

**Лабораторная работа №1**  
**«ИССЛЕДОВАНИЕ БАЗОВЫХ ФУНКЦИЙ**  
**ЯЗЫКА PYTHON»**

**Цель работы**

Изучение технологии подготовки и выполнения программ на языке Python, исследование свойств функций языка Python, используемых при обработке последовательностей, формирование навыков написания программ работы с классами на языке Python

**Постановка задачи**

1. Изучите основы языка Python, структуры данных и методы обработки списков, кортежей, множеств, словарей, классы и объекты Python, среду программирования на языке Python. Проверьте выполнение приведенных примеров в среде программирования.

2. Выполните задания 1 – 5 из раздела 1.3 в интерактивном режиме, используя возможности IPython. Результаты выполнения каждого задания внесите в отчет.

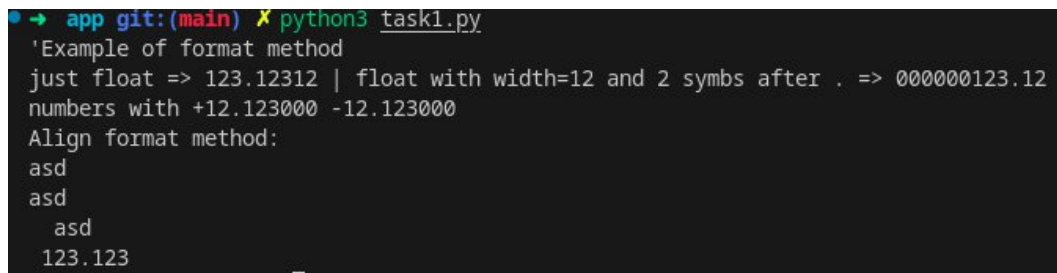
1.4.3. Определите в соответствии с заданием 6 из раздела 1.3 функции и методы для решения 3-х сформулированных задач. Используйте для редактирования функций и выполнения кода интегрированную среду Spider IDE. Зафиксируйте результаты выполнения функций во всех необходимых режимах. Выполните с помощью autograder.py автооценивание. При обнаружении ошибок отредактируйте код. Результаты автооценивания внести в отчет.

## Ход работы

1. Используя команды `dir` и `help`, были изучены следующие методы строкового типа: `'format'`, `'strip'`, `'lstrip'`, `'rstrip'`, `'capitalize'`, `'title'`, `'count'`, `'index'`, `'rindex'`, `'startswith'`, `'endswith'`, `'replace'`, `'split'`, `'rsplit'`, `'join'`, `'partition'`, `'rpartition'`. Примеры использования продемонстрированы на рисунках 1 – 17

### Листинг 1 – Пример использования метода `format`

```
#format
print("{0} {1}".format("Example", "of format method"))
print("just float => {0} | float with width=12 and 2 syms after . =>
{0:012.2f}".format(123.12312))
print("numbers with {0:+f} {1: f}".format(12.123, -12.123))
print("Align format method:")
print("{:5}".format("asd"))
print("{:<5}".format("asd"))
print("{:>5}".format("asd"))
print("{:= 6.3f}".format(123.12312))
```



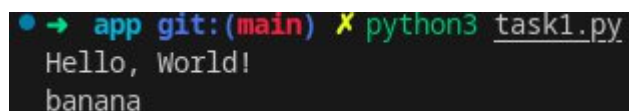
```
➔ app git:(main) X python3 task1.py
'Example of format method
just float => 123.12312 | float with width=12 and 2 syms after . => 000000123.12
numbers with +12.123000 -12.123000
Align format method:
asd
asd
  asd
123.123
```

Рисунок 1 – Метод `format`

### Листинг 2 – Пример использования метода `strip`

```
#strip
# Удаление пробелов в начале и конце строки
txt = " Hello, World! "
x = txt.strip()
print(x)

# Удаление определенных символов в начале и конце строки
txt = ".,,.,,rrttgg....banana....rrr"
x = txt.strip(",.grt")
print(x)
```



```
➔ app git:(main) X python3 task1.py
Hello, World!
banana
```

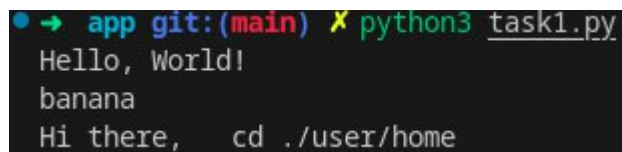
Рисунок 2 – Метод `strip`

### Листинг 3 – Пример использования метода lstrip

```
#lstrip
# Удаление пробелов в начале строки
txt = "    Hello, World!"
x = txt.lstrip()
print(x)

# Удаление определенных символов в начале строки
txt = ".,,.,,ssaaww....banana"
x = txt.lstrip(",.asw")
print(x)

# Удаление пробелов и других символов в начале строки
txt = "    Hi there,    cd ./user/home"
x = txt.lstrip("    ")
print(x)
```



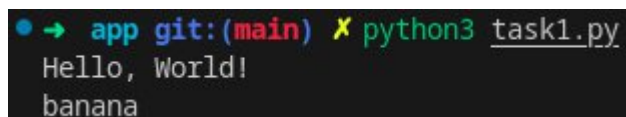
```
app git:(main) X python3 task1.py
Hello, World!
banana
Hi there,    cd ./user/home
```

Рисунок 3 – Метод lstrip

### Листинг 4 – Пример использования метода rstrip

```
#rstrip
# Удаление пробелов в конце строки
txt = "Hello, World!    "
x = txt.rstrip()
print(x)

# Удаление определенных символов в конце строки
txt = "banana...."
x = txt.rstrip(".")
print(x)
```



```
app git:(main) X python3 task1.py
Hello, World!
banana
```

Рисунок 4 – Метод rstrip

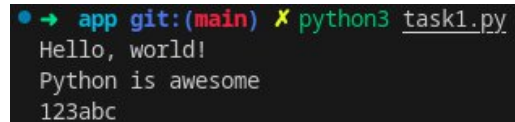
### Листинг 5 – Пример использования метода capitalize

```
#capitalize
string = "hello, world!"
capitalized_string = string.capitalize()
print(capitalized_string)

string = "PYTHON IS AWESOME"
```

```
capitalized_string = string.capitalize()
print(capitalized_string)

string = "123abc"
capitalized_string = string.capitalize()
print(capitalized_string)
```

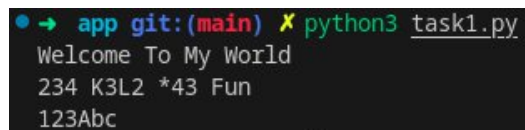


```
• → app git:(main) ✕ python3 task1.py
Hello, world!
Python is awesome
123abc
```

Рисунок 5 – Метод capitalize

## Листинг 6 – Пример использования метода title

```
#title
txt = "Welcome to my world"
x = txt.title()
print(x)
txt = "234 k3l2 *43 fun"
x = txt.title()
print(x)
txt = "123abc"
x = txt.title()
print(x)
```



```
• → app git:(main) ✕ python3 task1.py
Welcome To My World
234 K3L2 *43 Fun
123Abc
```

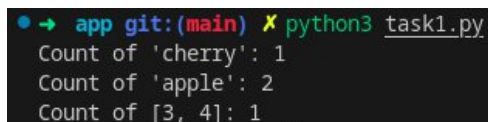
Рисунок 6 – Метод title

## Листинг 7 – Пример использования метода count

```
#count
fruits = ['apple', 'banana', 'cherry']
x = fruits.count("cherry")
print("Count of 'cherry':", x)

txt = "I love apples, apple are my favorite fruit"
x = txt.count("apple")
print("Count of 'apple':", x)

random = ['a', ('a', 'b'), ('a', 'b'), [3, 4]]
x = random.count([3, 4])
print("Count of [3, 4]:", x)
```



```
• → app git:(main) ✕ python3 task1.py
Count of 'cherry': 1
Count of 'apple': 2
Count of [3, 4]: 1
```

Рисунок 7 – Метод count

## Листинг 8 – Пример использования метода index

```
#index
fruits = ['apple', 'banana', 'cherry']
x = fruits.index("cherry")
print(x)

numbers = [4, 55, 64, 32, 16, 32]
x = numbers.index(32)
print(x)

animals = ["cat", "dog", "tiger"]
try:
    x = animals.index("bat")
except ValueError:
    x = -1
print(x)
```

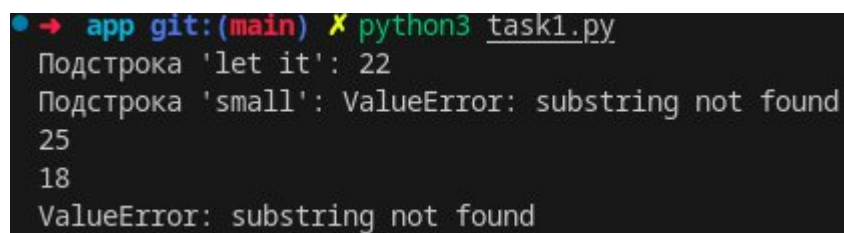


```
app git:(main) X python3 task1.py
2
3
-1
```

Рисунок 8 – Метод index

## Листинг 9 – Пример использования метода rindex

```
#rindex
quote = 'Let it be, let it be, let it be'
result = quote.rindex('let it')
print("Подстрока 'let it':", result)
try:
    result = quote.rindex('small')
except:
    result = "ValueError: substring not found"
print("Подстрока 'small':", result)
# Поиск подстроки с указанием начального и конечного индексов
quote = 'Do small things with great love'
print(quote.rindex('t', 2)) # ищем подстроку 't' начиная с индекса 2
print(quote.rindex('th', 6, 20)) # ищем подстроку 'th' в диапазоне от индекса 6 до
20
try:
    print(quote.rindex('o small ', 10, -1)) # ищем подстроку 'o small ' в диапазоне
от индекса 10 до конца строки
except:
    print("ValueError: substring not found")
```



```
app git:(main) X python3 task1.py
Подстрока 'let it': 22
Подстрока 'small': ValueError: substring not found
25
18
ValueError: substring not found
```

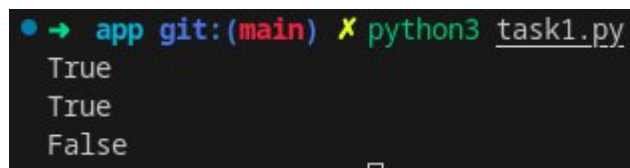
Рисунок 9 – Метод rindex

## Листинг 10 – Пример использования метода startswith

```
#startswith
text = "Python is easy to learn."
result = text.startswith('Python') # возвращает True
print(result)

text = "Python programming is easy."
result = text.startswith('programming', 7) # возвращает True
print(result)

text = "Python programming is easy."
result = text.startswith('programming is', 7, 18) # возвращает False
print(result)
```




```
• → app git:(main) X python3 task1.py
True
True
False
```

Рисунок 10 – Метод startswith

## Листинг 11 – Пример использования метода endswith

```
#endswith
txt = "Hello, welcome to my world."
print(txt.endswith(".")) # Output: True
print(txt.endswith("my world.")) # Output: True

txt = "Hello, welcome to my world."
print(txt.endswith("my world.", 5, 11)) # Output: False
```



```
• → app git:(main) X python3 task1.py
True
True
False
```

Рисунок 11 – Метод endswith

## Листинг 12 – Пример использования метода replace

```
#replace
txt = "I like bananas"
x = txt.replace("bananas", "apples")
print(x)
txt = "one one was a race horse, two two was one too."
x = txt.replace("one", "three")
print(x)
txt = "one one was a race horse, two two was one too."
x = txt.replace("one", "three", 2)
print(x)
```

```

• → app git:(main) ✖ python3 task1.py
I like apples
three three was a race horse, two two was three too.
three three was a race horse, two two was one too.

```

Рисунок 12 – Метод replace

### Листинг 13 – Пример использования метода split

```

#split
txt = "welcome to the jungle"
x = txt.split()
print(x)
txt = "hello, my name is Peter, I am 26 years old"
x = txt.split(", ")
print(x)
txt = "apple#banana#cherry#orange"
x = txt.split("#", 2)
print(x)

```

```

• → app git:(main) ✖ python3 task1.py
['welcome', 'to', 'the', 'jungle']
['hello', 'my name is Peter', 'I am 26 years old']
['apple', 'banana', 'cherry#orange']

```

Рисунок 13 – Метод split

### Листинг 14 – Пример использования метода rsplit

```

#rsplit
txt = "apple, banana, cherry"
x = txt.rsplit(", ")
print(x)
txt = "apple, banana, cherry"
x = txt.rsplit(", ", 1)
print(x)

```

```

• → app git:(main) ✖ python3 task1.py
['apple', 'banana', 'cherry']
['apple, banana', 'cherry']

```

Рисунок 14 – Метод rsplit

### Листинг 15 – Пример использования метода join

```

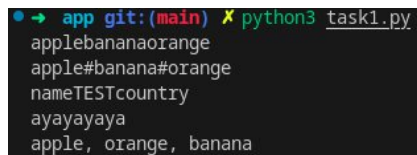
#join
my_list = ['apple', 'banana', 'orange']
s1 = ''.join(my_list)
print(s1)
my_tuple = ('apple', 'banana', 'orange')
s2 = '#'.join(my_tuple)
print(s2)

```

```

my_dict = {'name': 'John', 'country': 'Norway'}
s3 = 'TEST'.join(my_dict)
print(s3)
m1 = 'y'
m2 = 'aaaaa'
s4 = m1.join(m2)
print(s4)
my_set = {'apple', 'banana', 'orange'}
s5 = ', '.join(my_set)
print(s5)

```



```

app git:(main) X python3 task1.py
applebananaorange
apple#banana#orange
nameTESTcountry
ayayayaya
apple, orange, banana

```


Рисунок 15 – Метод join

## Листинг 16 – Пример использования метода partition

```

#partition
txt = "I could eat bananas all day"
x = txt.partition("bananas")
print(x)
txt = "Python is fun"
x = txt.partition("is")
print(x)
txt = "Python is fun, isn't it"
x = txt.partition("is")
print(x)

```



```

app git:(main) X python3 task1.py
('I could eat ', 'bananas', ' all day')
('Python ', 'is', ' fun')
('Python ', 'is', ' fun, isn't it')

```

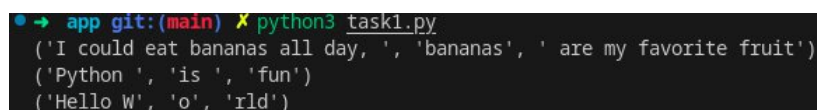
Рисунок 16 – Метод partition

## Листинг 17 – Пример использования метода rpartition

```

#rpartition
txt = "I could eat bananas all day, bananas are my favorite fruit"
x = txt.rpartition("bananas")
print(x)
string = "Python is fun"
print(string.rpartition('is '))
mystr = 'Hello World'
print(mystr.rpartition('o'))

```



```

app git:(main) X python3 task1.py
('I could eat bananas all day, ', 'bananas', ' are my favorite fruit')
('Python ', 'is ', 'fun')
('Hello W', 'o', 'rld')

```

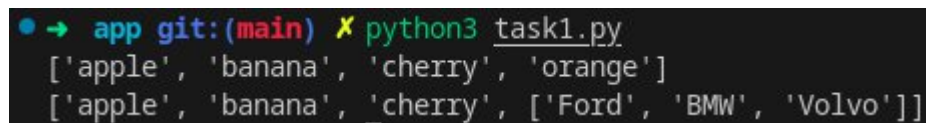
Рисунок 17 – Метод rpartition



2. Используя команды `dir` и `help`, были изучены следующие методы обработки списков: `'append'`, `'count'`, `'extend'`, `'index'`, `'insert'`, `'pop'`, `'remove'`, `'reverse'`, `'sort'`. Пример использования методов представлен на рисунках 18 – 26.

#### Листинг 18 – Пример использования метода `append`

```
#append
fruits = ['apple', 'banana', 'cherry']
fruits.append('orange')
print(fruits)
a = ["apple", "banana", "cherry"]
b = ["Ford", "BMW", "Volvo"]
a.append(b)
print(a)
```

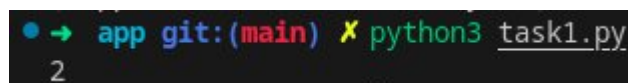


```
• → app git:(main) X python3 task1.py
['apple', 'banana', 'cherry', 'orange']
['apple', 'banana', 'cherry', ['Ford', 'BMW', 'Volvo']]
```

Рисунок 18 – Метод `append`

#### Листинг 19 – Пример использования метода `count`

```
points = [1, 4, 2, 9, 7, 8, 9, 3, 1]
x = points.count(9)
print(x)
```



```
• → app git:(main) X python3 task1.py
2
```

Рисунок 19 – Метод `count`

#### Листинг 20 – Пример использования метода `extend`

```
#extend
fruits = ['apple', 'banana', 'cherry']
cars = ['Ford', 'BMW', 'Volvo']
fruits.extend(cars)
print(fruits) # ['apple', 'banana', 'cherry', 'Ford', 'BMW', 'Volvo']
fruits = ['apple', 'banana', 'cherry']
points = (1, 4, 5, 9)
fruits.extend(points)
print(fruits) # ['apple', 'banana', 'cherry', 1, 4, 5, 9]
fruits = ['apple', 'banana', 'cherry']
string = "orange"
fruits.extend(string)
print(fruits) # ['apple', 'banana', 'cherry', 'o', 'r', 'a', 'n', 'g', 'e']
```

```

• → app git:(main) X python3 task1.py
['apple', 'banana', 'cherry', 'Ford', 'BMW', 'Volvo']
['apple', 'banana', 'cherry', 1, 4, 5, 9]
['apple', 'banana', 'cherry', 'o', 'r', 'a', 'n', 'g', 'e']

```

Рисунок 20 – Метод extend

### Листинг 21 – Пример использования метода index

```

#index
fruits = ['apple', 'banana', 'cherry']
x = fruits.index("cherry")
print(x)

list1 = [1, 2, 3, 4, 1, 1, 1, 4, 5]
print(list1.index(4, 4, 8))
list1 = [1, 2, 3, 4, 1, 1, 1, 4, 5]
try:
    print(list1.index(10))
except ValueError as e:
    print("Element not found")

```

```

• → app git:(main) X python3 task1.py
2
7
Element not found

```

Рисунок 21 – Метод index

### Листинг 22 – Пример использования метода insert

```

#insert
fruits = ['apple', 'banana', 'cherry']
fruits.insert(1, "orange")
print(fruits)

prime_numbers = [2, 3, 5, 7]
prime_numbers.insert(0, 1)
print(prime_numbers)

my_list = [1, 2, 3]
my_list.insert(3, [4, 5, 6])
print(my_list)

```

```

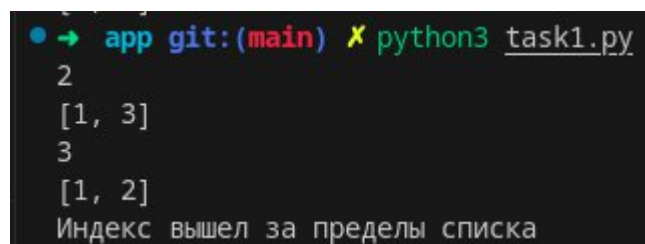
• → app git:(main) X python3 task1.py
['apple', 'orange', 'banana', 'cherry']
[1, 2, 3, 5, 7]
[1, 2, 3, [4, 5, 6]]

```

Рисунок 22 – Метод insert

### Листинг 23 – Пример использования метода pop

```
#pop
my_list = [1, 2, 3]
item = my_list.pop(1)
print(item)
print(my_list)
my_list = [1, 2, 3]
item = my_list.pop()
print(item)
print(my_list)
my_list = [1, 2, 3]
try:
    item = my_list.pop(4)
except IndexError:
    print("Индекс вышел за пределы списка")
```



```
app git:(main) X python3 task1.py
2
[1, 3]
3
[1, 2]
Индекс вышел за пределы списка
```

Рисунок 23 – Метод pop

### Листинг 24 – Пример использования метода remove

```
#remove
animals = ['cat', 'cat', 'dog', 'guinea pig', 'cat']
animals.remove('cat')
print('Обновленный список:', animals)
```



```
app git:(main) X python3 task1.py
Обновленный список: ['cat', 'dog', 'guinea pig', 'cat']
```

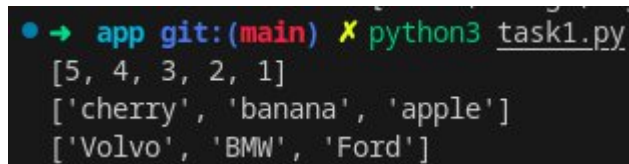
Рисунок 24– Метод remove

### Листинг 25 – Пример использования метода reverse

```
#reverse
numbers = [1, 2, 3, 4, 5]
numbers.reverse()
print(numbers)

fruits = ['apple', 'banana', 'cherry']
reversed_fruits = fruits[::-1]
print(reversed_fruits)

cars = ['Ford', 'BMW', 'Volvo']
reversed_cars = list(reversed(cars))
print(reversed_cars)
```



```

• → app git:(main) X python3 task1.py
[5, 4, 3, 2, 1]
['cherry', 'banana', 'apple']
['Volvo', 'BMW', 'Ford']
[]

```

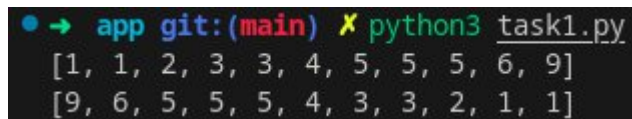
Рисунок 25 – Метод reverse

#### Листинг 26 – Пример использования метода sort

```

#sort
numbers = [3, 1, 4, 1, 5, 9, 2, 6, 5, 3, 5]
numbers.sort()
print(numbers)
numbers.sort(reverse=True)
print(numbers)

```



```

• → app git:(main) X python3 task1.py
[1, 1, 2, 3, 3, 4, 5, 5, 5, 6, 9]
[9, 6, 5, 5, 5, 4, 3, 3, 2, 1, 1]

```

Рисунок 26 – Метод sort

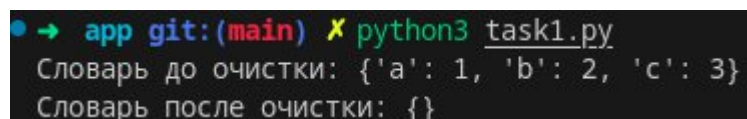
3. Используя команды `dir` и `help`, были изучены следующие методы обработки словарей: `'clear'`, `'copy'`, `'fromkeys'`, `'get'`, `'items'`, `'keys'`, `'pop'`, `'popitem'`, `'setdefault'`, `'update'`, `'values'`. Пример использования методов представлен на рисунках 27 –

#### Листинг 27 – Пример использования метода clear

```

#clear
my_dict = {'a': 1, 'b': 2, 'c': 3}
print("Словарь до очистки:", my_dict)
my_dict.clear()
print("Словарь после очистки:", my_dict)

```



```

• → app git:(main) X python3 task1.py
Словарь до очистки: {'a': 1, 'b': 2, 'c': 3}
Словарь после очистки: {}

```

Рисунок 27 – Метод clear

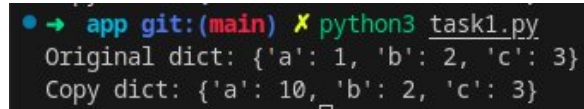
#### Листинг 28 – Пример использования метода copy

```

#copy
original_dict = {'a': 1, 'b': 2, 'c': 3}
copy_dict = original_dict.copy()

```

```
copy_dict['a'] = 10
print("Original dict:", original_dict)
print("Copy dict:", copy_dict)
```

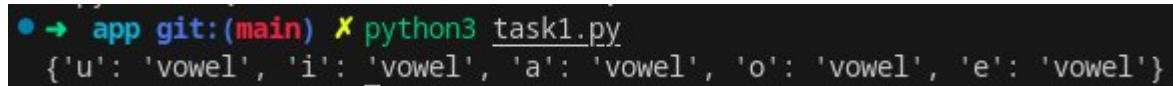


```
app git:(main) X python3 task1.py
Original dict: {'a': 1, 'b': 2, 'c': 3}
Copy dict: {'a': 10, 'b': 2, 'c': 3}
```

Рисунок 28 – Метод copy

### Листинг 29 – Пример использования метода fromkeys

```
#fromkeys
keys = {'a', 'e', 'i', 'o', 'u'}
value = 'vowel'
vowels = dict.fromkeys(keys, value)
print(vowels)
```




```
app git:(main) X python3 task1.py
{'u': 'vowel', 'i': 'vowel', 'a': 'vowel', 'o': 'vowel', 'e': 'vowel'}
```

Рисунок 29 – Метод fromkeys

### Листинг 30 – Пример использования метода get

```
#get
car = {"brand": "Ford", "model": "Mustang", "year": 1964}
x = car.get("model")
print(x)
```

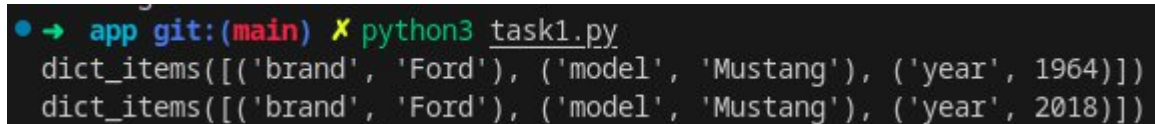


```
app git:(main) X python3 task1.py
Mustang
```

Рисунок 30 – Метод get

### Листинг 31 – Пример использования метода items

```
#items
car = {"brand": "Ford", "model": "Mustang", "year": 1964}
x = car.items()
print(x)
car["year"] = 2018
print(x)
```

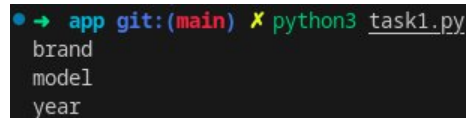


```
app git:(main) X python3 task1.py
dict_items([('brand', 'Ford'), ('model', 'Mustang'), ('year', 1964)])
dict_items([('brand', 'Ford'), ('model', 'Mustang'), ('year', 2018)])
```

Рисунок 31 – Метод items

### Листинг 32 – Пример использования метода keys

```
#keys
car = {"brand": "Ford", "model": "Mustang", "year": 1964}
for key in car.keys():
    print(key)
```



```
• → app git:(main) X python3 task1.py
brand
model
year
```

Рисунок 32 – Метод keys

### Листинг 33 – Пример использования метода pop

```
#pop
my_dict = {'a': 1, 'b': 2, 'c': 3}
removed_value = my_dict.pop('a')
print(removed_value)
print(my_dict)
```

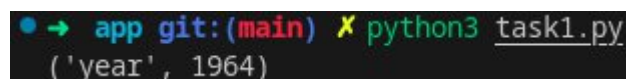


```
• → app git:(main) X python3 task1.py
1
{'b': 2, 'c': 3}
```

Рисунок 33 – Метод pop

### Листинг 34 – Пример использования метода popitem

```
#popitem
car = {"brand": "Ford", "model": "Mustang", "year": 1964}
x = car.popitem()
print(x)
```

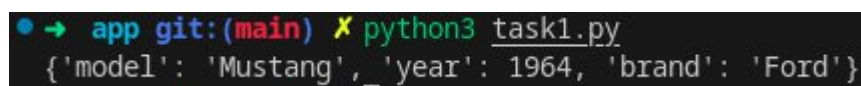


```
• → app git:(main) X python3 task1.py
('year', 1964)
```

Рисунок 34 – Метод popitem

### Листинг 35 – Пример использования метода setdefault

```
#setdefault
car = {"model": "Mustang", "year": 1964}
car.setdefault("brand", "Ford")
print(car)
```

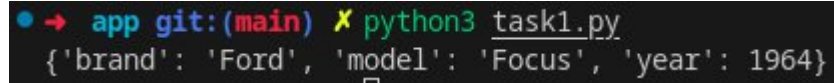


```
• → app git:(main) X python3 task1.py
{'model': 'Mustang', 'year': 1964, 'brand': 'Ford'}
```

Рисунок 35 – Метод setdefault

### Листинг 36 – Пример использования метода update

```
#update
car = {"brand": "Ford", "model": "Mustang", "year": 1964}
car.update({"model": "Focus"})
print(car)
```

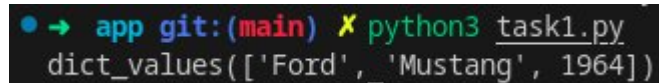


```
• → app git:(main) X python3 task1.py
{'brand': 'Ford', 'model': 'Focus', 'year': 1964}
```

Рисунок 36 – Метод update

### Листинг 37 – Пример использования метода values

```
#values
car = {"brand": "Ford", "model": "Mustang", "year": 1964}
print(car.values())
```



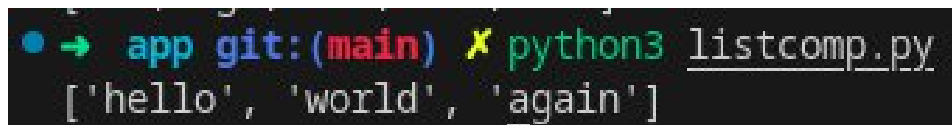
```
• → app git:(main) X python3 task1.py
dict_values(['Ford', 'Mustang', 1964])
```

Рисунок 37 – Метод values

4. Был написан код, позволяющий определить списковое включение, которое из списка строк генерирует версию нового списка, состоящего из строк, длина которых больше пяти и которые записаны символами нижнего регистра.

### Листинг 38 – Файл listcomp.py

```
string_list = ["Hello", "World", "It's", "Me", "Again"]
new_list = [str.lower() for str in string_list if len(str) >=5]
print(new_list)
```



```
• → app git:(main) X python3 listcomp.py
['hello', 'world', 'again']
```

Рисунок 38 – Результат выполнения программы

5. Была написана функция быстрой сортировки на Python, используя списки. Был использован первый элемент как точка деления списка.

### Листинг 39 – Файл quickSort.py

```
def quicksort(arr):
    if len(arr) <= 1:
        return arr
    else:
        pivot = arr[0]
        less = [x for x in arr[1:] if x <= pivot]
        greater = [x for x in arr[1:] if x > pivot]
        return quicksort(less) + [pivot] + quicksort(greater)

arr = [6, 2, 5, 7, 31, 16, 45, -12, 24, -54, 2, 4]
sorted_arr = quicksort(arr)
print(sorted_arr)
```

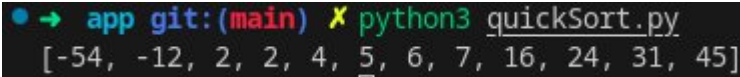


Рисунок 39 – Результат выполнения программы

6. Была написана функция сложения двух переменных на языке Python. Корректность написанной функции была проверена при помощи автооценивателя. Все тесты прошли проверку, что продемонстрировано на рисунке 40.

### Листинг40 – Функция сложения двух переменных

```
def add(a, b):
    "Возвращает сумму a и b"
    print("a = %s, b = %s => a + b = %s"%( a, b, (a+b)))
    return a+b
```

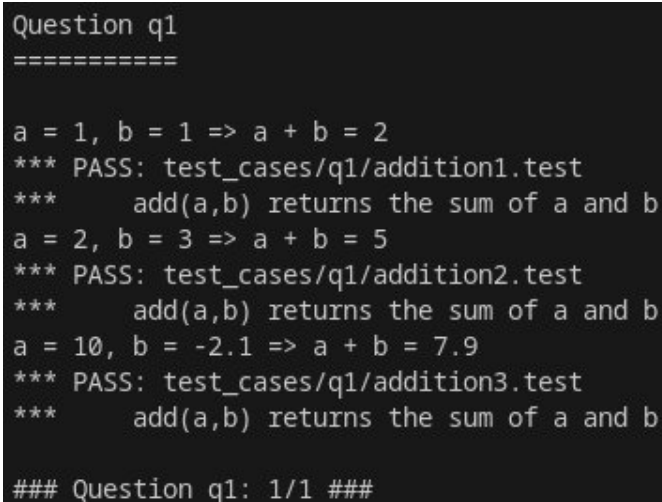


Рисунок 40 – Результат прохождения тестов



7. Была написана функция `buyLotsOfFruit(orderList)` в файле `buyLotsOfFruit.py`, которая принимает список-заказ из кортежей `(fruit, pound)` и возвращает стоимость заказа. Все написанные тесты успешно прошли проверку, результат выполнения функции представлен на рисунке 41.

#### Листинг41 – Функция `buyLotsOfFruit(orderList)`

```
def buyLotsOfFruit(orderList):  
    """  
        orderList: Список-заказ из кортежей (fruit, numPounds)  
  
        Возвращает стоимость заказа  
    """  
    totalCost = 0.0  
  
    for fruit, pound in orderList:  
        if fruit in fruitPrices:  
            totalCost += fruitPrices[fruit] * pound  
        else:  
            print("Error: Missing data about %s"%fruit)  
  
    return totalCost
```

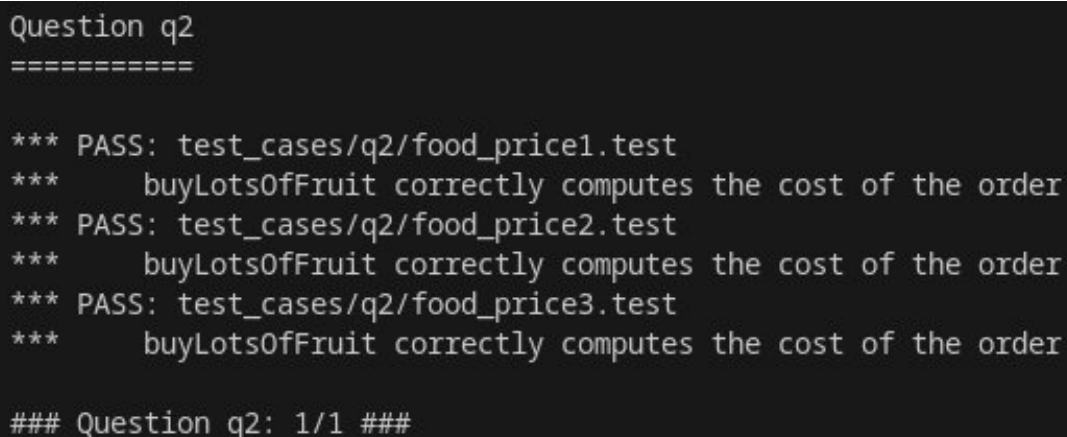
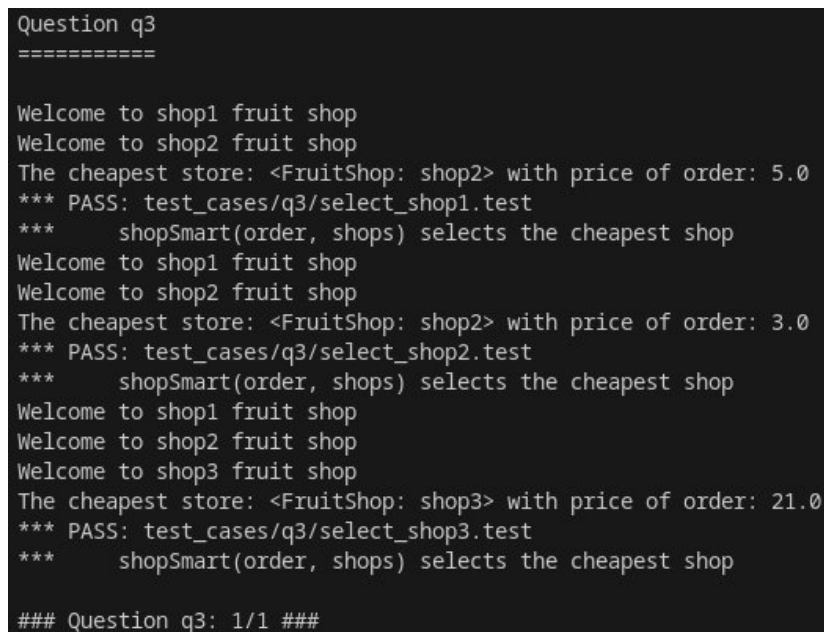
A screenshot of a terminal window showing the output of an automated testing tool. The text is as follows:  
Question q2  
=====  
  
\*\*\* PASS: test\_cases/q2/food\_price1.test  
\*\*\* buyLotsOfFruit correctly computes the cost of the order  
\*\*\* PASS: test\_cases/q2/food\_price2.test  
\*\*\* buyLotsOfFruit correctly computes the cost of the order  
\*\*\* PASS: test\_cases/q2/food\_price3.test  
\*\*\* buyLotsOfFruit correctly computes the cost of the order  
  
### Question q2: 1/1 ###  
The screenshot has a dark background with light-colored text.

Рисунок 41 – Результат автооценителя для функции  
`buyLotsOfFruits(orderList)`

8. Была написана функция `shopSmart(orderList, fruitShops)` в файле `shopSmart.py`, которая принимает список заказов `orderList` и список магазинов `FruitShops`, и возвращает магазин `FruitShop`, с наименьшей стоимостью заказа. Результат успешного прохождения тестов при помощи автооценителя представлен на рисунке 42.

## Листинг 42 – Функция shopSmart(orderList, fruitShop)

```
def shopSmart(orderList, fruitShops):
    """
    Возвращает магазин с минимальной стоимостью заказа
    orderList: Список-заказ из кортежей (fruit, numPound)
    fruitShops: Список магазинов типа shop
    """
    minimum = float("inf")
    cheapest = None
    for shop in fruitShops:
        price = shop.getPriceOfOrder(orderList)
        if (price < minimum):
            minimum = price
            cheapest = shop
    print("The cheapest store: %s with price of order: %s"%(shop, minimum))
    return cheapest
```



```
Question q3
=====

Welcome to shop1 fruit shop
Welcome to shop2 fruit shop
The cheapest store: <FruitShop: shop2> with price of order: 5.0
*** PASS: test_cases/q3/select_shop1.test
*** shopSmart(order, shops) selects the cheapest shop
Welcome to shop1 fruit shop
Welcome to shop2 fruit shop
The cheapest store: <FruitShop: shop2> with price of order: 3.0
*** PASS: test_cases/q3/select_shop2.test
*** shopSmart(order, shops) selects the cheapest shop
Welcome to shop1 fruit shop
Welcome to shop2 fruit shop
Welcome to shop3 fruit shop
The cheapest store: <FruitShop: shop3> with price of order: 21.0
*** PASS: test_cases/q3/select_shop3.test
*** shopSmart(order, shops) selects the cheapest shop

### Question q3: 1/1 ###
```

Рисунок 42 – Результат автооценителя для функции shopSmart(orderList, fruitShop)

## Выводы

В ходе выполнения лабораторной работы были изучены технологии подготовки и выполнения программ на языке Python, исследованы свойства функций языка Python, используемых при обработке последовательностей, сформированы навыки написания программ работы с классами на языке Python.