Square(3, 5)
17 print(f's = {s.calculate()}')
```

ПРОБЛЕМЫ

ВЫХОДНЫЕ ДАННЫЕ

КОНСОЛЬ ОТЛАДКИ

ТЕРМИНАЛ

```
source /Users/vitalybykador/PROJECTS/test/venv/bin/activate
● vitalybykador@Air-Vitaly test % source /Users/vitalybykador/PRO
● <r/PROJECTS/test/venv/bin/python /Users/vitalybykador/PROJECTS/
s = 15
○ (venv) vitalybykador@Air-Vitaly test %
```

Расширение конструктора родительского класса (не правильный вариант!)

```
14
15 class Cube(Square):
16     def __init__(self, width, height, deep):
17         self._lst_geometry = list()
18         self._lst_geometry.append(width)
19         self._lst_geometry.append(height)
20         self._lst_geometry.append(deep)
21
22
23 s = Square(3, 5)
24 print(f's = {s.calculate()}')
25
26 c = Cube(3, 5, 2)
27 print(f'c = {c.calculate()}')
```

ПРОБЛЕМЫ

ВЫХОДНЫЕ ДАННЫЕ

КОНСОЛЬ ОТЛАДКИ

ТЕРМИНАЛ

JU

- (venv) vityalybykador@Air-Vitaly test % /Users/vitalybykador/PROJECTS/test/for_lectures.py
s = 15
c = 30
- (venv) vityalybykador@Air-Vitaly test %

Расширение конструктора родительского класса

(правильный
вариант!)

super() – это обращение к
родительскому классу, то есть к
суперклассу.

```
14
15 class Cube(Square):
16     def __init__(self, width, height, deep):
17         super().__init__(width, height)
18         self._lst_geometry.append(deep)
19
20
21 s = Square(3, 5)
22 print(f's = {s.calculate()}')
23
24 c = Cube(3, 5, 3)
25 print(f'c = {c.calculate()}')
```

ПРОБЛЕМЫ

ВЫХОДНЫЕ ДАННЫЕ

КОНСОЛЬ ОТЛАДКИ

ТЕРМИНАЛ

```
• (venv) vitalitybykador@Air-Vitaly test % /Users/vitalybykador/PROJECTS/test/for_lectures.py
s = 15
c = 45
○ (venv) vitalitybykador@Air-Vitaly test %
```

```
2 class Square:
3     def __init__(self, width, height):
4         self._width = width
5         self._height = height
6
7     def area(self):
8         return self._width * self._height
```

Более адекватный вариант реализации классов Square и Cube

```
10
11 class Cube(Square):
12     def __init__(self, width, height, deep):
13         super().__init__(width, height)
14         self._deep = deep
15
16     def volum(self):
17         return super().area() * self._deep
18
19
20 sq = Square(3, 5)
21 print(f'sq = {sq.area()}')
22 print('')
23
24 cb = Cube(3, 5, 3)
25 print(f'cb = {cb.area()}')
26 print(f'cb = {cb.volum()}')
27
```

ПРОБЛЕМЫ ВЫХОДНЫЕ ДАННЫЕ КОНСОЛЬ ОТЛАДКИ ТЕРМИНАЛ

ECTS/test/for_lectures.py

sq = 15

cb = 15

cb = 45

o (venv) vityalybykador@Air-Vitaly test %

Общее представление о полиморфизме в ООП

Полиморфизм на греческом языке означает **«поли» - много**, а **«морфизм» - форма**, то есть полиморфизм дословно можно перевести как «многоформ».

В языках программирования с динамической типизацией, например, в языке программирования Python, полиморфизм часто встречается так как «утиная» типизация этому широко способствует.

В классическом смысле полиморфизм делят на параметрический (или истинный) и ad-hoc-полиморфизм (мнимый, специальный, специализированный).

Часто полиморфизм дают в следующем определении: один интерфейс — множество программных реализаций. Под это определение подпадает ad-hoc-полиморфизм.

Ad-hoc-полиморфизм во многих языках поддерживается по средствам перегрузки функций и методов, а в слабо типизированных языках (например, в языках с динамической типизацией поддерживающих принцип «утиной» типизации - как язык программирования Python) – по средствам (неявного) приведения типов.

Полиморфизм основанный на неявном анализе ТИПОВ

```
1  print(' ')
2
3  a = 1
4  b = 2
5  c = a + b
6  print(c)
7
10 a = 'к'
11 b = 'ю'
12 c = a + b
13 print(c)
14
15 print(' ')
16
```

PROBLEMS OUTPUT

PS C:\Users\bykvi\Microsoft/WindowsApps/python

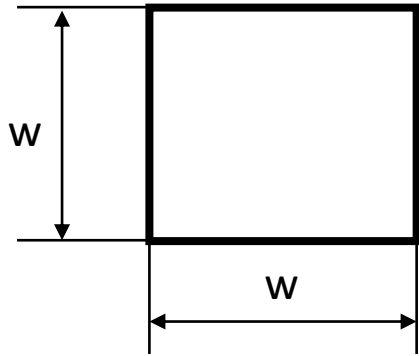
3

кю

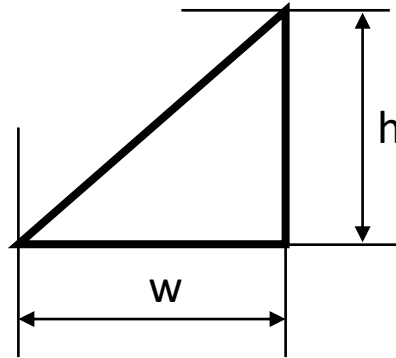
PVM анализирует тип переменных и в зависимости от этого в место общей операции «+» для **целочисленных** значений выполняет операцию **суммирования**, а для **символьных** значений операцию **конкатенации**.

Полиморфизм пользовательских типов

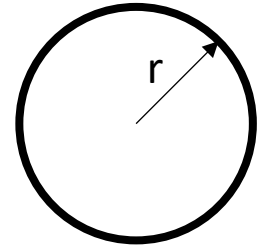
Геометрические фигуры



$$S = w^2$$

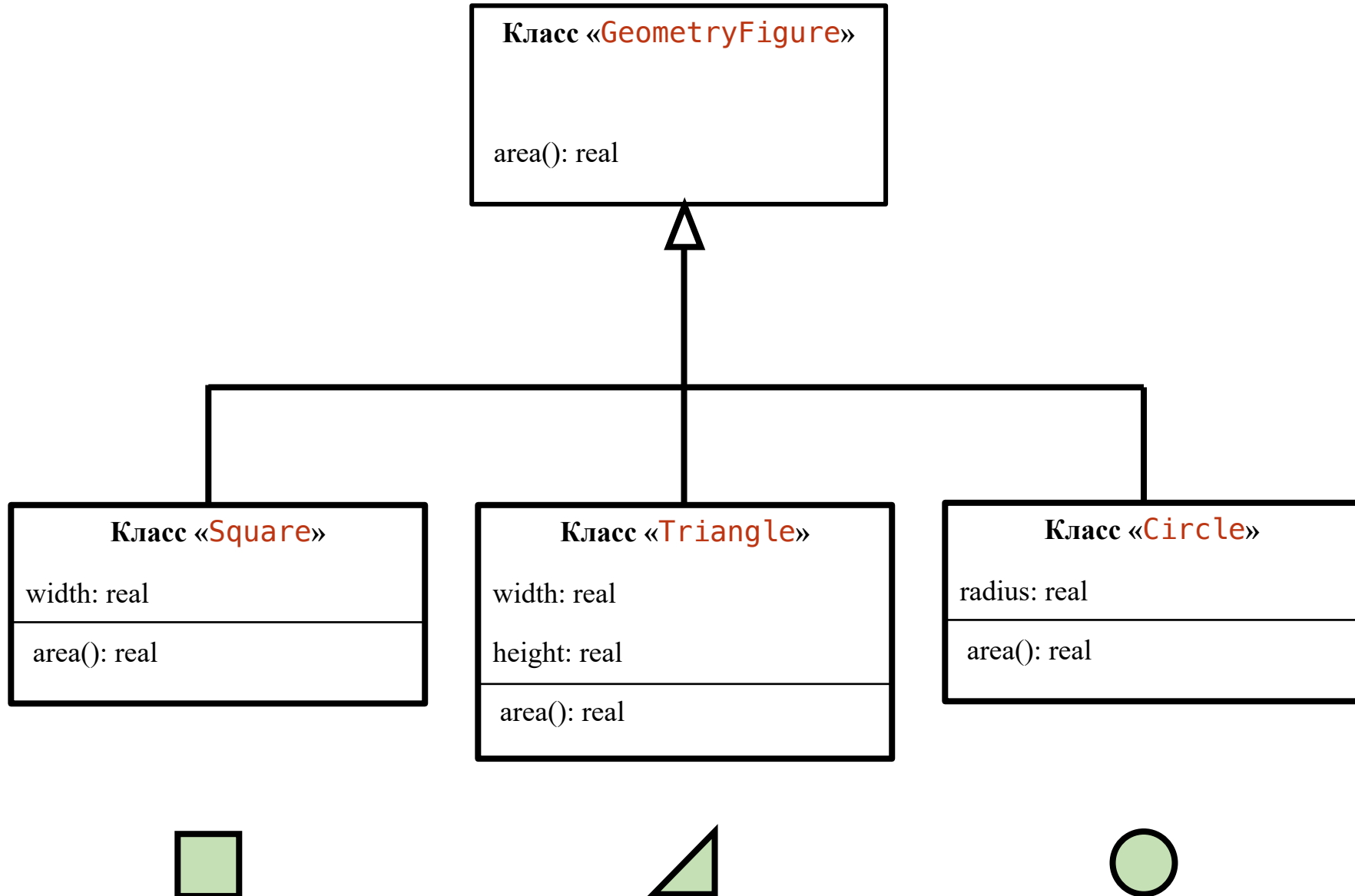


$$S = \frac{1}{2}w \cdot h$$



$$S = \pi r^2$$

Полиморфизм пользовательских типов



Полиморфизм пользовательских типов

```
4 class GeometryFigure:
5     def area(self):
6         return None
7
```

```
9 class Square(GeometryFigure):
10     def __init__(self, width):
11         if width >= 0:
12             self.__width = width
13
14     def area(self):
15         return self.__width**2
```

$$S = w^2$$

```
18 class Triangle(GeometryFigure):
19     def __init__(self, width, height):
20         if width >= 0:
21             self.__width = width
22         if height >= 0:
23             self.__height = height
24
25     def area(self):
26         return (1/2) * self.__width * self.__height
```

$$S = w \cdot h$$

```
29 class Circle(GeometryFigure):
30     def __init__(self, radius):
31         if radius >= 0:
32             self.__radius = radius
33
34     def area(self):
35         return math.pi * self.__radius**2
```

$$S = \pi r^2$$

Полиморфизм пользовательских типов

```
38     sq = Square(3)
39     print(f'sq = {sq.area()}')
40     print('')
41
42     tr = Triangle(2, 4)
43     print(f'tr = {tr.area()}')
44     print('')
45
46     cl = Circle(5)
47     print(f'cl = {cl.area()}')
48     print('')
49
```

ПРОБЛЕМЫ

ВЫХОДНЫЕ ДАННЫЕ

КОНСОЛЬ ОТ

```
• (venv) vityalybykador@Air-Vitaly test % /User
ECTS/test/for_lectures.py
sq = 9

tr = 4.0

cl = 78.53981633974483
```

Полиморфизм через стандартные dunder- методы

`__init__`

`__abs__`
`__add__`
`__and__`
`__bool__`
`__ceil__`
`__class__`
`__delattr__`
`__dir__`
`__divmod__`
`__doc__`
`__eq__`
`__float__`
`__floor__`
`__floordiv__`
`__format__`
`__ge__`
`__getattr__`
`__getnewargs__`
`__gt__`
`__hash__`
`__index__`

`__rmod__`
`__rmul__`
`__ror__`
`__round__`
`__rpow__`
`__rrshift__`
`__rshift__`
`__rsub__`
`__rtruediv__`
`__rxor__`
`__setattr__`
`__sizeof__`

`__str__`

`__sub__`
`__subclasshook__`
`__truediv__`
`__trunc__`
`__xor__`

`__init_subclass__`
`__int__`
`__invert__`
`__le__`
`__lshift__`
`__lt__`
`__mod__`
`__mul__`
`__ne__`
`__neg__`
`__new__`
`__or__`
`__pos__`
`__pow__`
`__radd__`
`__rand__`
`__rdivmod__`
`__reduce__`
`__reduce_ex__`
`__repr__`
`__rfloordiv__`
`__rlshift__`

Полиморфизм через стандартные dunder-методы

`__str__`

```
print(f'sq = {sq.area()}')
```

```
print(sq)
```



Полиморфизм через стандартные dunder-методы

```
class Square(GeometryFigure):
    def __init__(self, width):
        if width >= 0:
            self.__width = width

    def area(self):
        return self.__width**2

    def __str__(self):
        return f'square area = {self.area()}'
```

```
class Triangle(GeometryFigure):
    def __init__(self, width, height):
        if width >= 0:
            self.__width = width
        if height >= 0:
            self.__height = height

    def area(self):
        return (1/2) * self.__width * self.__height

    def __str__(self):
        return f'triangle area = {self.area()}'
```

```
class Circle(GeometryFigure):
    def __init__(self, radius):
        if radius >= 0:
            self.__radius = radius

    def area(self):
        return math.pi * self.__radius**2

    def __str__(self):
        return f'circle area = {self.area()}'
```


Полиморфизм через стандартные dunder-методы

Было

```
38 sq = Square(3)
39 print(f'sq = {sq.area()}')
40 print('')
41
42 tr = Triangle(2, 4)
43 print(f'tr = {tr.area()}')
44 print('')
45
46 cl = Circle(5)
47 print(f'cl = {cl.area()}')
48 print('')
49
```

ПРОБЛЕМЫ ВЫХОДНЫЕ ДАННЫЕ КОНСОЛЬ ОТ

```
• (venv) vitalitybykador@Air-Vitaly test % /User
ECTS/test/for_lectures.py
sq = 9

tr = 4.0

cl = 78.53981633974483
```

Стало

```
45 sq = Square(3)
46 print(sq)
47 print('')
48
49 tr = Triangle(2, 4)
50 print(tr)
51 print('')
52
53 cl = Circle(5)
54 print(cl)
55 print('')
56
```

ПРОБЛЕМЫ ВЫХОДНЫЕ ДАННЫЕ КОНСОЛЬ ОТ

```
• (venv) vitalitybykador@Air-Vitaly test % /User
square area = 9

triangle area = 4.0

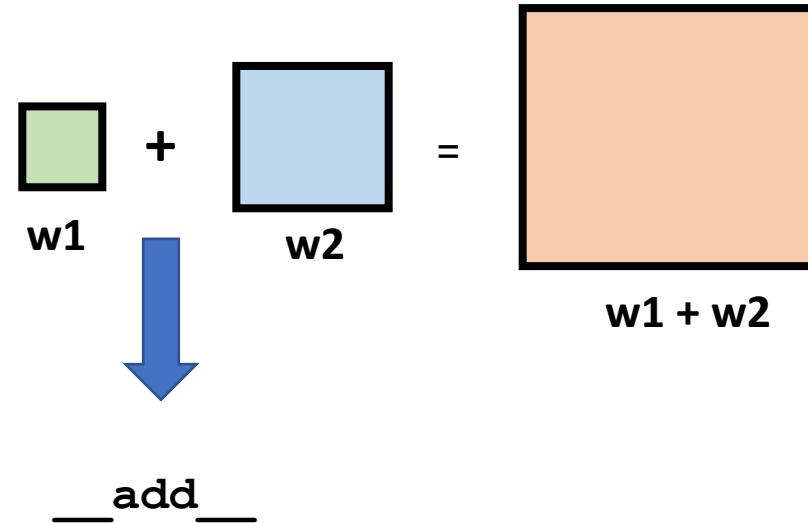
circle area = 78.53981633974483

○ (venv) vitalitybykador@Air-Vitaly test % █
```

Полиморфизм через стандартные dunder-методы

Определим операцию суммирования квадратов

```
sq1 = Square(3)
sq2 = Square(2)
sq3 = sq1 + sq2
```



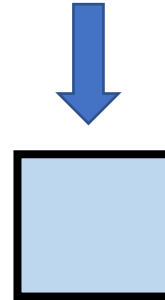
```
Traceback (most recent call last):
  File "/Users/vitalybykador/PROJECTS/test/for_lectures.py", line 62, in <module>
    sq3 = sq1 + sq2
TypeError: unsupported operand type(s) for +: 'Square' and 'Square'
(venv) vitalybykador@Air-Vitaly test %
```

Полиморфизм через стандартные dunder-методы

```
9 class Square(GeometryFigure):
10     def __init__(self, width):
11         if width >= 0:
12             self.__width = width
13
14     @property
15     def width(self):
16         return self.__width
17
18     def area(self):
19         return self.width**2
20
21     def __str__(self):
22         return f'square area = {self.area()}'
23
24     def __add__(self, obj):
25         return Square(self.width + obj.width)
```

Полиморфизм через стандартные dunder-методы

```
def __add__(self, obj):  
    return Square(self.width + obj.width)
```



Полиморфизм через стандартные dunder-методы

```
52  sq1 = Square(3)
53  print(f'sq1.width = {sq1.width}')
54  print(sq1)
55  print('')
56
57  sq2 = Square(2)
58  print(f'sq2.width = {sq2.width}')
59  print(sq2)
60  print('')
61
```

ПРОБЛЕМЫ

ВЫХОДНЫЕ ДАННЫЕ

КОНСОЛЬ ОТЛАДКИ

```
(venv) vitalitybykador@Air-Vitaly test % /Users/vitaly/
sq1.width = 3
square area = 9

sq2.width = 2
square area = 4
```

Полиморфизм через стандартные dunder-методы

```
52 sq1 = Square(3)
53 print(f'sq1.width = {sq1.width}')
54 print(sq1)
55 print('')
56
57 sq2 = Square(2)
58 print(f'sq2.width = {sq2.width}')
59 print(sq2)
60 print('')
61
62 sq3 = sq1 + sq2
63 print(f'sq3.width = {sq3.width}')
64 print(sq3)
65 print('')
66
```

ПРОБЛЕМЫ

ВЫХОДНЫЕ ДАННЫЕ

КОНСОЛЬ ОТЛАДКИ

```
(venv) vityalybykador@Air-Vitaly test % /Users/vitaly
sq1.width = 3
square area = 9

sq2.width = 2
square area = 4

sq3.width = 5
square area = 25
```