

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
ИНСТИТУТ ЦИФРОВОГО РАЗВИТИЯ**

**Отчет о лабораторной работе №16 по дисциплине основы программной
инженерии**

Выполнил:

Выходцев Егор Дмитриевич,
2 курс, группа ПИЖ-б-о-20-1,

Проверил:

Доцент кафедры инфокоммуникаций,
Воронкин Р.А.

Ставрополь, 2022 г

1. Примеры из методических указаний

```
e1.py x
1  ▶  #!/usr/bin/env python3
2      # -*- coding: utf-8 -*-
3
4  ⚙  class Figure:
5  ⚙      def __init__(self, color):
6      |         self.__color = color
7
8      @property
9      def color(self):
10     |         return self.__color
11
12     @color.setter
13     def color(self, c):
14     |         self.__color = c
15
16
17  ⚙  class Rectangle(Figure):
18  ⚙      def __init__(self, width, height, color):
19      |         super().__init__(color)
20      |         self.__width = width
21      |         self.__height = height
22
23     @property
24     def width(self):
25     |         return self.__width
26
27     @width.setter
28     def width(self, w):
29     |         if w > 0:
30     |             self.__width = w
31     |         else:
32     |             raise ValueError
33
34     @property
35     def height(self):
36     |         return self.__height
37
38     @height.setter
39     def height(self, h):
40
41  if __name__ == '__main__':
```

```

34         @property
35         def height(self):
36             return self.__height
37
38         @height.setter
39         def height(self, h):
40             if h > 0:
41                 self.__height = h
42             else:
43                 raise ValueError
44
45         def area(self):
46             return self.__width * self.__height
47
48
49     if __name__ == '__main__':
50         rect = Rectangle(10, 20, "green")
51         print(rect.width,
52               rect.height,
53               rect.color
54               )
55         rect.color = "red"
56         print(rect.color)
57

```

```

e1 x
C:\Users\Evil\PycharmProjects\LR#16\
10 20 green
red
Process finished with exit code 0

```

```
e1.py x e2.py x
4
5 class Figure:
6     def __init__(self, color):
7         self.__color = color
8
9     @property
10    def color(self):
11        return self.__color
12
13    @color.setter
14    def color(self, c):
15        self.__color = c
16
17    def info(self):
18        print("Figure")
19        print("Color: " + self.__color)
20
21
22    class Rectangle(Figure):
23        def __init__(self, width, height, color):
24            super().__init__(color)
25            self.__width = width
26            self.__height = height
27
28        @property
29        def width(self):
30            return self.__width
31
32        @width.setter
33        def width(self, w):
34            if w > 0:
35                self.__width = w
36            else:
37                raise ValueError
38
39        @property
40        def height(self):
41            return self.__height
42
```

```
e1.py x e2.py x
31
32     @width.setter
33     def width(self, w):
34         if w > 0:
35             self.__width = w
36         else:
37             raise ValueError
38
39     @property
40     def height(self):
41         return self.__height
42
43     @height.setter
44     def height(self, h):
45         if h > 0:
46             self.__height = h
47         else:
48             raise ValueError
49
50     def area(self):
51         return self.__width * self.__height
52
53     def info(self):
54         print("Rectangle")
55         print("Color: " + self.color)
56         print("Width: " + str(self.width))
57         print("Height: " + str(self.height))
58         print("Area: " + str(self.area()))
59
60
61     if __name__ == '__main__':
62         fig = Figure("orange")
63         fig.info()
64         rect = Rectangle(10, 20, "green")
65         rect.info()
66
```

```
e2 x
C:\Users\Evil\PycharmProjects\LR#16\
Figure
Color: orange
Rectangle
Color: green
Width: 10
Height: 20
Area: 200

Process finished with exit code 0
```

```
e3.py x
1 ▶ #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4
5 ⚙ class Table:
6     def __init__(self, l, w, h):
7         self.length = l
8         self.width = w
9         self.height = h
10
11
12 class DeskTable(Table):
13     def square(self):
14         return self.width * self.length
15
16
17 ▶ if __name__ == '__main__':
18     t1 = Table(1.5, 1.8, 0.75)
19     t2 = DeskTable(0.8, 0.6, 0.7)
20     print(t2.square())
21
```

```
e3 x
C:\Users\Evil\PycharmProjects\LR#1
0.48

Process finished with exit code 0
```

```
e3.py x e4.py x
1 ▶ #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 ⚙ class Table:
5 ⚙     def __init__(self, l, w, h):
6         self.length = l
7         self.width = w
8         self.height = h
9
10
11 class KitchenTable(Table):
12     def __init__(self, l, w, h, p):
13         Table.__init__(self, l, w, h)
14         self.places = p
15
16
17 ▶ if __name__ == '__main__':
18     t4 = KitchenTable(1.5, 2, 0.75, 6)
19
```

```
e3.py x e4.py x ex1.py x
1  ▶  #!/usr/bin/env python3
2      # -*- coding: utf-8 -*-
3
4
5  class Rational:
6      def __init__(self, a=0, b=1):
7          a = int(a)
8          b = int(b)
9          if b == 0:
10             raise ValueError()
11             self.__numerator = abs(a)
12             self.__denominator = abs(b)
13             self.__reduce()
14             # Сокращение дроби
15
16     def __reduce(self):
17         # Функция для нахождения наибольшего общего делителя
18         def gcd(a, b):
19             if a == 0:
20                 return b
21             elif b == 0:
22                 return a
23             elif a >= b:
24                 return gcd(a % b, b)
25             else:
26                 return gcd(a, b % a)
27
28         c = gcd(self.__numerator, self.__denominator)
29         self.__numerator //= c
30         self.__denominator //= c
31
32     @property
33     def numerator(self):
34         return self.__numerator
35
36     @property
37     def denominator(self):
38         return self.__denominator
39
```



```
e3.py x e4.py x ex1.py x
40 # Прочитать значение дроби с клавиатуры. Дробь вводится
41 # как a/b.
42 def read(self, prompt=None):
43     line = input() if prompt is None else input(prompt)
44     parts = list(map(int, line.split('/', maxsplit=1)))
45     if parts[1] == 0:
46         raise ValueError()
47     self.__numerator = abs(parts[0])
48     self.__denominator = abs(parts[1])
49     self.__reduce()
50
51 # Вывести дробь на экран
52 def display(self):
53     print(f"{self.__numerator}/{self.__denominator}")
54
55 # Сложение обыкновенных дробей.
56 def add(self, rhs):
57     if isinstance(rhs, Rational):
58         a = self.numerator * rhs.denominator + \
59             self.denominator * rhs.numerator
60         b = self.denominator * rhs.denominator
61         return Rational(a, b)
62     else:
63         raise ValueError()
64
65 # Вычитание обыкновенных дробей.
66 def sub(self, rhs):
67     if isinstance(rhs, Rational):
68         a = self.numerator * rhs.denominator - \
69             self.denominator * rhs.numerator
70         b = self.denominator * rhs.denominator
71         return Rational(a, b)
72     else:
73         raise ValueError()
74
75 # Умножение обыкновенных дробей.
76 def mul(self, rhs):
77     if isinstance(rhs, Rational):
78         a = self.numerator * rhs.numerator
```

```
e3.py × e4.py × ex1.py ×
79         b = self.denominator * rhs.denominator
80         return Rational(a, b)
81     else:
82         raise ValueError()
83
84     # Деление обыкновенных дробей.
85     def div(self, rhs):
86         if isinstance(rhs, Rational):
87             a = self.numerator * rhs.denominator
88             b = self.denominator * rhs.numerator
89             return Rational(a, b)
90         else:
91             raise ValueError()
92
93     # Отношение обыкновенных дробей.
94     def equals(self, rhs):
95         if isinstance(rhs, Rational):
96             return (self.numerator == rhs.numerator) and \
97                 (self.denominator == rhs.denominator)
98         else:
99             return False
100
101     def greater(self, rhs):
102         if isinstance(rhs, Rational):
103             v1 = self.numerator / self.denominator
104             v2 = rhs.numerator / rhs.denominator
105             return v1 > v2
106         else:
107             return False
108
109     def less(self, rhs):
110         if isinstance(rhs, Rational):
111             v1 = self.numerator / self.denominator
112             v2 = rhs.numerator / rhs.denominator
113             return v1 < v2
114         else:
115             return False
116
117
```

```
118 ▶ if __name__ == '__main__':  
119     r1 = Rational(3, 4)  
120     r1.display()  
121     r2 = Rational()  
122     r2.read("Введите обыкновенную дробь: ")  
123     r2.display()  
124     r3 = r2.add(r1)  
125     r3.display()  
126     r4 = r2.sub(r1)  
127     r4.display()  
128     r5 = r2.mul(r1)  
129     r5.display()  
130     r6 = r2.div(r1)  
131     r6.display()  
132
```

```
ex1 ×  
C:\Users\Evil\PycharmProjects\LR#16\venv  
3/4  
Введите обыкновенную дробь: 4/51  
4/51  
169/204  
137/204  
1/17  
16/153  
  
Process finished with exit code 0
```

```
e4.py x ex2.py x
1  ▶  1  #!/usr/bin/env python3
2      2  # -*- coding: utf-8 -*-
3
4      3  # Python program showing
5      4  # abstract base class work
6      5  from abc import ABC, abstractmethod
7
8      6
9      7  class Polygon(ABC):
10         8  @abstractmethod
11         9  def noofsides(self):
12             10  pass
13
14         11
15         12  class Triangle(Polygon):
16             13  # overriding abstract method
17             14  def noofsides(self):
18                 15  print("I have 3 sides")
19
20             16
21             17  class Pentagon(Polygon):
22                 18  # overriding abstract method
23                 19  def noofsides(self):
24                     20  print("I have 5 sides")
25
26                 21
27                 22  class Hexagon(Polygon):
28                     23  # overriding abstract method
29                     24  def noofsides(self):
30                         25  print("I have 6 sides")
31
32                     26
33                     27  class Quadrilateral(Polygon):
34                         28  # overriding abstract method
35                         29  def noofsides(self):
36                             30  print("I have 4 sides")
37
38                         31
39         32  # Driver code
40
41         33  Quadrilateral
```

```
39     # Driver code
40     if __name__ == '__main__':
41         R = Triangle()
42         R.noofsides()
43         K = Quadrilateral()
44         K.noofsides()
45         R = Pentagon()
46         R.noofsides()
47         K = Hexagon()
48         K.noofsides()
49
```

ex2 x

C:\Users\Evil\PycharmProjects\LR#1

I have 3 sides

I have 4 sides

I have 5 sides

I have 6 sides

Process finished with exit code 0

```
e4.py x ex3.py x
1  ▶  1  #!/usr/bin/env python3
2      2      # -*- coding: utf-8 -*-
3
4
5      5      # Python program showing
6      6      # abstract base class work
7      7      from abc import ABC
8
9
10     10     class Animal(ABC):
11         11         def move(self):
12             12             pass
13
14
15     15     class Human(Animal):
16         16         def move(self):
17             17             print("I can walk and run")
18
19
20     20     class Snake(Animal):
21         21         def move(self):
22             22             print("I can crawl")
23
24
25     25     class Dog(Animal):
26         26         def move(self):
27             27             print("I can bark")
28
29
30     30     class Lion(Animal):
31         31         def move(self):
32             32             print("I can roar")
33
34
35     35     if __name__ == '__main__':
36         36         # Driver code
37         37         R = Human()
38         38         R.move()
39         39         K = Snake()
39
if __name__ == '__main__'
```

```
ex3 x
C:\Users\Evil\PycharmProjects\LR#16
I can walk and run
I can crawl
I can bark
I can roar
Process finished with exit code 0
```

```
e4.py x ex3.py x e5.py x
1 ▶ #!/usr/bin/env python3
2   # -*- coding: utf-8 -*-
3
4
5   # Python program showing
6   # implementation of abstract
7   # class through subclassing
8   import abc
9
10
11  class parent:
12      def geeks(self):
13          pass
14
15
16  class child(parent):
17      def geeks(self):
18          print("child class")
19
20
21  if __name__ == '__main__':
22      print(issubclass(child, parent))
23      print(isinstance(child(), parent))
24
```

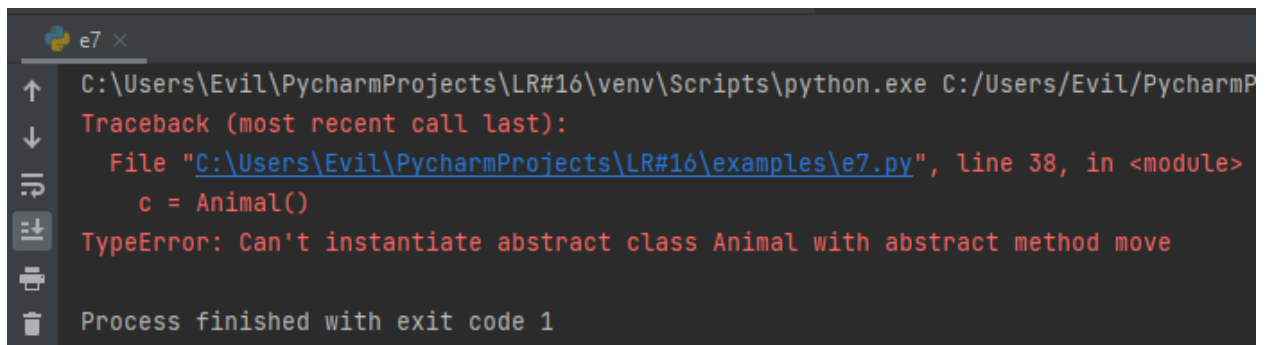
```
e5 x
C:\Users\Evil\PycharmProjects\LR#1
True
True
Process finished with exit code 0
```

```
e4.py x e6.py x
1 ▶ #!/usr/bin/env python3
2   # -*- coding: utf-8 -*-
3
4
5   # Python program invoking a
6   # method using super()
7   from abc import ABC
8
9
10  ⚙️ class R(ABC):
11  ⚙️     def rk(self):
12      |         print("Abstract Base Class")
13
14
15      class K(R):
16  ⚙️         def rk(self):
17      |             super().rk()
18      |             print("subclass")
19
20
21  ▶ if __name__ == '__main__':
22      |     r = K()
23      |     ⚡ r.rk()
24
```

```
e6 x
C:\Users\Evil\PycharmProjects\LR#1
Abstract Base Class
subclass
Process finished with exit code 0
```



```
e4.py x e6.py x e7.py x
1 ▶ #!/usr/bin/env python3
2   # -*- coding: utf-8 -*-
3
4
5   # Python program showing
6   # abstract class cannot
7   # be an instantiation
8   from abc import ABC, abstractmethod
9
10
11  ⚡ class Animal(ABC):
12      @abstractmethod
13  ⚡  def move(self):
14      pass
15
16
17  class Human(Animal):
18  ⚡  def move(self):
19      print("I can walk and run")
20
21
22  class Snake(Animal):
23  ⚡  def move(self):
24      print("I can crawl")
25
26
27  class Dog(Animal):
28  ⚡  def move(self):
29      print("I can bark")
30
31
32  class Lion(Animal):
33  ⚡  def move(self):
34      print("I can roar")
35
36
37  ▶ if __name__ == '__main__':
38      c = Animal()
39
```

A screenshot of a terminal window with a dark background. The window title is 'e7 x'. The command line shows the execution of a Python script: 'C:\Users\Evil\PycharmProjects\LR#16\venv\Scripts\python.exe C:/Users/Evil/PycharmP'. The output is a traceback starting with 'Traceback (most recent call last):', followed by the file path 'File "C:\Users\Evil\PycharmProjects\LR#16\examples\e7.py", line 38, in <module>' and the code line 'c = Animal()'. The error message is 'TypeError: Can't instantiate abstract class Animal with abstract method move'. At the bottom, it says 'Process finished with exit code 1'. On the left side of the terminal, there is a vertical toolbar with icons for navigation and editing.

```
e7 x
C:\Users\Evil\PycharmProjects\LR#16\venv\Scripts\python.exe C:/Users/Evil/PycharmP
Traceback (most recent call last):
  File "C:\Users\Evil\PycharmProjects\LR#16\examples\e7.py", line 38, in <module>
    c = Animal()
TypeError: Can't instantiate abstract class Animal with abstract method move
Process finished with exit code 1
```

2. Индивидуальное задание №1. Вариант 5.

Составить программу с использованием иерархии классов. Номер варианта необходимо получить у преподавателя. В раздел программы, начинающийся после инструкции `if __name__ == '__main__':` добавить код, демонстрирующий возможности разработанных классов.

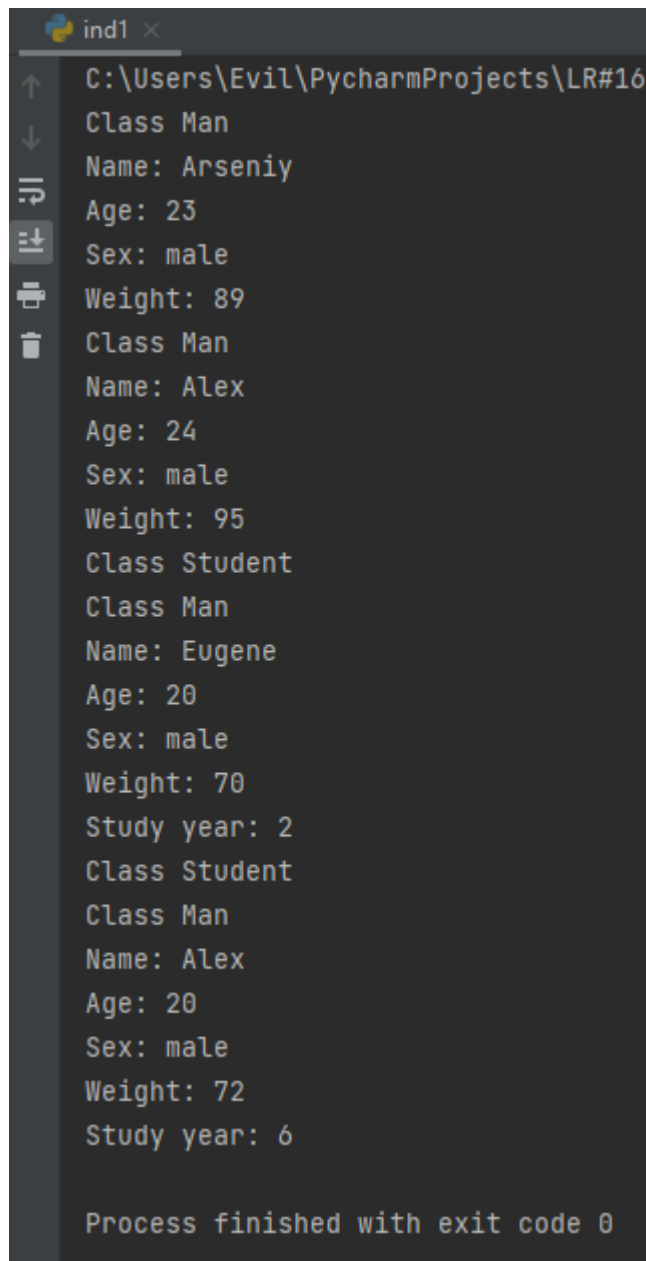
Создать класс `Man` (человек), с полями: имя, возраст, пол и вес. Определить методы переназначения имени, изменения возраста и изменения веса. Создать производный класс `Student`, имеющий поле года обучения. Определить методы переназначения и увеличения года обучения.

```
ind1.py x ex2.py x
1  ▶  1  #!/usr/bin/env python3
2      2  # -*- coding: utf-8 -*-
3
4
5  5  class Man:
6      6  def __init__(self, name, age, sex, weight):
7          7      self.__name = name
8          8      self.__age = age
9          9      self.__sex = sex
10         10      self.__weight = weight
11
12         12  def change_name(self, new_name):
13             13      self.__name = new_name
14
15         15  def change_age(self, new_age):
16             16      self.__age = new_age
17
18         18  def change_weight(self, new_weight):
19             19      self.__weight = new_weight
20
21  21  def print_info(self):
22          22      print("Class Man")
23          23      print(f"Name: {self.__name}")
24          24      print(f"Age: {self.__age}")
25          25      print(f"Sex: {self.__sex}")
26          26      print(f"Weight: {self.__weight}")
27
28
29  29  class Student(Man):
30      30  def __init__(self, name, age, sex, weight, study_year):
31          31      super().__init__(name, age, sex, weight)
32          32      self.__study_year = study_year
33
34         34  def change_study_year(self, new_st_year):
35             35      self.__study_year = new_st_year
36
37         37  def increment_study_year(self):
38             38      self.__study_year += 1
39
Student > print_info()
```

Рисунок 1 – Код программы

```
39
40 def print_info(self):
41     print("Class Student")
42     super().print_info()
43     print(f"Study year: {self.__study_year}")
44
45
46 if __name__ == '__main__':
47     man = Man('Arseniy', 23, 'male', 89)
48     man.print_info()
49     man.change_name('Alex')
50     man.change_age(24)
51     man.change_weight(95)
52     man.print_info()
53     st1 = Student('Eugene', 20, 'male', 70, 2)
54     st1.print_info()
55     st1.change_weight(72)
56     st1.change_name('Alex')
57     st1.increment_study_year()
58     st1.change_study_year(6)
59     st1.print_info()
60
```

Рисунок 2 – Код программы, продолжение



```
ind1 x
C:\Users\Evil\PycharmProjects\LR#16\
Class Man
Name: Arseniy
Age: 23
Sex: male
Weight: 89
Class Man
Name: Alex
Age: 24
Sex: male
Weight: 95
Class Student
Class Man
Name: Eugene
Age: 20
Sex: male
Weight: 70
Study year: 2
Class Student
Class Man
Name: Alex
Age: 20
Sex: male
Weight: 72
Study year: 6

Process finished with exit code 0
```

Рисунок 3 – Результат выполнения программы

3. Индивидуальное задание №2. Вариант 5.

В следующих заданиях требуется реализовать абстрактный базовый класс, определив в нем абстрактные методы и свойства. Эти методы определяются в производных классах. В базовых классах должны быть объявлены абстрактные методы ввода/вывода, которые реализуются в производных классах.

Вызывающая программа должна продемонстрировать все варианты вызова переопределенных абстрактных методов. Написать функцию вывода, получающую параметры базового класса по ссылке и демонстрирующую виртуальный вызов.

Создать абстрактный базовый класс Triangle для представления треугольника с виртуальными функциями вычисления площади и периметра. Поля данных должны включать две стороны и угол между ними. Определить классы-наследники: прямоугольный треугольник, равнобедренный треугольник, равносторонний треугольник со своими функциями вычисления площади и периметра.



```
1  #!/usr/bin/env python3
2  #- coding: utf-8 -*-
3
4
5  from math import pi, sqrt, cos, sin
6  from abc import ABC, abstractmethod
7
8
9  class Triangle(ABC):
10     @abstractmethod
11     def __init__(self):
12         pass
13
14     @abstractmethod
15     def square(self):
16         pass
17
18     @abstractmethod
19     def perimeter(self):
20         pass
21
22     def print_info(self):
23         print("Base class method")
24         print(f"The area of the triangle is: {self.square()}")
25         print(f"The perimeter of the triangle is: {self.perimeter()}")
26
27
28  class RectTriangle(Triangle):
29     def __init__(self, side1, side2, angle):
30         self.__side1 = side1
31         self.__side2 = side2
32         self.__angle = angle * pi
33
34     def square(self):
35         return 0.5 * self.__side1 * self.__side2
36
37     def perimeter(self):
38         return self.__side1 + self.__side2 + \
39             sqrt(self.__side1 ** 2 + self.__side2 ** 2 -
```

Рисунок 4 – Код программы

```
ind2.py x
40         2 * self.__side1 * self.__side2 * cos(self.__angle))
41
42
43     class IsosTriangle(Triangle):
44     def __init__(self, side1, side2, angle):
45         self.__side1 = side1
46         self.__side2 = side2
47         self.__angle = angle * pi
48
49     def square(self):
50         return 0.5 * self.__side1 * self.__side2 * sin(self.__angle)
51
52     def perimeter(self):
53         return self.__side1 + self.__side2 + \
54             sqrt(self.__side1 ** 2 + self.__side2 ** 2 -
55                 2 * self.__side1 * self.__side2 * cos(self.__angle))
56
57
58     class EquilTriangle(Triangle):
59     def __init__(self, side1, side2, angle):
60         if side1 != side2 or angle != 60:
61             raise ValueError
62         else:
63             self.__side1 = side1
64             self.__side2 = side2
65             self.__angle = angle * pi
66
67     def square(self):
68         return (sqrt(3.0) / 4) * self.__side1 ** 2
69
70     def perimeter(self):
71         return 3 * self.__side1
72
73
74     if __name__ == '__main__':
75         rt = RectTriangle(3, 5, 30)
76         print(f"Area of the Rect Triangle is: {rt.square()}")
77         print(f"Perimeter of the Rect Triangle is: {rt.perimeter()}")
78         it = IsosTriangle(4, 4, 67)
79
Triangle > __init__()
```

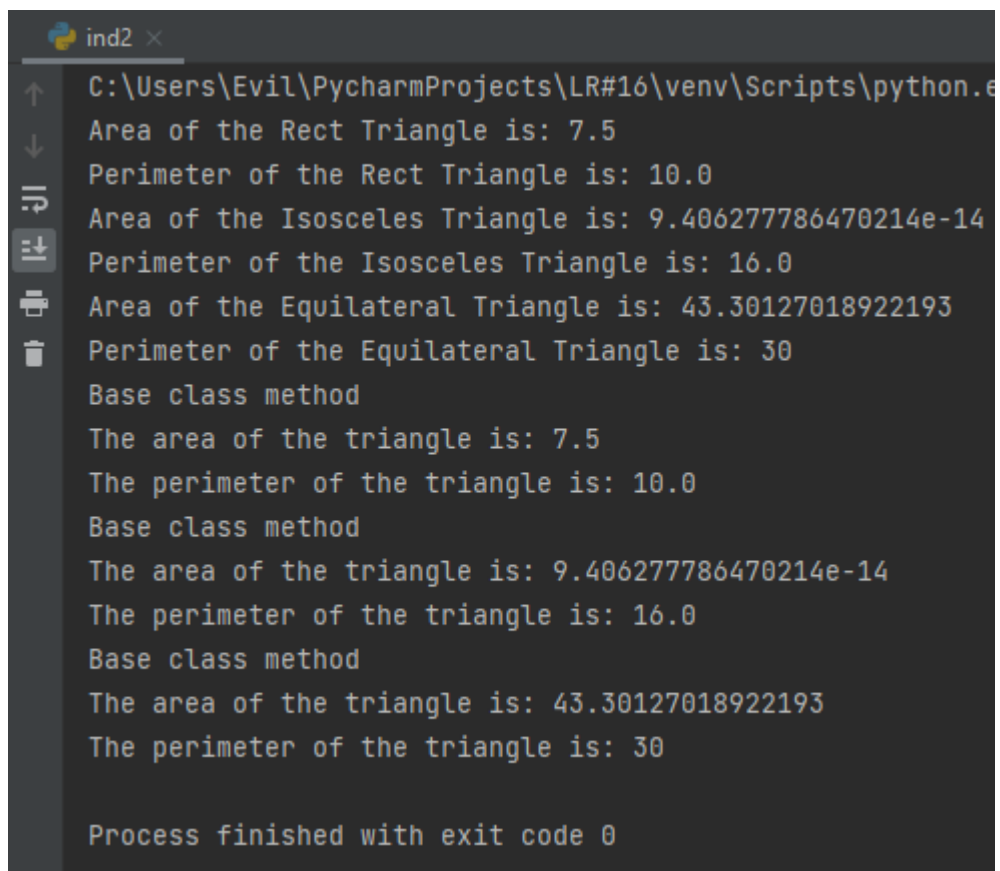
Рисунок 5 – Код программы, продолжение

```

79     print(f"Area of the Isosceles Triangle is: {it.square()}")
80     print(f"Perimeter of the Isosceles Triangle is: {it.perimeter()}")
81     et = EquilTriangle(10, 10, 60)
82     print(f"Area of the Equilateral Triangle is: {et.square()}")
83     print(f"Perimeter of the Equilateral Triangle is: {et.perimeter()}")
84     rt.print_info()
85     it.print_info()
86     et.print_info()
87

```

Рисунок 6 – Код программы, продолжение



```

ind2 x
C:\Users\Evil\PycharmProjects\LR#16\venv\Scripts\python.exe
Area of the Rect Triangle is: 7.5
Perimeter of the Rect Triangle is: 10.0
Area of the Isosceles Triangle is: 9.406277786470214e-14
Perimeter of the Isosceles Triangle is: 16.0
Area of the Equilateral Triangle is: 43.30127018922193
Perimeter of the Equilateral Triangle is: 30
Base class method
The area of the triangle is: 7.5
The perimeter of the triangle is: 10.0
Base class method
The area of the triangle is: 9.406277786470214e-14
The perimeter of the triangle is: 16.0
Base class method
The area of the triangle is: 43.30127018922193
The perimeter of the triangle is: 30

Process finished with exit code 0

```

Рисунок 7 – Результат выполнения программы

4. Ответы на контрольные вопросы

1. Что такое наследование как оно реализовано в языке Python?

Синтаксически создание класса с указанием его родителя выглядит так:

```
class имя_класса(имя_родителя1, [имя_родителя2,..., имя_родителя_n])
```

`super` – это ключевое слово, которое используется для обращения к родительскому классу.

2. Что такое полиморфизм и как он реализован в языке Python?

Полиморфизм, как правило, используется с позиции переопределения методов базового класса в классе наследнике.

Переопределение прописывается в классе-наследнике.

3. Что такое "утиная" типизация в языке программирования Python?

Утиная типизация – это концепция, характерная для языков программирования с динамической типизацией, согласно которой конкретный тип или класс объекта не важен, а важны лишь свойства и методы, которыми этот объект обладает. Другими словами, при работе с объектом его тип не проверяется, вместо этого проверяются свойства и методы этого объекта. Такой подход добавляет гибкости коду, позволяет полиморфно работать с объектами, которые никак не связаны друг с другом и могут быть объектами разных классов. Единственное условие, чтобы все эти объекты поддерживали необходимый набор свойств и методов.

4. Каково назначение модуля abc языка программирования Python?

По умолчанию Python не предоставляет абстрактных классов. Python поставляется с модулем, который обеспечивает основу для определения абстрактных базовых классов (ABC), и имя этого модуля - ABC. ABC работает, декорируя методы базового класса как абстрактные, а затем регистрируя конкретные классы как реализации абстрактной базы.

5. Как сделать некоторый метод класса абстрактным?

Метод становится абстрактным, если он украшен ключевым словом `@abstractmethod`.

6. Как сделать некоторое свойство класса абстрактным?

Абстрактные классы включают в себя атрибуты в дополнение к методам, вы можете потребовать атрибуты в конкретных классах, определив их с помощью `@abstractproperty`.

7. Каково назначение функции `isinstance` ?

Встроенная функция `isinstance(obj, Cls)` , используемая при реализации методов арифметических операций и операций отношения, позволяет узнать что некоторый объект `obj` является либо экземпляром класса `Cls` либо экземпляром одного из потомков класса `Cls`.