



Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное
учреждение высшего образования
«Московский государственный технический университет имени
Н. Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н. Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

ОТЧЕТ

по Лабораторной работе №7

по курсу «Анализ Алгоритмов»

на тему: «Поиск по словарю»

Студент группы ИУ7-56Б

(Подпись, дата)

Мансуров В. М.
(Фамилия И.О.)

Преподаватель

(Подпись, дата)

Волкова Л. Л.
(Фамилия И.О.)

Преподаватель

(Подпись, дата)

Строганов Ю. В..
(Фамилия И.О.)

Москва — 2023 г.

Содержание

Введение	3
1 Аналитическая часть	4
1.1 Словарь как структура данных	4
1.2 Алгоритм полного перебора	5
1.3 Требования к программе	5
2 Конструкторская часть	7
2.1 Требования к программному обеспечению	7
2.2 Описание используемых типов данных	7
2.3 Разработка алгоритмов	7
2.4 Вывод	8
3 Технологическая часть	9
3.1 Средства реализации	9
3.2 Сведения о модулях программы	9
3.3 Реализация алгоритмов	9
4 Исследовательская часть	14
4.1 Формализация объекта и его признака	14
4.2 Анкетирование	15
4.3 Построение функции принадлежности термам	18
4.4 Тестирование	19
Заключение	22
Список использованных источников	23

Введение

В процессе развития компьютерных систем количество обрабатываемых данных увеличивалось, вследствие чего множество операций над наборами данных стали выполняться очень долго, поскольку чаще всего это был обычный перебор. Это вызвало необходимость создать новые алгоритмы, которые решают поставленную задачу на порядок быстрее стандартного решения прямого обхода. В том числе это касается и словарей, в которых одной из основных операций является операция поиска.

Целью данной лабораторной работы является получение навыков поиска по словарю при ограничении на значение признака, заданном при помощи лингвистической переменной..

Для поставленной цели необходимо выполнить следующие задачи:

- 1) формализовать объект по варианту и его признак;
- 2) составить анкета для заполнения респондентом;
- 3) провести анкетирование респондентов;
- 4) построить функцию принадлежности термам числовых значений признака, описываемого лингвистической переменной, на основе статистической обработки мнений респондентов, выступающих в роли экспертов;
- 5) описать алгоритм поиска в словаре объектов;
- 6) описать структуру данных словаря;
- 7) реализовать описанный алгоритм поиска в словаре;
- 8) описать и обосновать результаты в виде отчета о выполненной лабораторной работе, выполненном как расчётно-пояснительная записка к работе.

1 Аналитическая часть

В данном разделе будут рассмотрены словарь как структура данных и алгоритм полного перебора, а также представлены требования к разрабатываемой программе.

1.1 Словарь как структура данных

Словарь [1] — абстрактный тип данных (интерфейс к хранилищу данных), позволяющий хранить пары вида «(ключ, значение)» и поддерживающий операции добавления пары, а также поиска и удаления пары по ключу:

- 1) $insert(k, v)$;
- 2) $find(k)$;
- 3) $remove(k)$.

В паре (k, v) : v называется значением, ассоциированным с ключом k . Где k — это ключ, а v — значение. Семантика и названия вышеупомянутых операций в разных реализациях ассоциативного массива могут отличаться.

Операция поиска $find(k)$ возвращает значение, ассоциированное с заданным ключом, или некоторый специальный объект, означающий, что значения, ассоциированного с заданным ключом, нет. Две другие операции ничего не возвращают (за исключением, возможно, информации о том, успешно ли была выполнена данная операция).

Словарь с точки зрения интерфейса удобно рассматривать как обычный массив, в котором в качестве индексов можно использовать не только целые числа, но и значения других типов — например, строки (именно по этой причине словарь также иногда называют «ассоциативным массивом»).

1.2 Алгоритм полного перебора

Алгоритмом полного перебора [2] называют метод решения задачи, при котором по очереди рассматриваются все возможные варианты. В случае реализации алгоритма в рамках данной работы будут последовательно перебираться ключи словаря до тех пор, пока не будет найден нужный.

Трудоёмкость алгоритма зависит от того, присутствует ли искомый ключ в словаре, и, если присутствует — насколько он далеко от начала массива ключей. Пусть на старте алгоритм затрагивает k_0 операций, а при сравнении k_1 операций.

Пусть алгоритм нашёл элемент на первом сравнении (лучший случай), тогда будет затрачено $k_0 + k_1$ операций, на втором — $k_0 + 2 \cdot k_1$, на последнем (худший случай) — $k_0 + N \cdot k_1$. Если ключа нет в массиве ключей, то мы сможем понять это, только перебрав все ключи, таким образом трудоёмкость такого случая равно трудоёмкости случая с ключом на последней позиции. Трудоёмкость в среднем может быть рассчитана как математическое ожидание по формуле (1.1), где Ω — множество всех возможных случаев.

$$\sum_{i \in \Omega} p_i \cdot f_i = k_0 + k_1 \cdot \left(1 + \frac{N}{2} - \frac{1}{N+1}\right) \quad (1.1)$$

1.3 Требования к программе

К разрабатываемой в данной работе программе предъявляется ряд требований:

- 1) на вход будет подаваться строка, на основании которой производится поиск;
- 2) на выходе — результат поиска в словаре;

- 3) программа не должна аварийно завершаться при отсутствии ключа в словаре.

Вывод

В данном разделе были рассмотрены словарь как структура данных и алгоритм полного перебора, а также приведены требования к разрабатываемой программе.

2 Конструкторская часть

В этом разделе будут представлено описание используемых типов данных, а также схемы алгоритмов поиска в словаре.

2.1 Требования к программному обеспечению

К программе предъявлены ряд требований:

- иметь интерфейс ввода вопроса;
- работа с словарями и массивами;
- работа со строками.

2.2 Описание используемых типов данных

При реализации алгоритмов будут использованы следующие типы данных:

- 1) словарь — встроенный тип `dict` [3] в Python[4] будет использован в созданном классе `Dictionary`;
- 2) массив ключей — встроенный тип `list` [5] в Python[4];
- 3) длина массива/словаря — целое число `int`.

2.3 Разработка алгоритмов

На рисунке 2.1 представлена схема алгоритма поиска в словаре полным перебором.

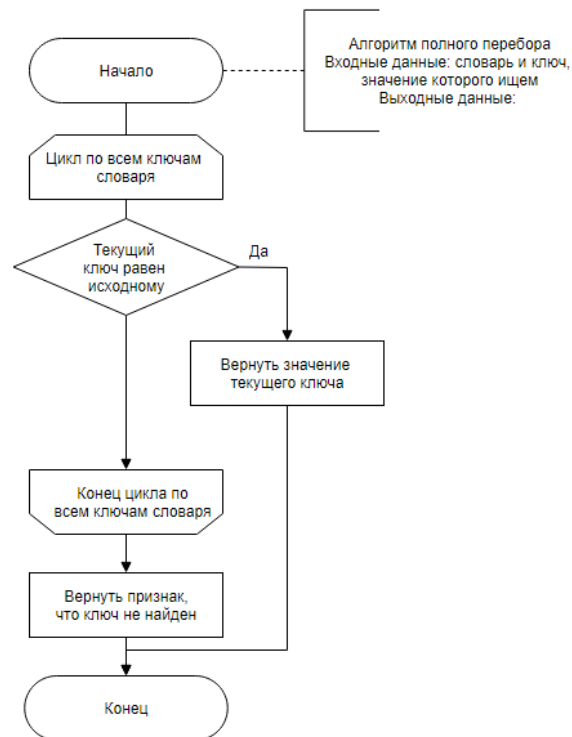


Рисунок 2.1 – Схема алгоритма поиска в словаре полным перебором

2.4 Вывод

В данном разделе была построена схема алгоритма, рассматриваемого в лабораторной работе, были описаны классы эквивалентности для тестирования, структура программы.

3 Технологическая часть

В данном разделе рассмотрены средства реализации, а также представлены листинги реализаций алгоритма расчета термовой частота для всех термов из выборки документов.

3.1 Средства реализации

В данной работе для реализации был выбран язык программирования *Python* [4]. Данный выбор обусловлен соответствием с требованиями выдвинутыми в конструкторской части, а именно в языке программирования *Python* имеются встроенные типы данных — словарь и массив и инструменты для поиска подстроки в строке.

3.2 Сведения о модулях программы

Данная программа разбита на следующие модули:

Программа состоит из четырех модулей:

- 1) *main.py* — файл, содержащий точку входа;
- 2) *utils.py* — файл, содержащий служебные алгоритмы;
- 3) *terms.py* — файл, содержащий термы и константы программы;
- 4) *dish.py* — файл, содержащий код класса *Dish*, формализуемого рассматриваемый объект.

3.3 Реализация алгоритмов

В листингах 3.1 – 3.4 приведены реализации алгоритма выделения наиболее информативных терминов для каждого документа. В качестве

термов в данной реализации рассматриваются слова, состоящие из латинских букв. В качестве документов рассматриваются строки, состоящие из таких слов, пробелов и знаков пунктуации.

Листинг 3.1 – Реализация проверки запроса на корректность

```
1 def check_request(req):
2     request = req.lower().split()
3     res = 0
4
5     not_valid = True
6     for word in request:
7         if damLev(word, "блюдо") < 3:
8             not_valid = False
9             break
10
11     if not_valid:
12         print("В запросе должна идти речь о блюдах")
13         return 0
14
15     if "калорийность" not in req and \
16         "калорийности" not in req:
17         print("В запросе должна идти речь о калорийности блюдах")
18         return 0
19
20     res = check_term(req)
21
22     return res
```

Листинг 3.2 – Реализация поиска термина

```
1 def check_term(req):
2     result = NOT_FOUND
3     if 'очень некалорийной' in req or \
4         'очень некалорийная' in req:
5         result = VERY_NOT_CALORIE
6     elif 'некалорийная' in req or \
7         'некалорийной' in req:
8         result = NOT_CALORIE
9     elif 'малая' in req or \
10         'малой' in req:
11         result = LOW_CALORIE
12     elif 'средняя' in req or \
13         'средней' in req:
14         result = MEDIUM_CALORIE
15     elif 'высокая' in req or \
16         'высокой' in req:
17         result = HIGH_CALORIE
18     elif 'не съем' in req:
19         result = NOT_EAT
20     elif 'очень не высокая' in req or \
21         'очень не высокой' in req:
22         result = VERY_NOT_HIGH_CALORIE
23     elif 'не очень высокая' in req or \
24         'не очень высокой' in req:
25         result = NOT_VERY_HIGH_CALORIE
26     else:
27         print("Речь должна идти о каком-то из термов!")
28     return result
```

Листинг 3.3 – Реализация поиска интервала значений для объекта по терму

```
1 def get_interval(term):
2     is_not = False
3     interval = []
4     if term > 7:
5         is_not = True
6
7     if term == VERY_NOT_CALORIE:
8         interval = [10, 25]
9     elif term == NOT_CALORIE:
10        interval = [26, 96]
11    elif term == LOW_CALORIE:
12        interval = [97, 294]
13    elif term == MEDIUM_CALORIE:
14        interval = [295, 512]
15    elif term == HIGH_CALORIE:
16        interval = [513, 954]
17    elif term == NOT_EAT:
18        interval = [955, 2000]
19    elif term == NOT_VERY_HIGH_CALORIE:
20        interval = [10, 112]
21    elif term == VERY_NOT_HIGH_CALORIE:
22        interval = [263, 523]
23
24    return interval, is_not
```

Листинг 3.4 – Реализация алгоритма поиска полным перебором

```
1 def search(data , border):  
2     res = []  
3     for key in data.keys():  
4         if data[key].calories >= border[0] and  
5             data[key].calories <= border[1]:  
6             res.append(data[key])  
7     return res
```

Вывод

В данном разделе был представлен листинг рассматриваемого алгоритма поиска в словаре, приведена информация о средствах реализации, сведения о модулях программы.

4 Исследовательская часть

В данном разделе приведены постановка эксперимента.

4.1 Формализация объекта и его признака

Согласно согласованному варианту, формализуем объект «блюда» следующим образом: определим набор данных и признак объекта, на основании которого составим набор термом.

Набор данных для объекта:

- название блюда — строка;
- тип блюда — строка;
- родина блюда — строка.

Согласно варианту, числовым признаком, по которому будет производиться поиск объектов, является калорийность. Калорийность выражается количеством калорий на 100 грамм блюда — вещественное число.

Определим следующие термы, соответствующие признаку «калорийность»:

- 1) очень не калорийное (сокр. ОН);
- 2) не калорийное (сокр. Н);
- 3) малой (сокр. М);
- 4) средней (сокр. С);
- 5) высокой(сокр. В);
- 6) очень не высокой(сокр. ОНВ);
- 7) не очень высокой (сокр. НОВ);
- 8) «не съем».

Также введем универсальное для задачи множество оцениваемой величины (калорийность) K :

$$K = \{10, 20, 30, 40, 50, 100, 150, 200, 250, 300, 350, 400, 450, 500, 550, 600, 650, 700, 750, 800, 850, 900, 950, 1000\} \quad (4.1)$$

4.2 Анкетирование

Было проведено анкетирование следующих респондентов:

- 1) Назиров Илхом, группа ИУ7-55Б — Респондент 1;
- 2) Калашников Сергей, группа ИУ7-53Б — Респондент 2;
- 3) Лубянская Анастасия, группа ИУ7-56Б — Респондент 3;
- 4) Шагалов Вячеслав, группа ИУ7-51Б — Респондент 4;
- 5) Морозов Дмитрий, группа ИУ7-52Б — Респондент 5;
- 6) Нарандаев Дамир, группа ИУ7-52Б — Респондент 6;
- 7) Вязовцев Максим, группа ИУ7-55Б — Респондент 7;
- 8) Загайнов Никита, группа ИУ7-52Б — Респондент 8.

Респонденты, выступающие в качестве экспертов, для каждого из приведенных выше термов указали соответствующий промежуток, элементами которого являются числа из введенного для поставленной задачи множества оцениваемой величины.

Результаты анкетирования перечисленных респондентов продемонстрированы в таблицах 4.1 – 4.2. В данных таблицах термы соответствуют обозначенным в 4.1 сокращенным термам.

Таблица 4.1 – Результаты анкетирования (Часть 1)

Респондент	Терм, ккал/100 гр					
	ОН	Н	М	С	В	не съем
1	[10; 30)	[30; 100)	[100; 300)	[300; 550)	[550; 850)	>850
2	[40; 100)	[10; 40)	[100; 400)	[400; 600]	[600; 1000)	>1000
3	[10; 100)	[100; 250)	[250; 350)	[350; 600]	[600; 1000]	>1000
4	10	20	(20; 150)	[150; 550)	[550; 1000)	>1000
5	10	(10; 40)	[40; 50)	[50; 150)	[150; 650)	>650
6	[10; 30)	[30; 50)	[50; 100)	[100; 200)	[200; 550)	>550
7	[10; 50)	[50; 200)	[200; 350)	[350; 550)	[500; 1000]	>1000
8	[10; 40)	[40; 150)	[150; 300)	[300; 450)	[450; 600)	>600

Таблица 4.2 – Результаты анкетирования (Часть 2)

Респондент	Терм, ккал/100 гр	
	ОНВ	НОВ
1	[10; 150)	[150; 850)
2	[10; 100)	[500; 700)
3	[10; 350)	[350; 600)
4	[10; 350)	[350; 600)
5	[10; 150)	[200; 400)
6	[10; 100)	[350; 550)
7	[10; 350)	[350; 600)
8	[10; 100)	[100; 400)

Результаты анкетирования перечисленных респондентов для каждого значения множества K продемонстрированы в таблицах 4.3 – 4.4. В данных таблицах термы соответствуют обозначенным в 4.1 сокращенным термам. В ячейках таблиц хранятся количество респондентов указавший значение, которое будет соответствовать терму.

Таблица 4.3 – Результаты анкетирования по значениям множества K (Часть 1)

Терм	Калории, ккал/100 гр													
	10	20	30	40	50	100	150	200	250	300	350	400	450	500
ОН	7	5	4	3	2	0	0	0	0	0	0	0	0	0
Н	1	3	4	3	3	3	2	1	0	0	0	0	0	0
М	0	0	1	2	2	3	3	4	5	3	1	0	0	0
С	0	0	0	0	1	2	2	1	1	3	5	6	5	5
В	0	0	0	0	0	0	1	2	2	2	2	2	3	4
ОНВ	8	8	8	8	8	5	3	3	3	3	0	0	0	0
НОВ	0	0	0	0	0	1	2	3	3	3	7	5	5	6
не съем	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Таблица 4.4 – Результаты анкетирования по значениям множества K (Часть 2)

Терм	Калории, ккал/100 гр										
	550	600	650	700	750	700	850	900	950	1000	>1000
ОН	0	0	0	0	0	0	0	0	0	0	0
Н	0	0	0	0	0	0	0	0	0	0	0
М	0	0	0	0	0	0	0	0	0	0	0
С	2	0	0	0	0	0	0	0	0	0	0
В	4	6	5	5	5	5	4	4	4	3	2
ОНВ	0	0	0	0	0	0	0	0	0	0	0
НОВ	5	2	2	1	1	1	0	0	0	0	0
не съем	0	1	2	3	3	3	3	4	4	8	8

4.3 Построение функции принадлежности термам

Для этого для каждого значения из K для каждого термина из перечисленных найдем количество респондентов, согласно которым значение из H удовлетворяет сопоставляемому терму. Данное значение поделим на количество респондентов — это и будет значением функции t для термина в точке. Графики функций принадлежности числовых значений калорийности термам, приведен на рисунке 4.1.

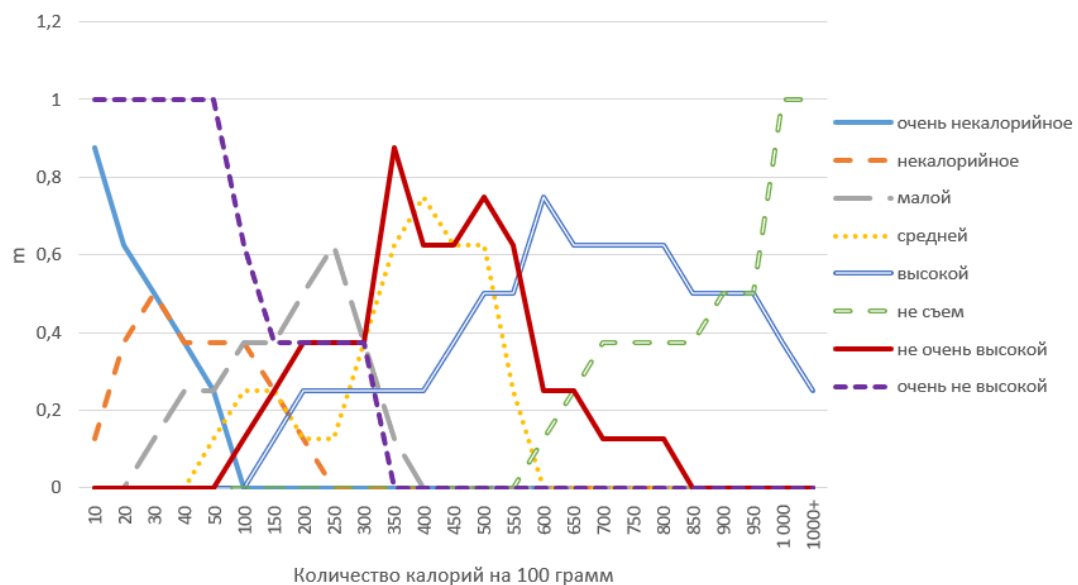


Рисунок 4.1 – Графики функций принадлежности числовых значений переменной термам, описывающим группы значений лингвистической переменной

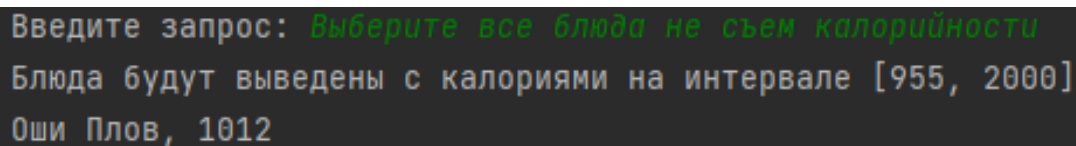
В соответствии с полученным графиком будем считать блюда:

- 1) очень некалорийной, если их значение их калорийность лежит в промежутке $[10; 25]$ калорий на 100 грамм;
- 2) некалорийной, если их значение их калорийность лежит в промежутке $[26; 96]$ калорий на 100 грамм;
- 3) малой калорийности, если их значение их калорийность лежит в промежутке $[96; 294]$ калорий на 100 грамм;

- 4) средней калорийности, если их значение их калорийность лежит в промежутке [295; 512] калорий на 100 грамм;
- 5) высокой калорийности, если их значение их калорийность лежит в промежутке [513; 954] калорий на 100 грамм;
- 6) «не съем» калорийности, если их значение их калорийность больше 955 калорий на 100 грамм;
- 7) очень не высокой калорийности, если их значение их калорийность лежит в промежутке [10; 112] калорий на 100 грамм;
- 8) не очень высокой калорийности, если их значение их калорийность лежит в промежутке [263; 523] калорий на 100 грамм;

4.4 Тестирование

В этом разделе представлены запросы пользователя и результаты их обработки.



```
Введите запрос: Выберите все блюда не съем калорийности
Блюда будут выведены с калориями на интервале [955, 2000]
Оши Плов, 1012
```

Рисунок 4.2 – Запрос 1

Введите запрос: *Выберите все блюда у которых малая калорийность*
Блюда будут выведены с калориями на интервале [97, 294]
Ассорти мясное, 264
Баклажаны по-корейски готовые, 109
Винегрет, 135
Афинский салат, 128
Салат Берендей, 205
Салат Бордо, 214
Салат Вита, 205
Грибной салат , 156
Грибной салат, 156
Салат Жемчужина, 103
Халвайтар, 205.4
Маства, 135.7
Курутоб, 205.4
Манты, 210.97
Дамлама, 100.9
Ношхухпа, 130.9
Фатир, 245
Любиева, 150
Мастобай гелакдор, 250
Эчпочмак, 238
Манная каша, 98
Калитки, 157.1
Калач, 249
Песочные кольца с арахисом, 220
Каравай, 276
Курник, 272.18
Шарлотка, 145.7
Голубцы, 110
Пельмени, 275

Рисунок 4.3 – Запрос 2

Введите запрос: *Какие блюда средней калорийности*
Блюда будут выведены с калориями на интервале [295, 512]
Венский салат, 299
Самбуса Вараки, 368
Адыгейский сыр, 420
Перловка, 352
Коврижка, 309
Пастила, 324
Макароны по-флотски, 450
Бефстроганов, 355.4

Рисунок 4.4 – Запрос 3

Введите запрос: *Выберите все блюда людой калорийности*
Выберите все блюда людой калорийности
Речь должна идти о каком-то из термов!

Рисунок 4.5 – Запрос 4

Введите запрос: *Выберите все блюда России*
В запросе должна идти речь о калорийности блюдах

Рисунок 4.6 – Запрос 5

Заключение

Поставленная цель достигнута: получен навык поиска по словарю при ограничении на значение признака, заданного при помощи лингвистической переменной.

В ходе выполнения лабораторной работы были решены все задачи:

- 1) формализован объект по варианту и его признак;
- 2) составлена анкета для заполнения респондентом;
- 3) проведено анкетирование респондентов;
- 4) построена функцию принадлежности термам числовых значений признака, описываемого лингвистической переменной, на основе статистической обработки мнений респондентов, выступающих в роли экспертов;
- 5) описан алгоритм поиска в словаре объектов;
- 6) описана структуру данных словаря;
- 7) реализован описанный алгоритм поиска в словаре;
- 8) полученные результаты описаны в виде отчета о выполненной лабораторной работе, выполненном как расчётно-пояснительная записка к работе.

Список использованных источников

- 1 С. Шапошникова. Словари — Лаборатория линуксоида [Электронный ресурс]. — Режим доступа:
<https://younglinux.info/python/dictionary> (дата обращения: 28.01.2023).
- 2 Н. Нильсон. Искусственный интеллект. Методы поиска решений. М.: Мир, 1973. — С. 273.
- 3 dict Python [Электронный ресурс]. Режим доступа: <https://docs.python.org/3/library/2to3.html?highlight=dict#to3fixer-dict> (дата обращения: 28.01.2023).
- 4 Welcome to Python [Электронный ресурс]. Режим доступа:
<https://www.python.org> (дата обращения: 28.01.2023).
- 5 list Python [Электронный ресурс]. Режим доступа: <https://docs.python.org/3/library/pdb.html?highlight=list#pdbcommand-list> (дата обращения: 28.01.2023).