

**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 №14 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
5      class Book:
6          material = "paper"
7          cover = "paperback"
8          all_books = []
9
10
11  ▶  if __name__ == '__main__':
12      print(Book.material)
13      print(Book.cover)
14      print(Book.all_books)
15
```

```
e1 x
↑ C:\Users\student-09-525\PycharmProje
↓ paper
  paperback
  []
  Process finished with exit code 0
```

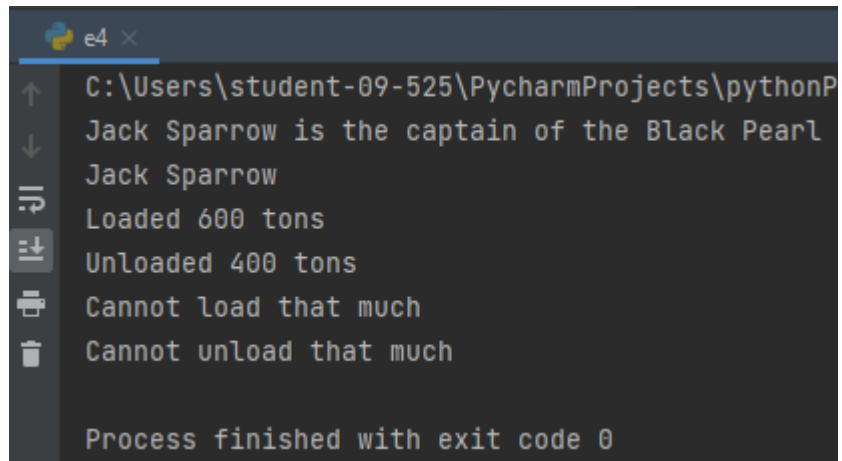
```
e1.py x e2.py x
1  ▶  #!/usr/bin/env python3
2      # -*- coding: utf-8 -*-
3
4
5      class River:
6          # список всех рек
7          all_rivers = []
8
9          def __init__(self, name, length):
10             self.name = name
11             self.length = length
12             # добавляем текущую реку в список всех рек
13             River.all_rivers.append(self)
14
15
16  ▶  if __name__ == '__main__':
17       volga = River("Волга", 3530)
18       seine = River("Сена", 776)
19       nile = River("Нил", 6852)
20       # далее печатаем все названия рек
21       for river in River.all_rivers:
22           print(river.name)
23
```

```
e2 x
↑ C:\Users\student-09-525\PycharmProj
↓ Волга
↻ Сена
↻ Нил
⌕ Process finished with exit code 0
```

```
e1.py x e2.py x e3.py x
1 ▶ #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4
5 class River:
6     all_rivers = []
7
8     def __init__(self, name, length):
9         self.name = name
10        self.length = length
11        River.all_rivers.append(self)
12
13    def get_info(self):
14        print("Длина {0} равна {1} км".format(self.name, self.length))
15
16
17 ▶ if __name__ == '__main__':
18     volga = River("Волга", 3530)
19     seine = River("Сена", 776)
20     nile = River("Нил", 6852)
21     volga.get_info()
22     seine.get_info()
23     nile.get_info()
24
```

```
e3 x
C:\Users\student-09-525\PycharmPro
Длина Волга равна 3530 км
Длина Сена равна 776 км
Длина Нил равна 6852 км
Process finished with exit code 0
```

```
e1.py × e2.py × e3.py × e4.py ×
4
5 class Ship:
6     def __init__(self, name, capacity):
7         self.name = name
8         self.capacity = capacity
9         self.cargo = 0
10
11     def load_cargo(self, weight):
12         if self.cargo + weight <= self.capacity:
13             self.cargo += weight
14             print("Loaded {} tons".format(weight))
15         else:
16             print("Cannot load that much")
17
18     def unload_cargo(self, weight):
19         if self.cargo - weight >= 0:
20             self.cargo -= weight
21             print("Unloaded {} tons".format(weight))
22         else:
23             print("Cannot unload that much")
24
25     def name_captain(self, cap):
26         self.captain = cap
27         print("{} is the captain of the {}".format(self.captain, self.name))
28
29
30 if __name__ == '__main__':
31     black_pearl = Ship("Black Pearl", 800)
32     black_pearl.name_captain("Jack Sparrow")
33     print(black_pearl.captain)
34     black_pearl.load_cargo(600)
35     black_pearl.unload_cargo(400)
36     black_pearl.load_cargo(700)
37     black_pearl.unload_cargo(300)
38
```

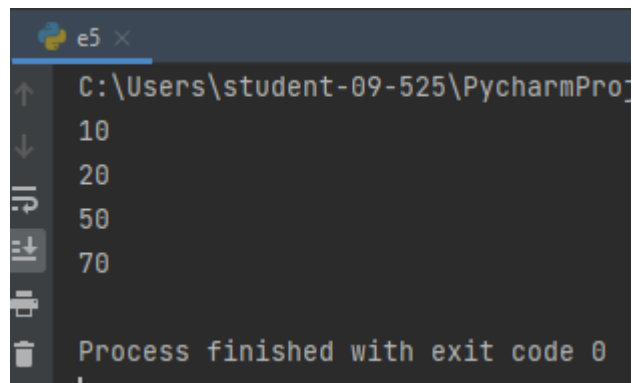
A screenshot of a PyCharm terminal window. The title bar shows a Python icon, the text 'e4', and a close button. The terminal content is as follows:

```
C:\Users\student-09-525\PycharmProjects\pythonP
Jack Sparrow is the captain of the Black Pearl
Jack Sparrow
Loaded 600 tons
Unloaded 400 tons
Cannot load that much
Cannot unload that much

Process finished with exit code 0
```

On the left side of the terminal, there is a vertical toolbar with icons for: running (a green play button), stepping through (a right arrow), searching (a magnifying glass), and other standard IDE actions.

```
e1.py × e2.py × e3.py × e4.py × e5.py ×
5 class Rectangle:
6     def __init__(self, width, height):
7         self.__width = width
8         self.__height = height
9
10    @property
11    def width(self):
12        return self.__width
13
14    @width.setter
15    def width(self, w):
16        if w > 0:
17            self.__width = w
18        else:
19            raise ValueError
20
21    @property
22    def height(self):
23        return self.__height
24
25    @height.setter
26    def height(self, h):
27        if h > 0:
28            self.__height = h
29        else:
30            raise ValueError
31
32    def area(self):
33        return self.__width * self.__height
34
35
36 ▶ if __name__ == '__main__':
37     rect = Rectangle(10, 20)
38     print(rect.width)
39     print(rect.height)
40     rect.width = 50
    if __name__ == '__main__':
```



The image shows a terminal window from the PyCharm IDE. The window has a title bar with a Python logo and the text 'e5 x'. On the left side of the terminal, there is a vertical toolbar with icons for navigating through the code (up, down, search, etc.). The terminal output shows the following:

```
C:\Users\student-09-525\PycharmPro  
10  
20  
50  
70  
  
Process finished with exit code 0
```



```

ex1.py x
1  ▶  #!/usr/bin/env python3
2      # -*- coding: utf-8 -*-
3      class Rational:
4          def __init__(self, a=0, b=1):
5              a = int(a)
6              b = int(b)
7              if b == 0:
8                  raise ValueError()
9              self.__numerator = abs(a)
10             self.__denominator = abs(b)
11             self.__reduce()
12             # Сокращение дроби
13
14         def __reduce(self):
15             # Функция для нахождения наибольшего общего делителя
16             def gcd(a, b):
17                 if a == 0:
18                     return b
19                 elif b == 0:
20                     return a
21                 elif a >= b:
22                     return gcd(a % b, b)
23                 else:
24                     return gcd(a, b % a)
25
26             c = gcd(self.__numerator, self.__denominator)
27             self.__numerator //= c
28             self.__denominator //= c
29
30         @property
31         def numerator(self):
32             return self.__numerator
33
34         @property
35         def denominator(self):
36             return self.__denominator
37
38         # Прочитать значение дроби с клавиатуры. Дробь вводится
39         # как a/b.
40         def read(self, prompt=None):
41             line = input() if prompt is None else input(prompt)
42             parts = list(map(int, line.split('/', maxsplit=1)))
43             if parts[1] == 0:
44                 raise ValueError()
45             self.__numerator = abs(parts[0])
46             self.__denominator = abs(parts[1])
47             self.__reduce()
48
49         # Вывести дробь на экран
50         def display(self):
51             print(f"{self.__numerator}/{self.__denominator}")
52
53         # Сложение обыкновенных дробей.
54         def add(self, rhs):
55             if isinstance(rhs, Rational):

```

```

ex1.py x
55     if isinstance(rhs, Rational):
56         a = self.numerator * rhs.denominator + \
57             self.denominator * rhs.numerator
58         b = self.denominator * rhs.denominator
59         return Rational(a, b)
60     else:
61         raise ValueError()
62
63     # Вычитание обыкновенных дробей.
64     def sub(self, rhs):
65         if isinstance(rhs, Rational):
66             a = self.numerator * rhs.denominator - \
67                 self.denominator * rhs.numerator
68             b = self.denominator * rhs.denominator
69             return Rational(a, b)
70         else:
71             raise ValueError()
72
73     # Умножение обыкновенных дробей.
74     def mul(self, rhs):
75         if isinstance(rhs, Rational):
76             a = self.numerator * rhs.numerator
77             b = self.denominator * rhs.denominator
78             return Rational(a, b)
79         else:
80             raise ValueError()
81
82     # Деление обыкновенных дробей.
83     def div(self, rhs):
84         if isinstance(rhs, Rational):
85             a = self.numerator * rhs.denominator
86             b = self.denominator * rhs.numerator
87             return Rational(a, b)
88         else:
89             raise ValueError()
90
91     # Отношение обыкновенных дробей.
92     def equals(self, rhs):
93         if isinstance(rhs, Rational):
94             return (self.numerator == rhs.numerator) and \
95                 (self.denominator == rhs.denominator)
96         else:
97             return False
98
99     def greater(self, rhs):
100         if isinstance(rhs, Rational):
101             v1 = self.numerator / self.denominator
102             v2 = rhs.numerator / rhs.denominator
103             return v1 > v2
104         else:
105             return False
106
107     def less(self, rhs):
108         if isinstance(rhs, Rational):
109             v1 = self.numerator / self.denominator

```

Rational > read()

```

106
107     def less(self, rhs):
108         if isinstance(rhs, Rational):
109             v1 = self.numerator / self.denominator
110             v2 = rhs.numerator / rhs.denominator
111             return v1 < v2
112         else:
113             return False
114
115
116 ▶ if __name__ == '__main__':
117     r1 = Rational(3, 4)
118     r1.display()
119     r2 = Rational()
120     r2.read("Введите обыкновенную дробь: ")
121     r2.display()
122     r3 = r2.add(r1)
123     r3.display()
124     r4 = r2.sub(r1)
125     r4.display()
126     r5 = r2.mul(r1)
127     r5.display()
128     r6 = r2.div(r1)
129     r6.display()
130

```

```

ex1 x
C:\Users\student-09-525\PycharmProje
3/4
Введите обыкновенную дробь: 5/6
5/6
19/12
1/12
5/8
10/9
Process finished with exit code 0

```

2. Индивидуальное задание №1

Парой называется класс с двумя полями, которые обычно имеют имена `first` и `second`. Требуется реализовать тип данных с помощью такого класса. Во всех заданиях обязательно должны присутствовать:

- метод инициализации `__init__` ; метод должен контролировать значения аргументов на корректность;
- ввод с клавиатуры `read` ;
- вывод на экран `display` .

Реализовать внешнюю функцию с именем `make_тип()` , где `тип` — тип реализуемой структуры. Функция должна получать в качестве аргументов значения для полей структуры и возвращать структуру требуемого типа. При передаче ошибочных параметров следует выводить сообщение и заканчивать работу.

Номер варианта необходимо уточнить у преподавателя. В раздел программы, начинающийся после инструкции `if __name__ == '__main__':` добавить код, демонстрирующий возможности разработанного класса.

Вариант 5: Поле `first` — дробное положительное число, цена товара; поле `second` — целое положительное число, количество единиц товара. Реализовать метод `cost()` — вычисление стоимости товара.

```
ind1.py x
1  ▶ #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4
5  def is_number(s):
6      try:
7          float(s)
8      except ValueError:
9          return False
10     return True
11
12
13  def make_product(first, second):
14     if is_number(first) and is_number(second) and first > 0 and second > 0:
15         product = Product(first, second)
16         return product
17     else:
18         raise ValueError
19
20
21  class Product:
22     def __init__(self, first=0.0, second=0):
23         if is_number(first) and is_number(second):
24             if first > 0 and second > 0:
25                 self.__first = first
26                 self.__second = second
27             else:
28                 raise ValueError
29         else:
30             raise ValueError
31
32     def read(self):
33         self.__first = float(input("Enter the first value: "))
34         self.__second = int(input("Enter the second value: "))
35
36     def display(self):
37         print(f"First value: {self.__first}")
38         print(f"Second value: {self.__second}")
39
Product
```

Рисунок 1 – Код программы

```

39
40     def cost(self):
41         return self.__first * self.__second
42
43
44     if __name__ == '__main__':
45         p1 = make_product(2.25, 4)
46         p1.display()
47         p1.read()
48         print(f"The product's cost: {p1.cost()}")
49         p2 = make_product("aaddfd", 4)
50         p2.display()
51         p2.cost()
52

```

Рисунок 2 – Код программы, продолжение

```

ind1 x
"C:\Users\Evil\PycharmProjects\LR #14\venv\Scripts\python.exe" "C:/Users/Evil/Py
First value: 2.25
Second value: 4
The product's cost: 9.0
Enter the first value: 52.5
Enter the second value: 2
The product's cost: 105.0
Traceback (most recent call last):
  File "C:\Users\Evil\PycharmProjects\LR #14\ind1.py", line 50, in <module>
    p2 = make_product("aaddfd", 4)
  File "C:\Users\Evil\PycharmProjects\LR #14\ind1.py", line 18, in make_product
    raise ValueError
ValueError
Process finished with exit code 1

```

Рисунок 3 – Результат выполнения программы для создания типа с корректными вводными данными и с некорректными

3. Индивидуальное задание №2

Составить программу с использованием классов и объектов для решения задачи. Во всех заданиях, помимо указанных в задании операций, обязательно должны быть реализованы следующие методы:

- метод инициализации `__init__` ;
- ввод с клавиатуры `read` ;
- вывод на экран `display` .

Номер варианта необходимо уточнить у преподавателя. В раздел программы, начинающийся после инструкции `if __name__ == '__main__':` добавить код, демонстрирующий возможности разработанного класса.

Вариант 5: создать класс `Angle` для работы с углами на плоскости, задаваемыми величиной в градусах и минутах. Обязательно должны быть реализованы: перевод в радианы, приведение к диапазону 0-360, увеличение и уменьшение угла на заданную величину, получение синуса, сравнение углов.

```
ind1.py × ind2.py ×
1 ▶ #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4
5     from math import pi, sin
6
7
8     class Angle:
9     def __init__(self, degrees=0.0, minutes=0):
10         self.__degrees = degrees
11         self.__minutes = minutes
12         self.__radians = 0
13
14     def read(self):
15         self.__degrees = float(input("Enter the degrees: "))
16         self.__minutes = int(input("Enter the minutes: "))
17
18     def to_degrees(self):
19         return self.__degrees + self.__minutes / 60
20
21     def to_radians(self):
22         self.__radians = self.__degrees * pi / 180 \
23             + self.__minutes * pi / (180 * 60)
24         return self.__radians
25
26     def normalize_to_360(self):
27         return self.to_degrees() % 360
28
29     def inc_angle(self, value):
30         self.__degrees += value
31         print("The angle has been incremented")
32
33     def dec_angle(self, value):
34         self.__degrees -= value
35         print("The angle has been decremented")
36
37     def find_sin(self):
38         return sin(self.to_radians())
39
40 if __name__ == '__main__':
41     while True:
42         command = input("Enter command: ")
43         if command == 'q':
44             break
45         elif command == 'r':
46             angle.read()
47         elif command == 'd':
48             angle.to_degrees()
49         elif command == 'r':
50             angle.to_radians()
51         elif command == 'n':
52             angle.normalize_to_360()
53         elif command == 'i':
54             angle.inc_angle(1)
55         elif command == 'd':
56             angle.dec_angle(1)
57         elif command == 's':
58             angle.find_sin()
59         else:
60             print("Invalid command")
```

Рисунок 4 – Код программы


```
ind1.py × ind2.py ×
40 def compare(self, degrees):
41     if self.to_degrees() == degrees:
42         print(f"The angles {self.to_degrees()} and {degrees} are equal")
43     else:
44         print(
45             f"The angles {self.to_degrees()} and {degrees} are not equal"
46         )
47
48 def display(self):
49     print(f"The angle degrees: {self.__degrees}")
50     print(f"The angle minutes: {self.__minutes}")
51     print(f"The angle radians: {self.to_radians()}")
52     print(f"The angle sin: {self.find_sin()}")
53
54
55 ▶ if __name__ == '__main__':
56     print("The options are:\n"
57         "1 - Convert to degrees\n"
58         "2 - Convert to radians\n"
59         "3 - Normalize to 0 - 360 diapason\n"
60         "4 - Increment angle by the value\n"
61         "5 - Decrement angle by the value\n"
62         "6 - Find the sin of the angle\n"
63         "7 - Compare with another angle\n"
64         "8 - Read the angle degrees and minutes from the keyboard\n"
65         "9 - Display the angle information\n"
66         "0 - Exit the program\n"
67     )
68     angle = Angle()
69     while True:
70         command = int(input("Enter the command: "))
71         if command == 1:
72             print(angle.to_degrees())
73         elif command == 2:
74             print(angle.to_radians())
75         elif command == 3:
76             print(angle.normalize_to_360())
77         elif command == 4:
78             val = float(input("Enter the value to increment on: "))
79
if __name__ == '__main__' > while True > elif command == 5
```

Рисунок 5 – Код программы, продолжение

```
78         val = float(input("Enter the value to increment on: "))
79         angle.inc_angle(val)
80     elif command == 5:
81         val = float(input("Enter the value to decrement on: "))
82         angle.dec_angle(val)
83     elif command == 6:
84         print(angle.find_sin())
85     elif command == 7:
86         degrees2 = float(input("Enter the degrees of the 2nd angle"))
87         minutes2 = int(input("Enter the minutes of the 2nd angle"))
88         angle2 = Angle(degrees2, minutes2)
89         angle.compare(angle2.to_degrees())
90     elif command == 8:
91         angle.read()
92     elif command == 9:
93         angle.display()
94     elif command == 0:
95         break
96
```

Рисунок 6 – Код программы, продолжение

```
ind2 x
"C:\Users\Evil\PycharmProjects\LR #14\venv\Scripts\python.exe" "C:/Users/E
The options are:
1 - Convert to degrees
2 - Convert to radians
3 - Normalize to 0 - 360 diapason
4 - Increment angle by the value
5 - Decrement angle by the value
6 - Find the sin of the angle
7 - Compare with another angle
8 - Read the angle degrees and minutes from the keyboard
9 - Display the angle information
0 - Exit the program

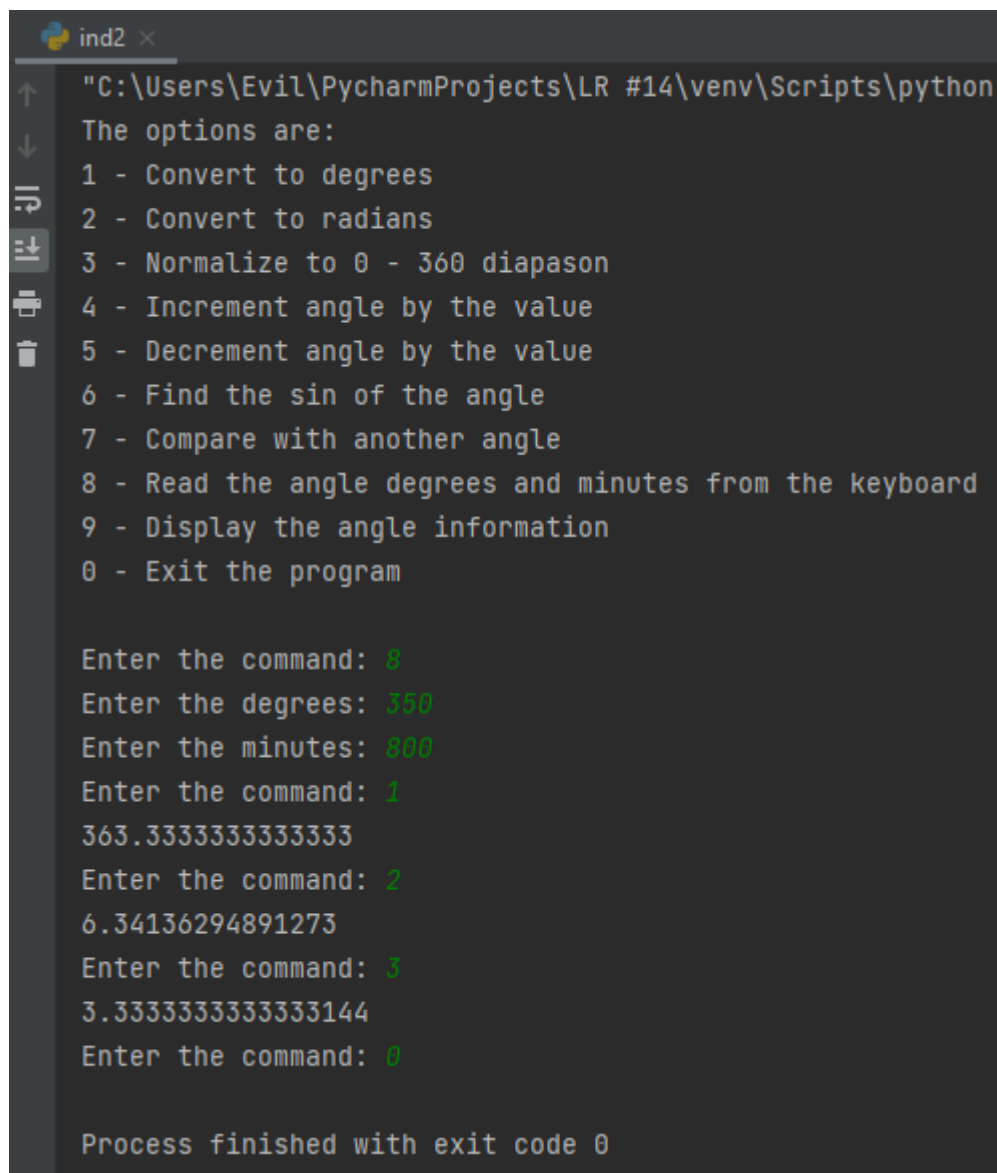
Enter the command: 8
Enter the degrees: 56
Enter the minutes: 56
Enter the command: 1
56.93333333333333
Enter the command: 2
0.9936741208021049
Enter the command: 3
56.93333333333333
Enter the command: 4
Enter the value to increment on: 94
The angle has been incremented
Enter the command: 6
0.4858269580752267
Enter the command: 7
Enter the degrees of the 2nd angle: 150
Enter the minutes of the 2nd angle: 56
The angles 150.93333333333334 and 150.93333333333334 are equal
Enter the command: 9
The angle degrees: 150.0
The angle minutes: 56
The angle radians: 2.634283617676775
The angle sin: 0.4858269580752267
Enter the command: 7
Enter the degrees of the 2nd angle: 87
Enter the minutes of the 2nd angle: 34
The angles 150.93333333333334 and 87.56666666666666 are not equal
```

Рисунок 7 – Результат выполнения программы

```
The angles 150.93333333333334 and 87.56666666666666 are not equal
Enter the command: 5
Enter the value to decrement on: 3
The angle has been decremented
Enter the command: 9
The angle degrees: 147.0
The angle minutes: 56
The angle radians: 2.5819237401169453
The angle sin: 0.5309056540598305
Enter the command: 0

Process finished with exit code 0
```

Рисунок 8 – Результат выполнения программы, продолжение



```
ind2 x
"C:\Users\Evil\PycharmProjects\LR #14\venv\Scripts\python
The options are:
1 - Convert to degrees
2 - Convert to radians
3 - Normalize to 0 - 360 diapason
4 - Increment angle by the value
5 - Decrement angle by the value
6 - Find the sin of the angle
7 - Compare with another angle
8 - Read the angle degrees and minutes from the keyboard
9 - Display the angle information
0 - Exit the program

Enter the command: 8
Enter the degrees: 350
Enter the minutes: 800
Enter the command: 1
363.3333333333333
Enter the command: 2
6.34136294891273
Enter the command: 3
3.3333333333333144
Enter the command: 0

Process finished with exit code 0
```

Рисунок 9 – Приведение к диапазону 0-360

4. Ответы на контрольные вопросы

1. Как осуществляется объявление класса в языке Python?

Классы объявляются с помощью ключевого слова `class` и имени класса:

```
class MyClass:
```

```
    var = ... # некоторая переменная
```

```
    def do_smt(self):
```

```
        # какой-то метод
```

2. Чем атрибуты класса отличаются от атрибутов экземпляра?

Атрибут класса - это атрибут, общий для всех экземпляров класса. Атрибуты класса определены внутри класса, но вне каких-либо методов. Их значения одинаковы для всех экземпляров этого класса. Так что вы можете рассматривать их как тип значений по умолчанию для всех наших объектов.

Атрибуты экземпляра определяются в методах и хранят информацию, специфичную для экземпляра.

3. Каково назначение методов класса?

Методы определяют функциональность объектов, принадлежащих конкретному классу.

4. Для чего предназначен метод `__init__()` класса?

Метод `__init__` является конструктором. Конструкторы - это концепция объектно-ориентированного программирования. Класс может иметь один и только один конструктор. Если `__init__` определен внутри класса, он автоматически вызывается при создании нового экземпляра класса. Метод `__init__` указывает, какие атрибуты будут у экземпляров нашего класса.

5. Каково назначение `self` ?

Аргумент `self` представляет конкретный экземпляр класса и позволяет нам получить доступ к его атрибутам и методам. В примере с `__init__` мы создаем атрибуты для конкретного экземпляра и присваиваем им значения аргументов метода. Важно использовать параметр `self` внутри метода, если мы хотим сохранить значения экземпляра для последующего использования.

В большинстве случаев нам также необходимо использовать параметр `self` в других методах, потому что при вызове метода первым аргументом, который ему передается, является сам объект.

6. Как добавить атрибуты в класс?

Новый атрибут класса указывается через точку после названия класса, затем ему присваивается определенное значение.

7. Как осуществляется управление доступом к методам и атрибутам в языке Python?

Хорошим тоном считается, что для чтения/изменения какого-то атрибута должны использоваться специальные методы, которые называются `getter/setter`, их можно реализовать, но ничего не помешает изменить атрибут напрямую. При этом есть соглашение, что метод или атрибут, который начинается с нижнего подчеркивания, является скрытым, и снаружи класса трогать его не нужно (хотя сделать это можно).

Если же атрибут или метод начинается с двух подчеркиваний, то тут напрямую вы к нему уже не обратитесь (простым образом).

8. 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`.