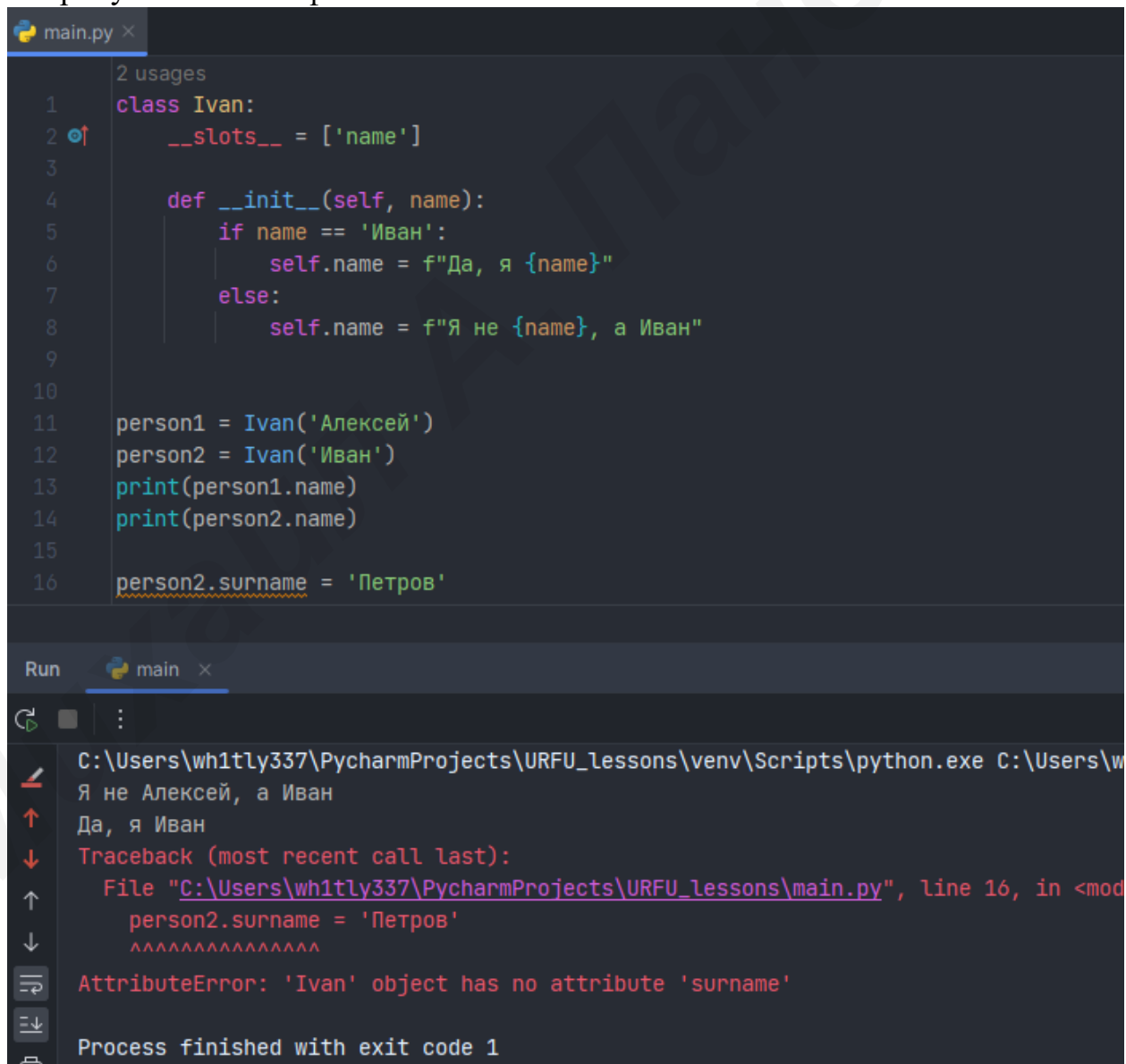


Лабораторная работа 9

ТЕМА 9. Концепции и принципы ООП

Лабораторные задания:

- 1) Допустим, что вы решили оригинально и немного странно познакомиться с человеком. Для этого у вас должен быть написан свой класс на Python, который будет проверять угадал ваше имя человек или нет. Для этого создайте класс, указав в свойствах только имя. Далее создайте функцию `__init__()`, а в ней сделайте проверку на то угадал человек ваше имя или нет. Также можете проверить что будет, если в этой функции указав атрибут, который не указан в вашем классе, например, попробуйте вызвать фамилию.

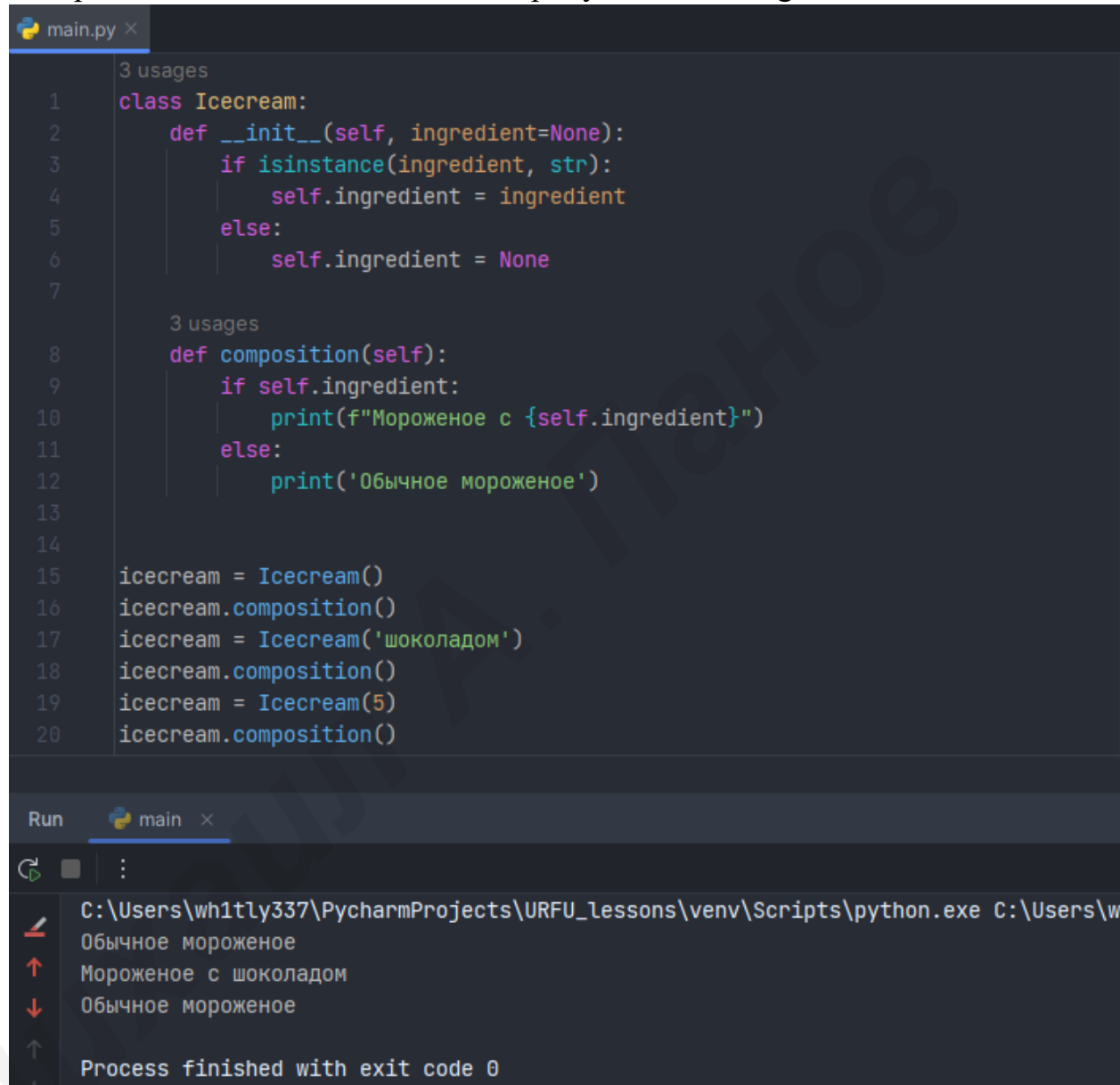


```
main.py x
2 usages
1 class Ivan:
2     __slots__ = ['name']
3
4     def __init__(self, name):
5         if name == 'Иван':
6             self.name = f"Да, я {name}"
7         else:
8             self.name = f"Я не {name}, а Иван"
9
10
11 person1 = Ivan('Алексей')
12 person2 = Ivan('Иван')
13 print(person1.name)
14 print(person2.name)
15
16 person2.surname = 'Петров'
```

```
Run main x
C:\Users\wh1tly337\PycharmProjects\URFU_lessons\venv\Scripts\python.exe C:\Users\w
Я не Алексей, а Иван
Да, я Иван
Traceback (most recent call last):
  File "C:\Users\wh1tly337\PycharmProjects\URFU_lessons\main.py", line 16, in <mod
    person2.surname = 'Петров'
    ^^^^^^^^^^^^^^^^^
AttributeError: 'Ivan' object has no attribute 'surname'
Process finished with exit code 1
```

- 2) Вам дали важное задание, написать продавцу мороженого программу, которая будет писать добавили ли топпинг в мороженое и цену после возможного изменения. Для этого вам нужно написать класс, в котором

будет определяться изменили ли состав мороженого или нет. В этом классе реализуйте метод, выводящий на печать «Мороженое с {ТОППИНГ}» в случае наличия добавки, а иначе отобразится следующая фраза: «Обычное мороженое». При этом программа должна воспринимать как топпинг только атрибуты типа string.



```
main.py x
3 usages
1 class Icecream:
2     def __init__(self, ingredient=None):
3         if isinstance(ingredient, str):
4             self.ingredient = ingredient
5         else:
6             self.ingredient = None
7
8     3 usages
9     def composition(self):
10        if self.ingredient:
11            print(f"Мороженое с {self.ingredient}")
12        else:
13            print('Обычное мороженое')
14
15 icecream = Icecream()
16 icecream.composition()
17 icecream = Icecream('шоколадом')
18 icecream.composition()
19 icecream = Icecream(5)
20 icecream.composition()
```

Run main x

C:\Users\wh1tly337\PycharmProjects\URFU_lessons\venv\Scripts\python.exe C:\Users\w
Обычное мороженое
Мороженое с шоколадом
Обычное мороженое
Process finished with exit code 0

- 3) Петя – начинающий программист и на занятиях ему сказали реализовать икапсу...что-то. А вы хороший друг Пети и ко всему прочему прекрасно знаете, что икапсу...что-то – это инкапсуляция, поэтому решаете помочь вашему другу с написанием класса с инкапсуляцией. Ваш класс будет не просто инкапсуляцией, а классом с сеттером, геттером и деструктором. После написания класса вам необходимо продемонстрировать что все написанные вами функции работают. Также вас необходимо объяснить Пете почему на скриншоте ниже в консоли выводится ошибка.

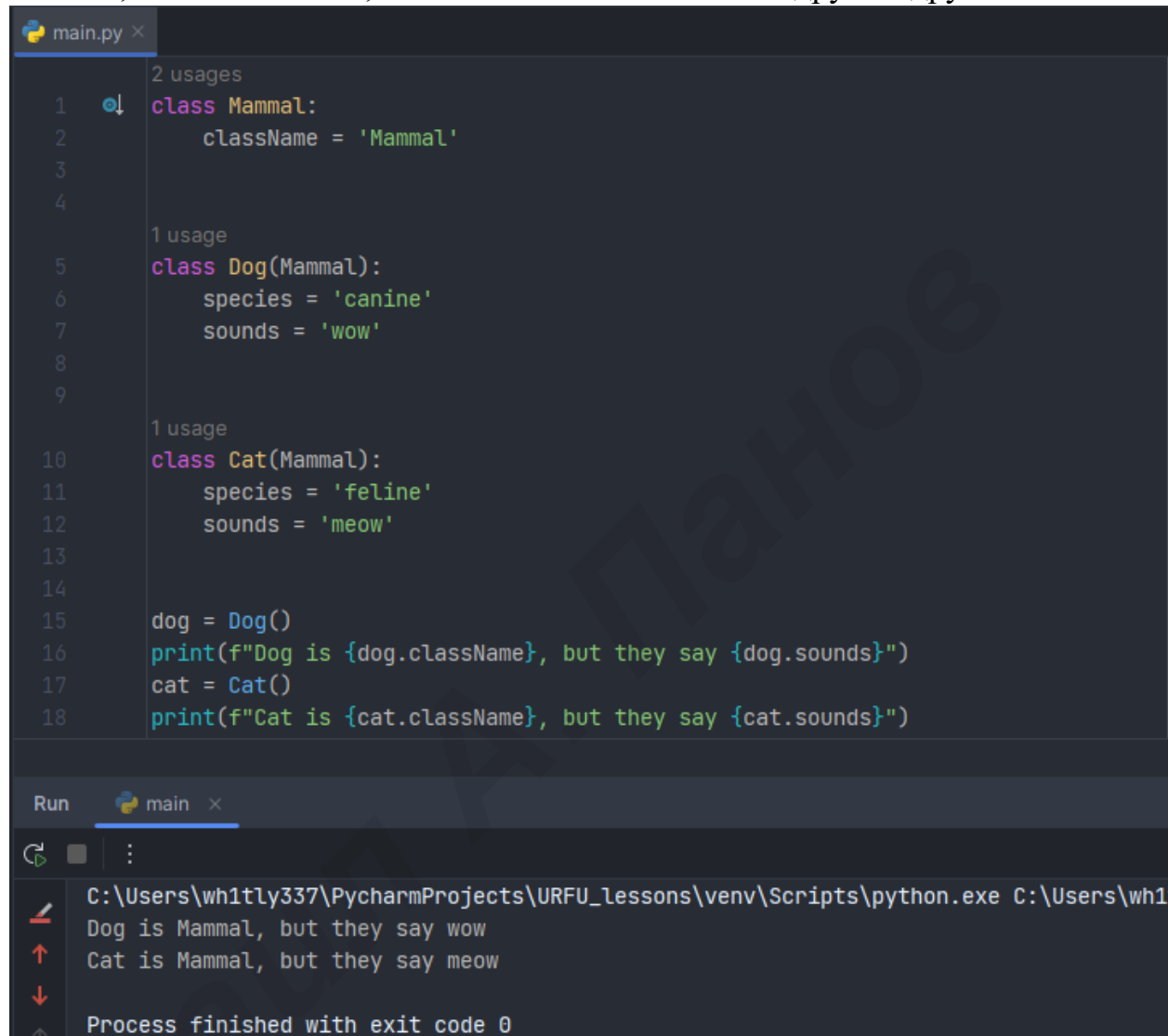
```
main.py x
1 usage
2 class MyClass:
3     def __init__(self, value):
4         self._value = value
5
6     3 usages
7     def set_value(self, value): # установка значения атрибута
8         self._value = value
9
10    5 usages
11    def get_value(self): # получение значения атрибута
12        return self._value
13
14    2 usages
15    def del_value(self): # удаление атрибута
16        del self._value
17
18    value = property(get_value, set_value, del_value, "Свойство value")
19
20    obj = MyClass(42)
21    print(obj.get_value())
22    obj.set_value(45)
23    print(obj.get_value())
24    obj.set_value(100)
25    print(obj.get_value())
26    obj.del_value()
27    print(obj.get_value())

Run main x
C:\Users\wh1tly337\PycharmProjects\URFU_lessons\venv\Scripts\python.exe C:\Users\w
42
45
100
Traceback (most recent call last):
  File "C:\Users\wh1tly337\PycharmProjects\URFU_lessons\main.py", line 24, in <mod
    print(obj.get_value())
    ^^^^^^^^^^^^^^^^^^^
  File "C:\Users\wh1tly337\PycharmProjects\URFU_lessons\main.py", line 9, in get_v
    return self._value
    ^^^^^^^^^^^^^^^
AttributeError: 'MyClass' object has no attribute '_value'. Did you mean: 'value'?

Process finished with exit code 1
```

- 4) Вам прекрасно известно, что кошки и собаки являются млекопитающими, но компьютер этого не понимает, поэтому вам нужно написать три класса: Кошки, Собаки, Млекопитающие. И при помощи

“наследования” объяснить компьютеру что кошки и собаки – это млекопитающие. Также добавьте какой-нибудь свой атрибут для кошек и собак, чтобы показать, что они чем-то отличаются друг от друга.



```
main.py x
2 usages
1 class Mammal:
2     className = 'Mammal'
3
4
5 1 usage
6 class Dog(Mammal):
7     species = 'canine'
8     sounds = 'wow'
9
10 1 usage
11 class Cat(Mammal):
12     species = 'feline'
13     sounds = 'meow'
14
15 dog = Dog()
16 print(f"Dog is {dog.className}, but they say {dog.sounds}")
17 cat = Cat()
18 print(f"Cat is {cat.className}, but they say {cat.sounds}")

Run main x
C:\Users\wh1tly337\PycharmProjects\URFU_lessons\venv\Scripts\python.exe C:\Users\wh1
Dog is Mammal, but they say wow
Cat is Mammal, but they say meow
Process finished with exit code 0
```

- 5) На разных языках здороваются по-разному, но суть остается одинаковой, люди друг с другом здороваются. Давайте вместе с вами реализуем программу с полиморфизмом, которая будет описывать всю суть первого предложения задачи. Для этого мы можем выбрать два языка, например, русский и английский и написать для них отдельные классы, в которых будет в виде атрибута слово, которым здороваются на этих языках. А также напишем функцию, которая будет выводить информацию о том, как на этих языках здороваются. Заметьте, что для решения поставленной задачи мы использовали декоратор `@staticmethod`, поскольку нам не нужны обязательные параметры-ссылки вроде `self`.

```
main.py ×
1 usage
  class Russian:
    1 usage (1 dynamic)
    @staticmethod
    3 def greeting():
    4     print("Привет")
    5
    6
    1 usage
    7 class English:
    8     1 usage (1 dynamic)
    9     @staticmethod
    10    def greeting():
    11        print("Hello")
    12
    2 usages
    13 def greet(language):
    14     language.greeting()
    15
    16
    17 ivan = Russian()
    18 greet(ivan)
    19 john = English()
    20 greet(john)

Run main ×
C:\Users\wh1tly337\PycharmProjects\URFU_lessons\venv\Scripts\python.exe C:\Users\wh1
Привет
Hello
Process finished with exit code 0
```