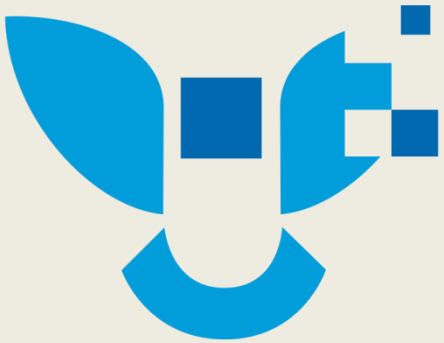
Вятский государственный университет
Цифровые кафедры
Разработка прикладного программного обеспечения

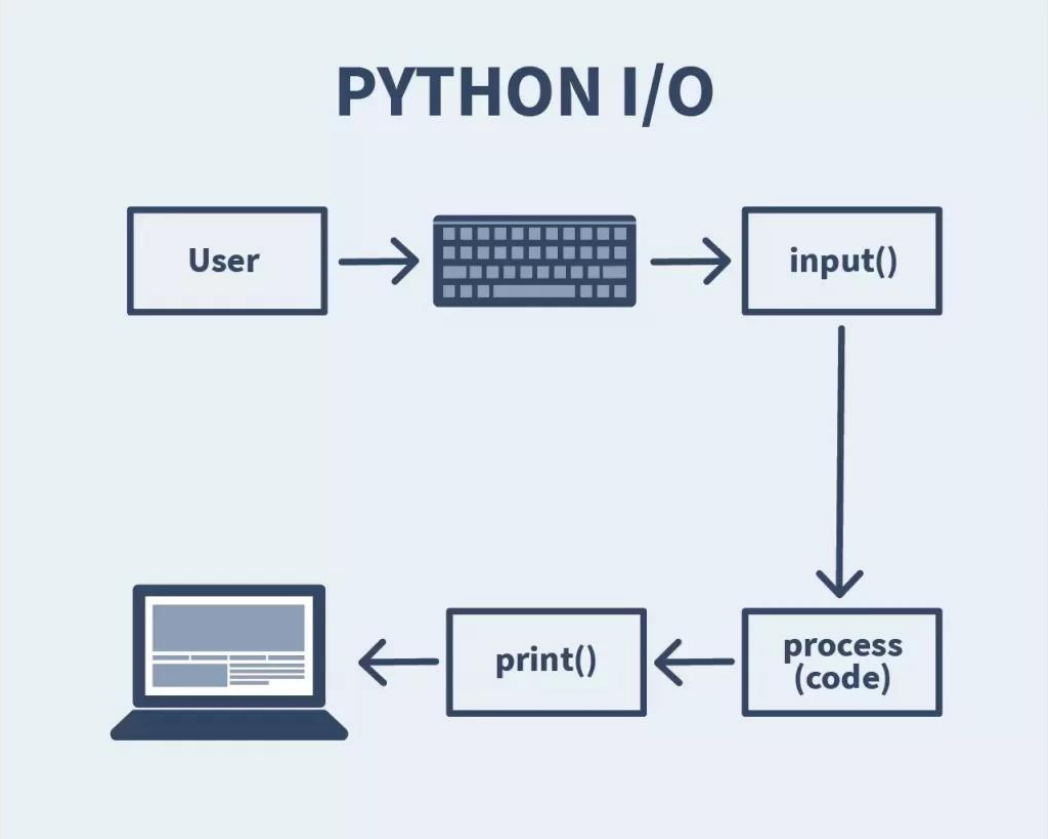
Открыть ящик ПИТОНА

Долженкова Мария Львовна,
кандидат технических наук, заведующий кафедрой ЭВМ
Нижегородова Маргарита Владимировна,
кандидат педагогических наук, доцент кафедры САУ
Шмакова Наталья Александровна,
старший преподаватель кафедры САУ

Дата 18.10.2022



Ввод/вывод в программе

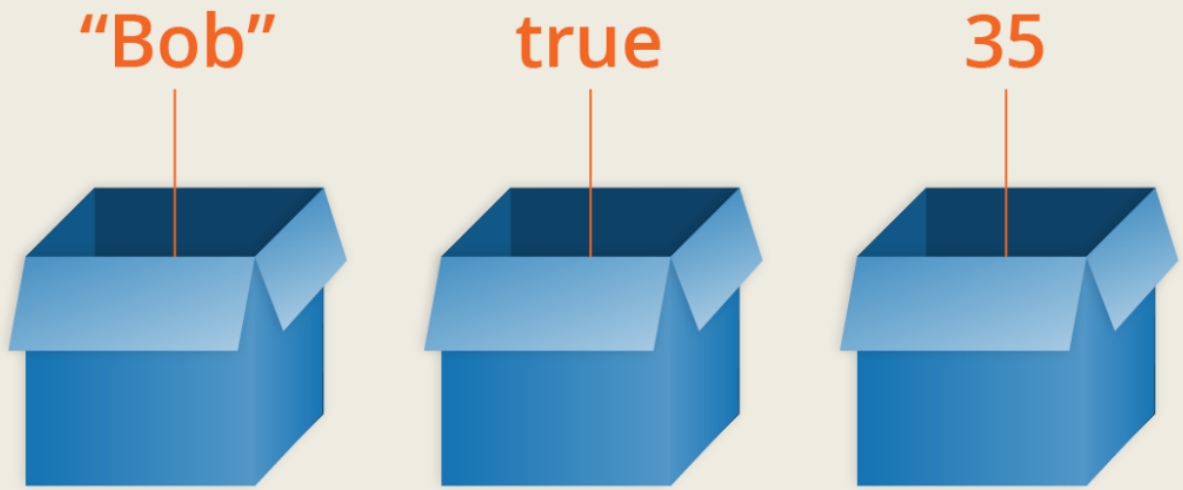


- `input()` – ввод данных
- `print()` – вывод данных в консоль

Example

Поговорим о данных

Тип данных определяет множество значений, набор операций, которые можно применять к таким значениям и способ реализации хранения значений и выполнения операций.



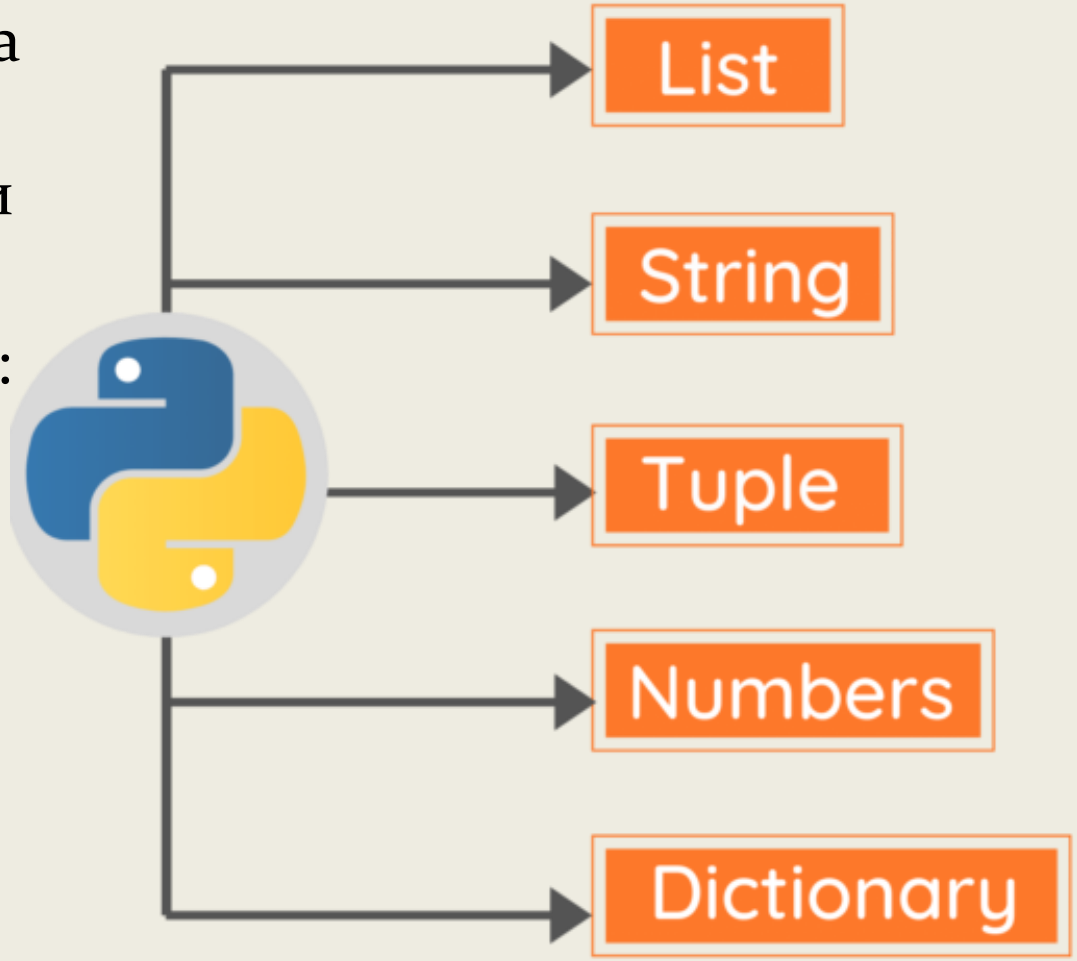
Динамическая типизация

- Сильная типизация не позволяет производить операции в выражениях с данными различных типов, нельзя складывать, например, строки и числа, нужно все приводить к одному типу.
- При динамической типизации тип переменной определяется непосредственно при выполнении программы, то есть одна и та же переменная, при многократной инициализации, может связываться с объектами разных типов.

```
a = 10 # переменная a ссылается на целое число
a = 'hello' # переменная a теперь ссылается на строку
```

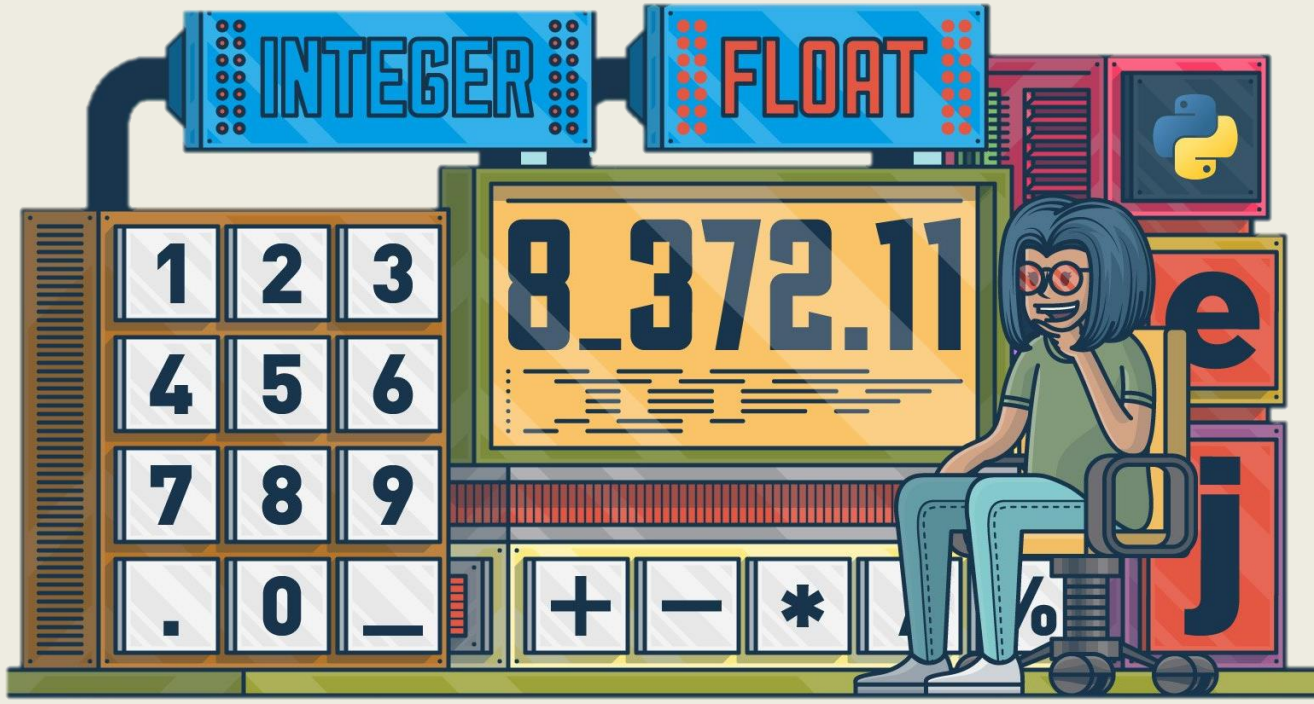
Типы данных

- В Python типы данных можно разделить на встроенные и невстроенные, которые можно использовать при импортировании соответствующих модулей.
- К основным встроенным типам относятся:
  - Логические переменные
  - Числа
  - Списки
  - Кортежи
  - Строки
  - Словари



Числа

- **int** – целое число
- **float** – число с плавающей точкой
- **complex** – комплексное число. Состоит из двух чисел с плавающей точкой, представляющих соответственно его действительную и мнимую части.

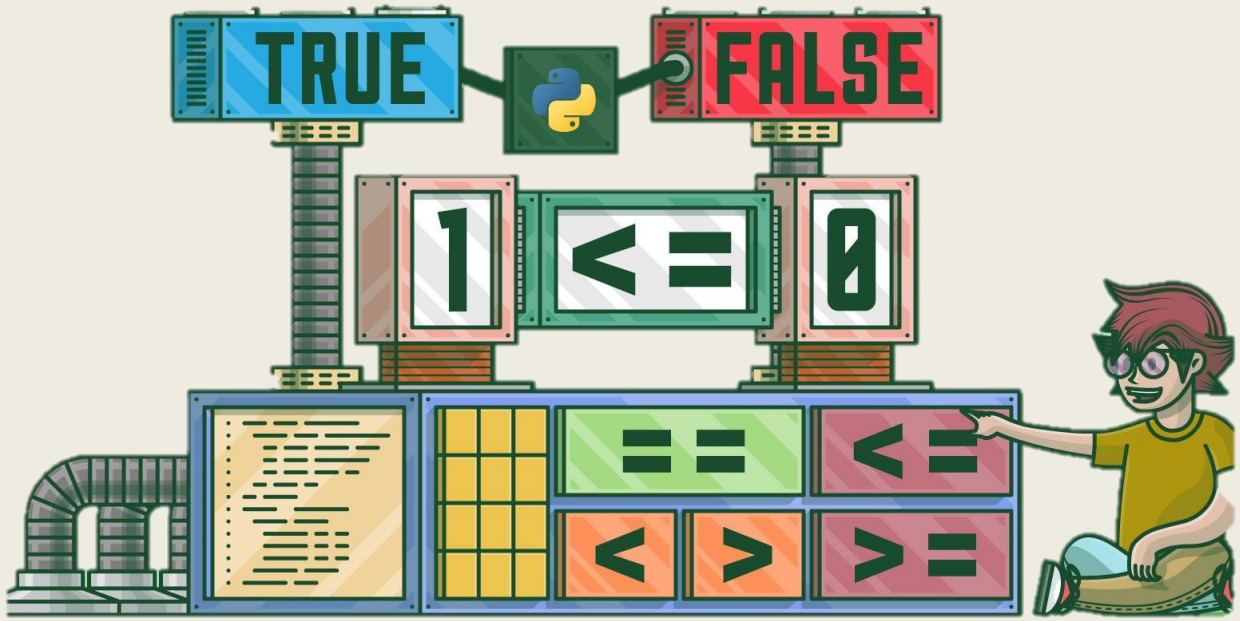



```
z = complex(1, 2)
print(z) # вывод на экран: (1 + 2j)
print(z.real) # вывод на экран: (1.0)
print(z.imag) # вывод на экран: (2.0)
```

- Example1
- Example1_1

Логические переменные

- Логические переменные (bool). Примитивный тип данных, который принимает два значения – истина (True) или ложь (False).
- True и False ведут себя как числа 1 и 0.
- Используются в качестве результата логических операций.



Списки

– Это структура данных для хранения объектов различных типов, напоминающая массив. Каждому объекту соответствует индекс (место) в списке.

– Длина списка может изменяться за счет добавления или извлечения объектов. Пустой список имеет нулевую длину. Первый элемент списка имеет индекс 0, последний – (n - 1), где n – длина списка.

- list1 = [] # пустой список
- list2 = [1, 'two', 3.2]

Example 2

Кортежи

– tuple – кортеж. Представляет последовательность элементов, которая во многом похожа на список за тем исключением, что мы не можем добавлять, удалять и изменять элементы в кортеже.

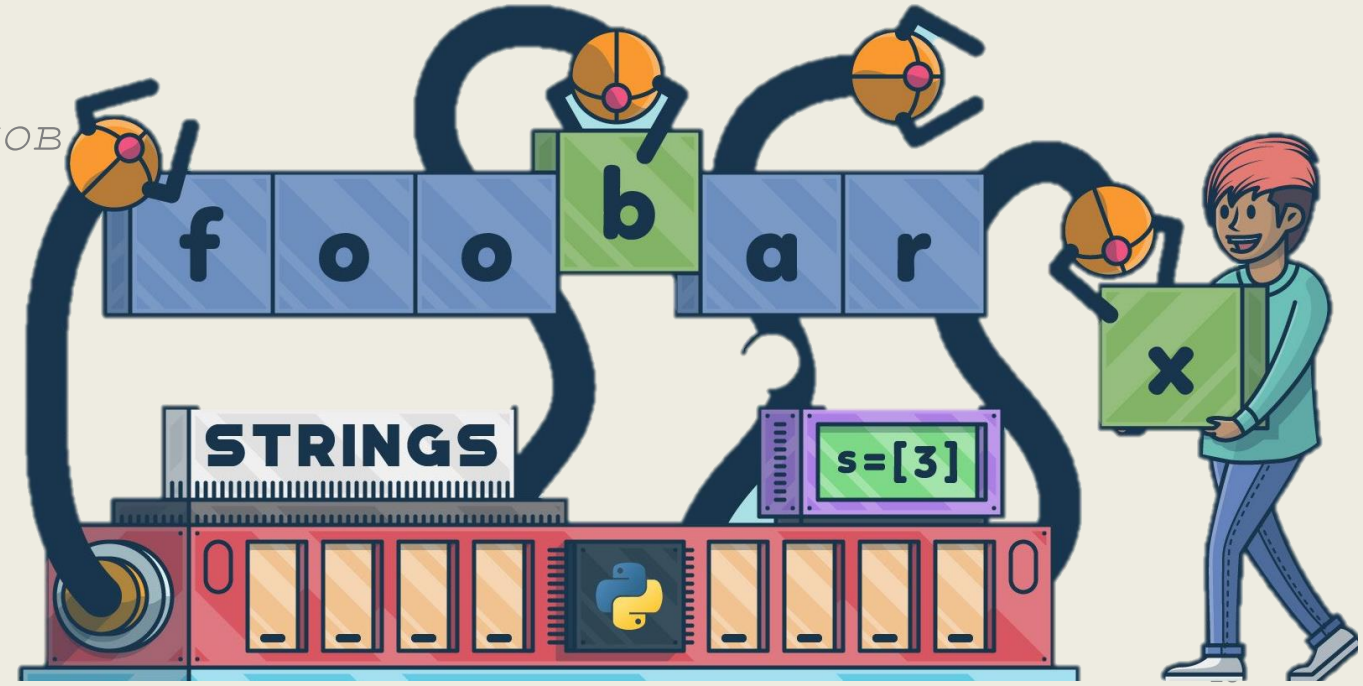
- tuple1 = (1, 'two', [1, 2]) # кортеж из трех объектов: целого числа 1, строки 'two' и списка из двух элементов.
- s = 'today'
- tuple2 = tuple(s) # кортеж ('t', 'o', 'd', 'a', 'y')

Example3

Строки

Это упорядоченная последовательность символов, которая предназначена для хранения информации в виде простого текста. К символам строки, как и к элементам списка, можно обращаться по индексу, первый символ имеет индекс 0.

```
s = '' # пустая строка
s = 'today' # строка их 5 символов
# s[0] – символ 't'
# s[3] – символ 'a'
```

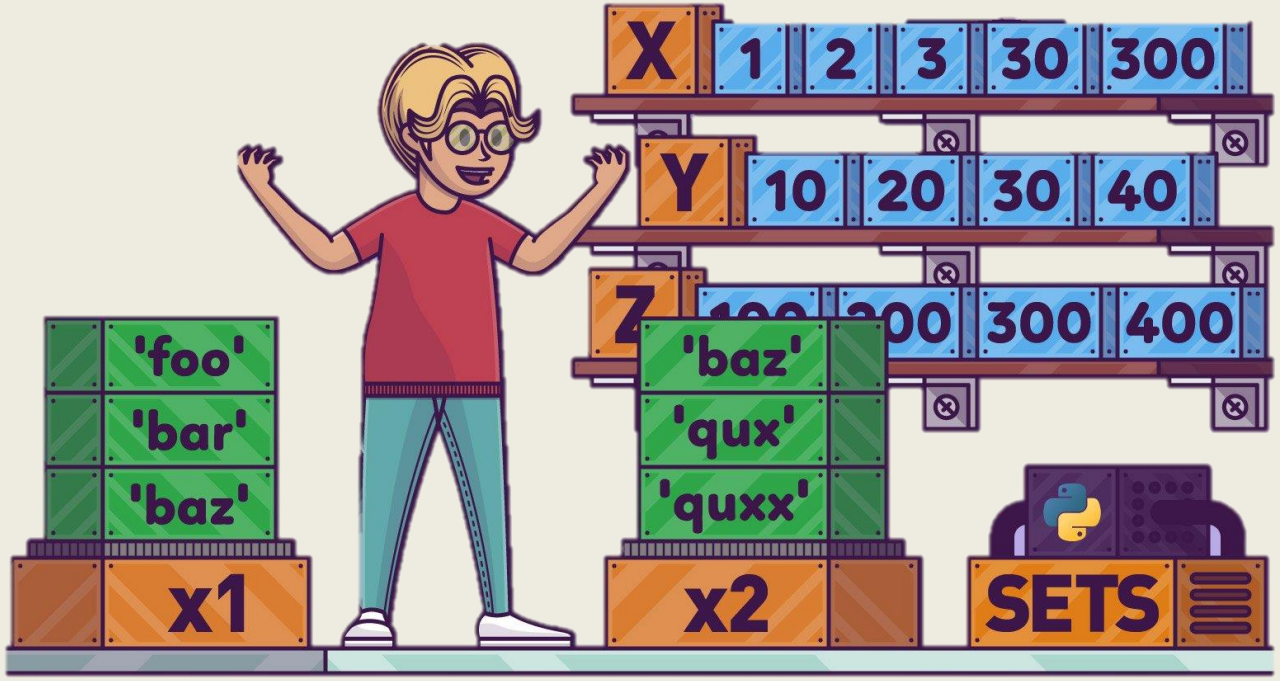


Множества

Множества – set. Это набор неупорядоченных уникальных (неповторяющихся) объектов, объединенных по какому-либо признаку. Возможны два варианта задания множества.

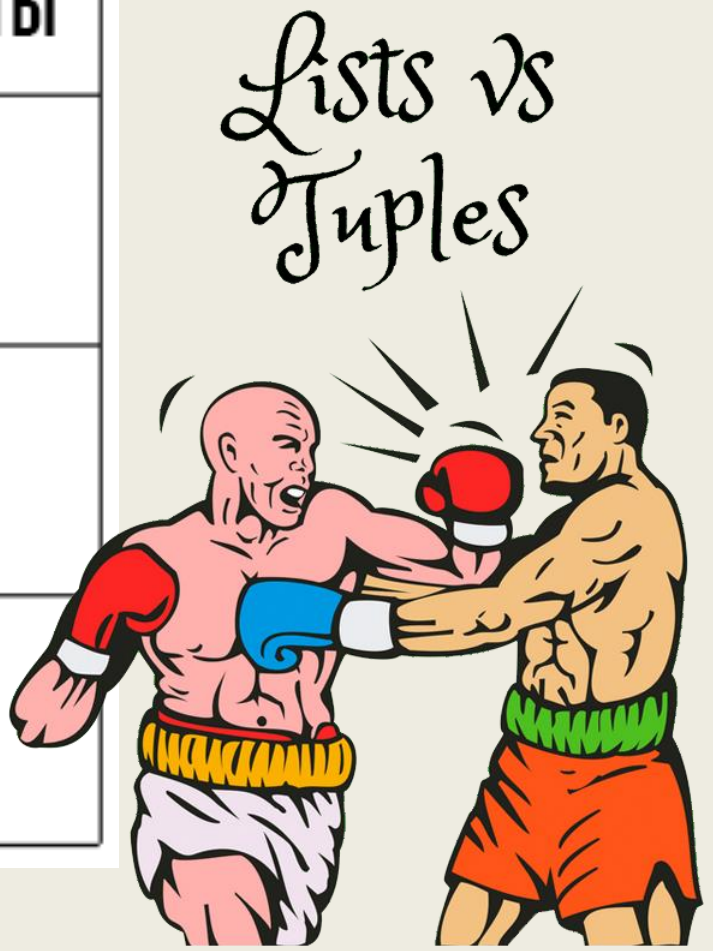
```
# множество из трех элементов
set1 = {'1', '2', '3'}
l = ['1', '2', '3']
set2 = set(l)
```

Множество – это неупорядоченный набор, порядок символов каждый раз может меняться!



В чем разница?

	ИЗМЕНЕНИЕ	ПОРЯДОК	ИНДЕКСАЦИЯ	ДУБЛИКАТЫ
List	✓	✓	✓	✓
Tuple	✗	✓	✓	✓
Set	✓	✗	✗	✗

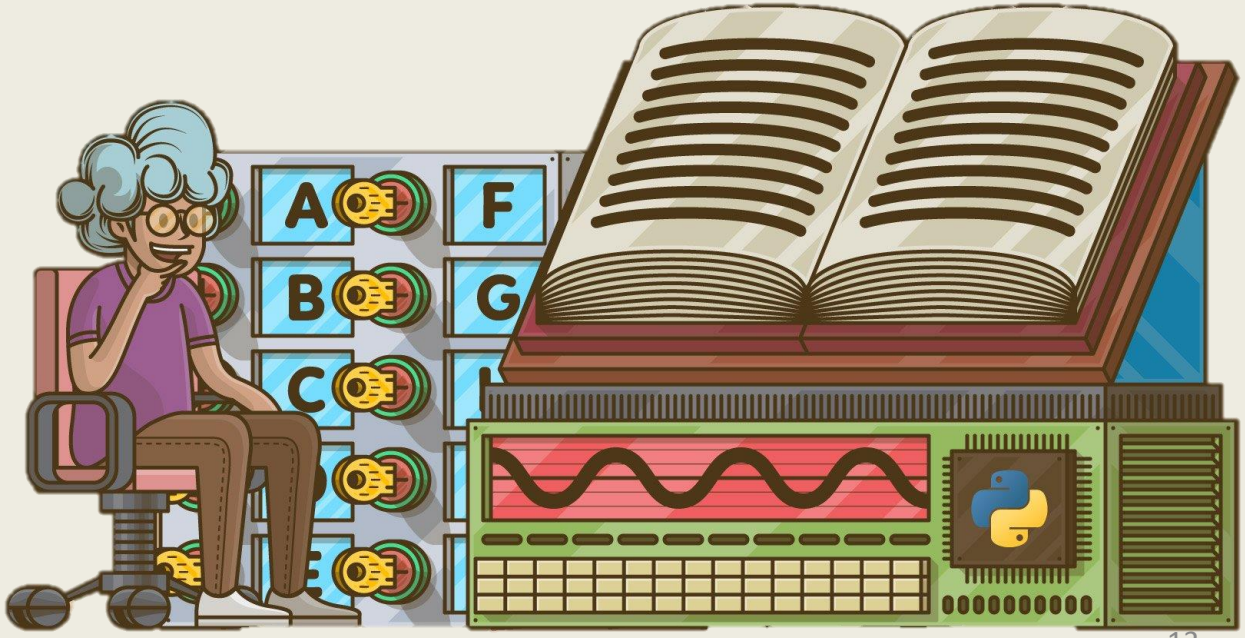


Словари

Является набором данных, однако не считается последовательностью, так как представляет собой неупорядоченный набор пар ключ-значение.

При этом каждый ключ в рамках одного словаря является уникальным. Объявим и инициализируем словарь из трех элементов и выведем на экран значение третьего элемента.

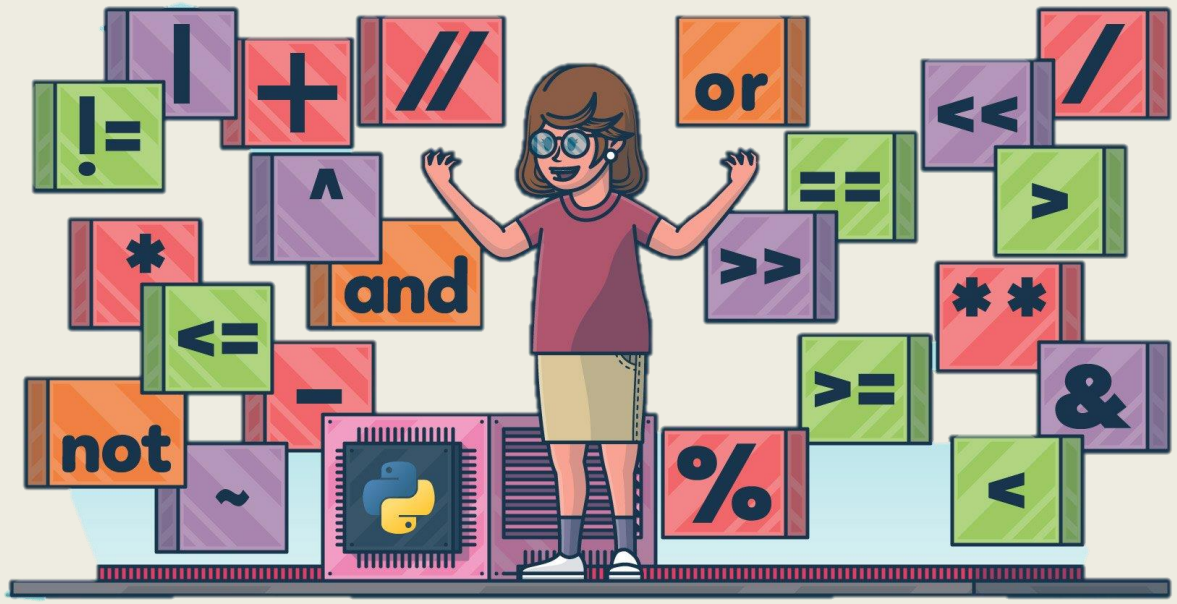
Размер словаря может изменяться за счет добавления или извлечения элементов.



Example6

Арифметические операции

- Операторы - специальные символы, которые выполняют арифметические и логические вычисления.
- Значения, на которые действует оператор, называются операндами
- Арифметические операторы используются для выполнения математических операций – сложения, вычитания, умножения и так далее.



Арифметические операции

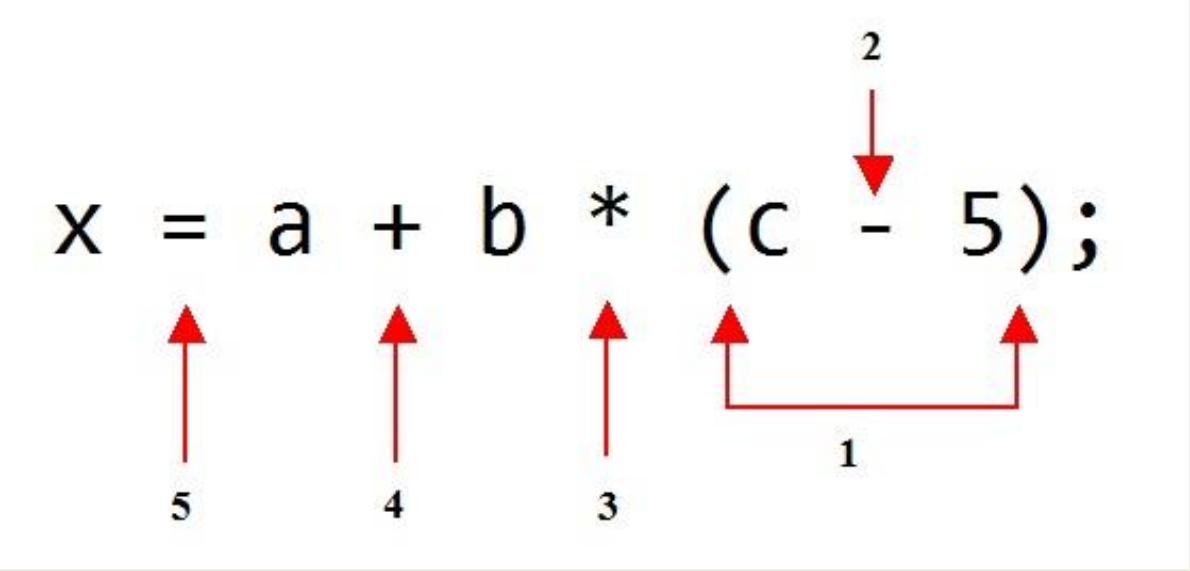
+	Сложение двух операндов
-	Вычитание правого оператора из левого
*	Умножение двух операндов
/	Деление левого операнда на правый
%	Остаток от деления левого операнда на правый
//	Деление с округлением
**	Показатель степени – левый операнд возводится

Приоритет операций

Приоритет определяет то, в какой последовательности должны выполняться операции.

Приоритет можно (и нужно) задавать круглыми скобками.

1. скобки
2. **(возведение в степень)
3. *, /, //, %
4. +, -



Example 11, 11_1

Встроенные функции для работы с числами

- Python имеет довольно обширную библиотеку математических функций, представленную в модуле math, который можно импортировать в программу.
- `abs(x)` Вычисляет модуль числа x
- `round(x)` Округляет x до ближайшего целого
- `min(x1, x2,… ,x_n)` Находит минимальное, среди указанных чисел
- `max(x1, x2,… ,x_n)` Находит максимальное, среди указанных чисел
- `pow(x, y)` Возводит x в степень y

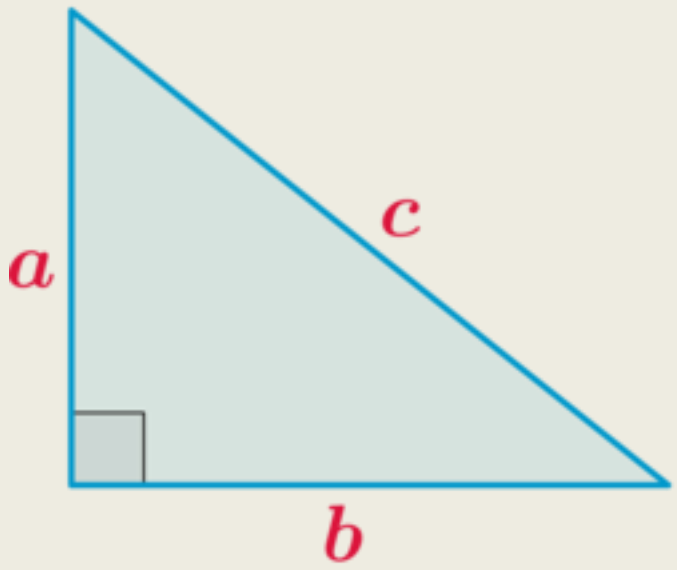
Example 12

Особые операторы

- `in` и not `in` – операторы принадлежности в Python. Они проверяют, есть ли значение или переменная в последовательности (строке, списке, кортеже, множестве или словаре). Иначе говоря, проверяют вхождение элемента в набор данных.
- **`in`** Если значение или переменная есть в последовательности возвращает значение **True**
- **not in** Если значения или переменной нет в последовательности возвращает значение **True**

Задача 1

- *Задача 1.* В двух строках вводятся два числа (числа целые, положительные, не превышают 1000). Это катеты треугольника. Найдите гипотенузу этого треугольника.



$$c^2 = a^2 + b^2$$

Задача 2

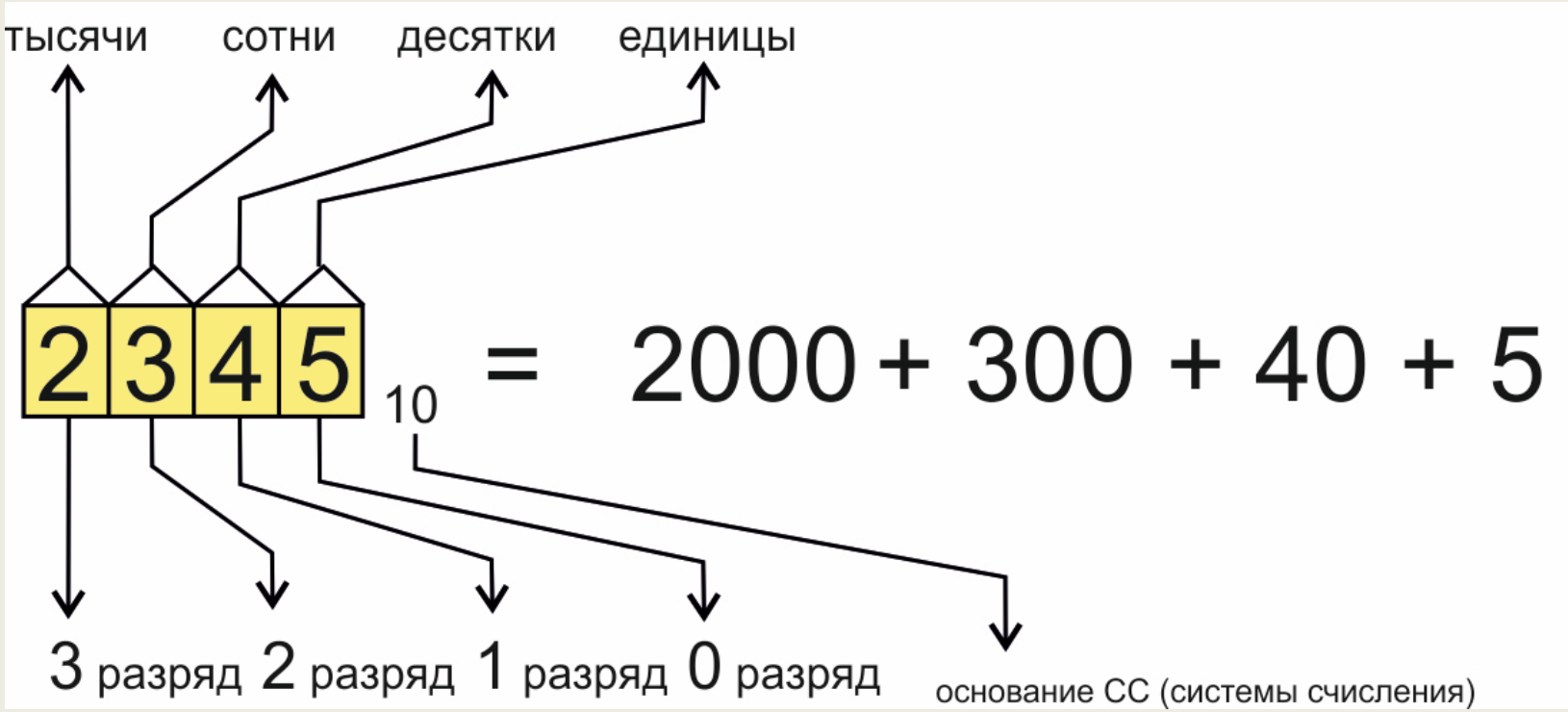
- *Задача 2*. Напишите программу, которая считывает целое число и выводит текст: сначала фразу "*The next number for the number* ", затем введенное число, затем слово " is ", окруженное пробелами, затем следующее за введенным число, наконец, знак точки без пробела.
- Аналогично в следующей строке для предыдущего числа. Пробелы, знаки препинания, заглавные и строчные буквы важны!

Задача 3

- *Задача 3*. N школьников делят K яблок поровну, неделящийся остаток остается в корзинке. Сколько яблок останется в корзинке? Сколько яблок достанется каждому школьнику?

Задача 4

- *Задача 4*. Дано неотрицательное целое число. Найдите число десятков в его десятичной записи (то есть вторую справа цифру его десятичной записи).



Задача 5

- *Задача 5*. Электронные часы показывают время в формате h:mm:ss, то есть сначала записывается количество часов, потом обязательно двузначное количество минут, затем обязательно двузначное количество секунд. Количество минут и секунд при необходимости дополняются до двузначного числа нулями. С начала суток прошло n секунд. Выведите, что покажут часы.

Задача 6

- *Задача 6*. Дано четырехзначное число. Определите, является ли его десятичная запись симметричной. Если число симметричное, то выведите 1, иначе выведите любое другое целое число.

Что можно делать со строками?

- Для работы со строками в Питоне предусмотрены операторы и специальные функции. Рассмотрим их.
- Оператор «+» – это соединение двух строк или конкатенация. Конкатенация строк означает соединение строк вместе от первого до последнего символа для создания новой строки.
- В Python довольно просто выполняется дублирование строки.

```
one = 'ай ‘
msg = one * 10
print(msg)
```

Example 20

Есть ли строка в подстроке?

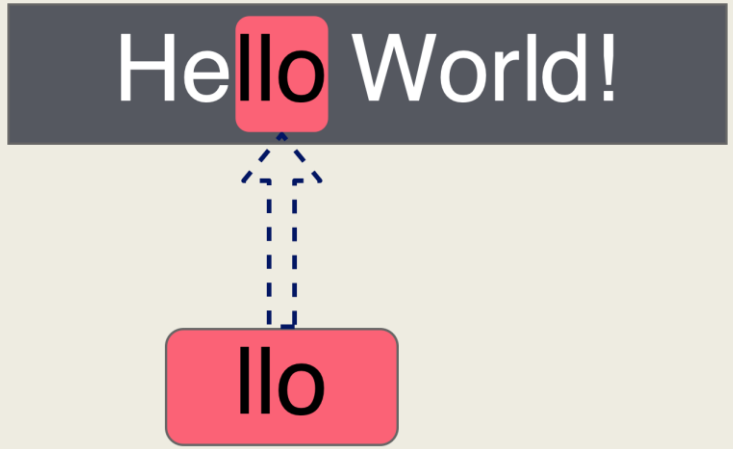
- Оператор **`in`**. Он возвращает True, если подстрока присутствует и False, если отсутствует.
- Для сравнения строк используются операторы сравнения «==» (равно), «!=» (неравно), «<» (меньше), «>» (больше). При этом в строках рассматриваются по очереди отдельные символы и их регистр:

– цифра условно меньше, чем любая буква из алфавита;
– алфавитная буква в верхнем регистре меньше, чем буква в нижнем регистре;
– чем раньше буква в алфавите, тем она меньше.

'1' < ‘A’

'A' < ‘a’

'a' < 'b'

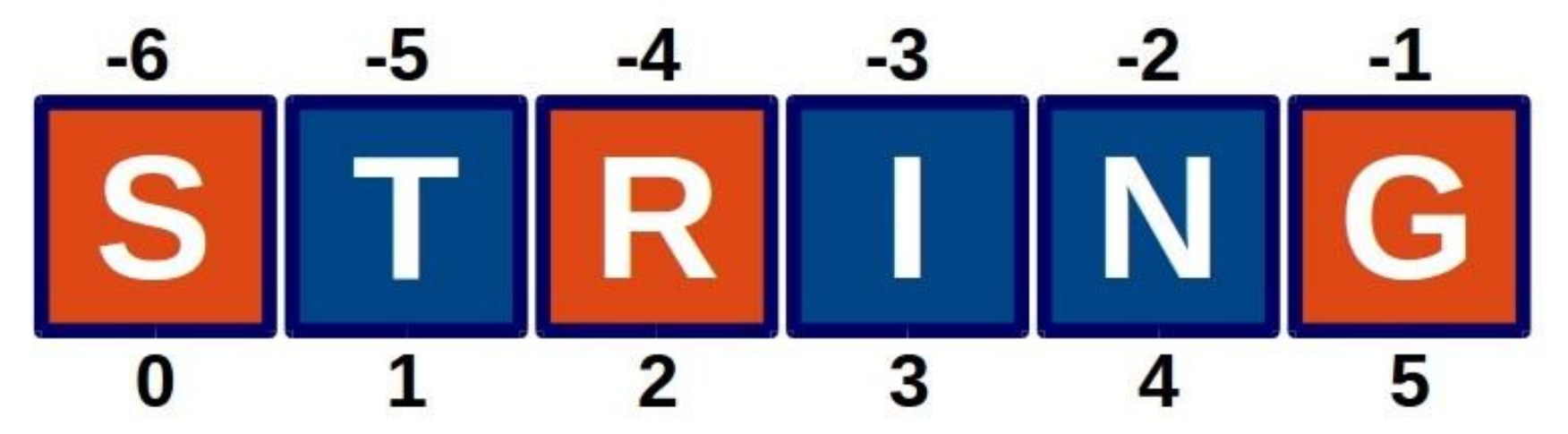


Функции для работы со строками

- Функция **`str(n)`** преобразует значение другого типа к строке.
- **`len(s)`** - определения длины строки.
- **`chr(s)`** - получение символа по его коду.
- **`ord(s)`** – для получение кода ASCII по символу.

Что нужно знать про строки?

- Строка – упорядоченная последовательность и у каждого символа в строке есть свой порядковый номер.
- Если указать неверный индекс, например, str1[12], то возникнет ошибка.
- Отрицательные индексы означают движение по строке с конца в начало.

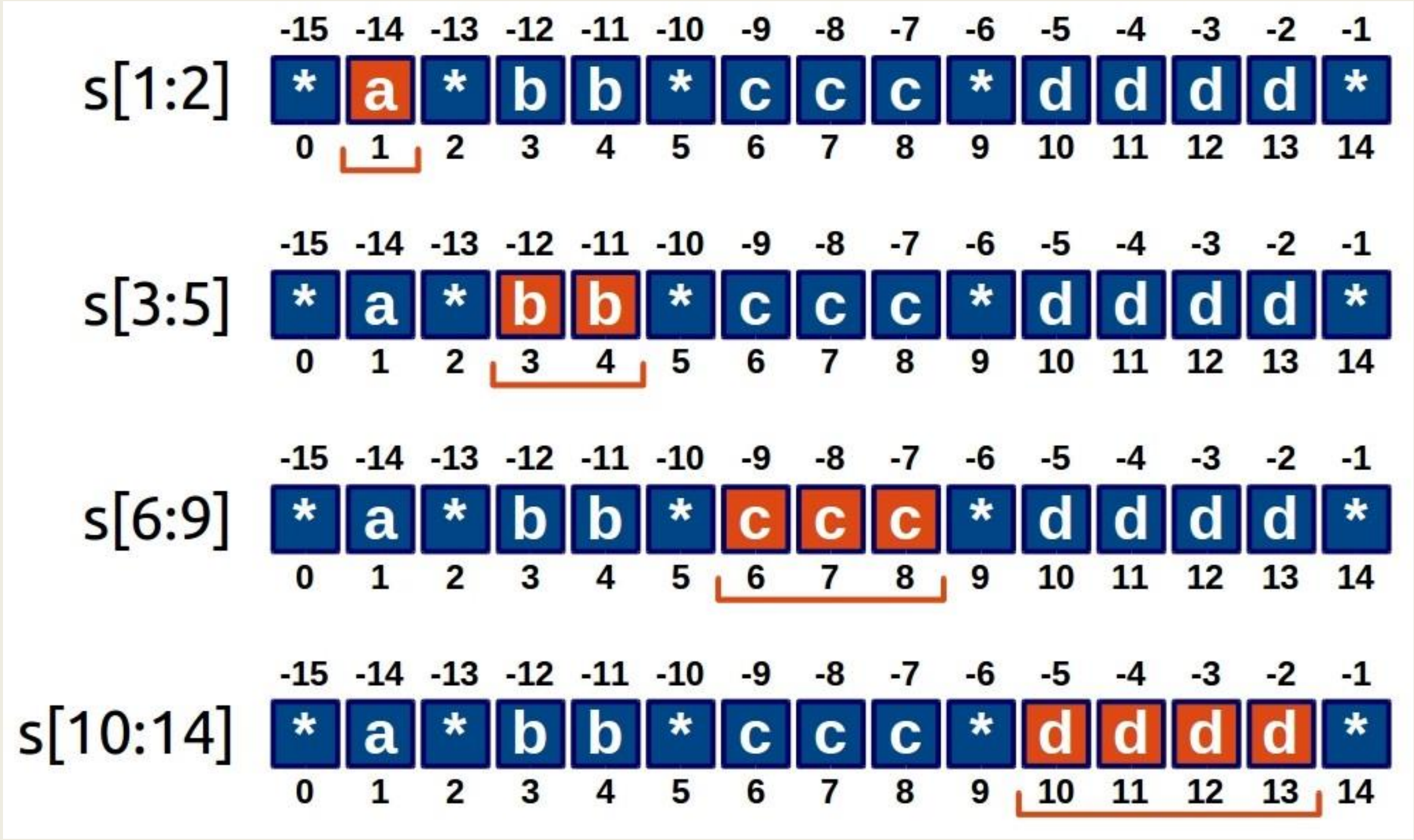


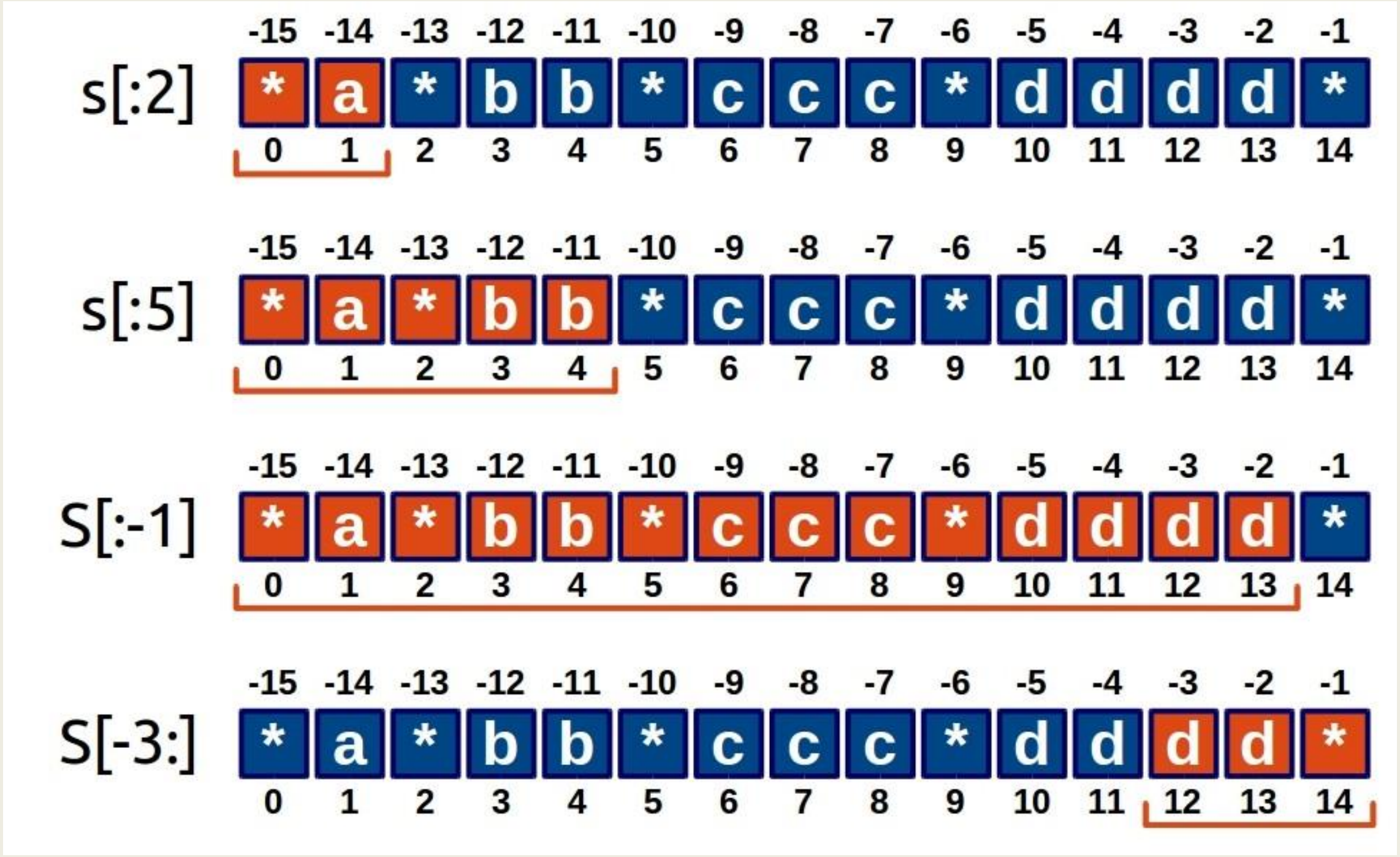
Срезы

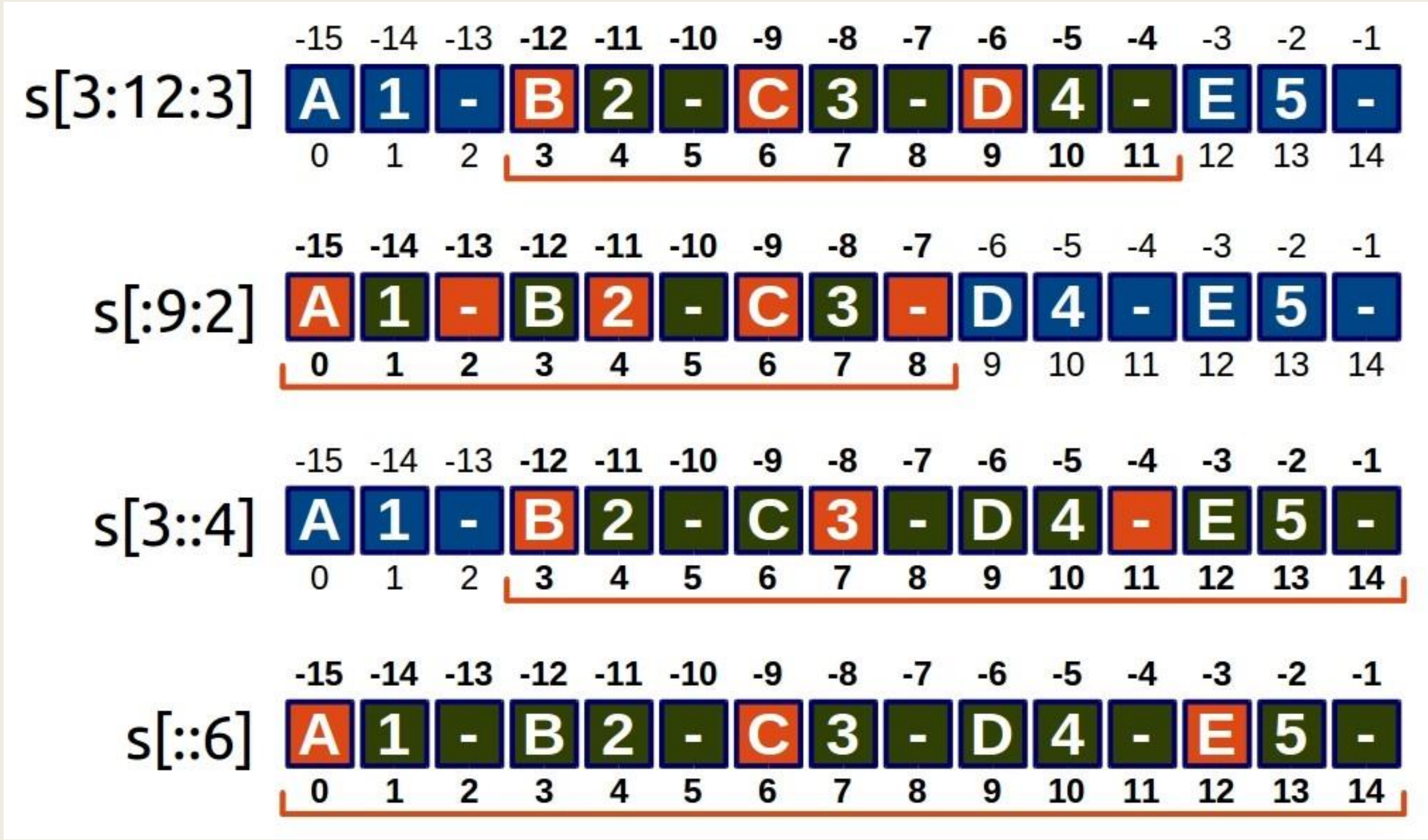
Срезы для строк в Python – это механизм, с помощью которого извлекается подстрока по указанным параметрам: начальный индекс (start), конечный индекс (stop) и шаг (step), которые указываются через символ «:».

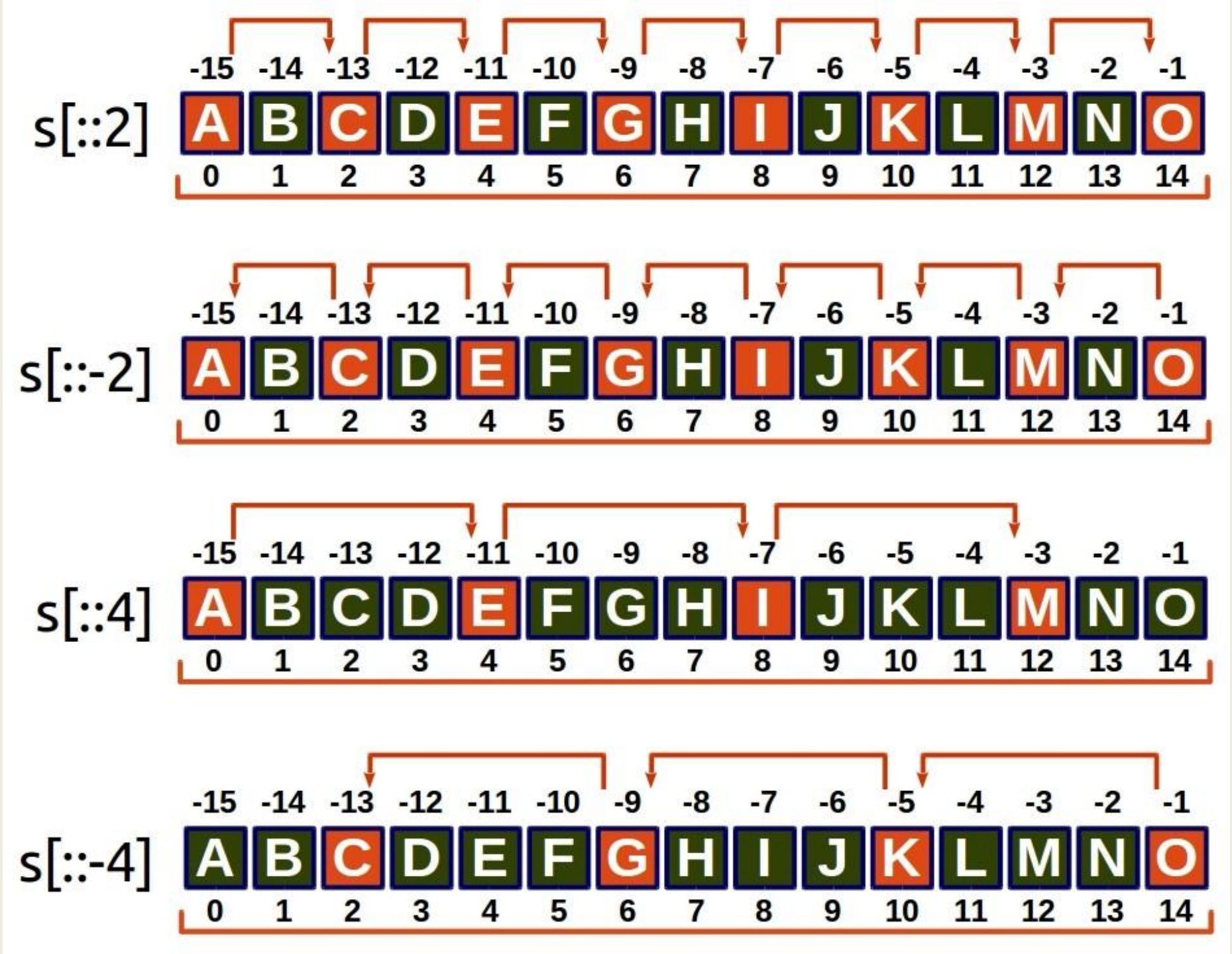
Срезы очень – мощный механизм с большим количеством вариаций, например, если не указать вторую границу, то извлечение произойдет до конца строки, то же самое с первой границей (началом строки).

[start:stop:[step]]


s[1:2]
-15 -14 -13 -12 -11 -10 -9 -8 -7 -6 -5 -4 -3 -2 -1
* a * b b * c c c * d d d d *
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14
s[3:5]
-15 -14 -13 -12 -11 -10 -9 -8 -7 -6 -5 -4 -3 -2 -1
* a * b b * c c c * d d d d *
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14
s[6:9]
-15 -14 -13 -12 -11 -10 -9 -8 -7 -6 -5 -4 -3 -2 -1
* a * b b * c c c * d d d d *
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14
s[10:14]
-15 -14 -13 -12 -11 -10 -9 -8 -7 -6 -5 -4 -3 -2 -1
* a * b b * c c c * d d d d *
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14


s[:2]
-15 -14 -13 -12 -11 -10 -9 -8 -7 -6 -5 -4 -3 -2 -1
* a * b b * c c c * d d d d *
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14
s[:5]
-15 -14 -13 -12 -11 -10 -9 -8 -7 -6 -5 -4 -3 -2 -1
* a * b b * c c c * d d d d *
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14
S[:-1]
-15 -14 -13 -12 -11 -10 -9 -8 -7 -6 -5 -4 -3 -2 -1
* a * b b * c c c * d d d d *
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14
S[-3:]
-15 -14 -13 -12 -11 -10 -9 -8 -7 -6 -5 -4 -3 -2 -1
* a * b b * c c c * d d d d *
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14


s[3:12:3]
-15 -14 -13 -12 -11 -10 -9 -8 -7 -6 -5 -4 -3 -2 -1
A 1 - B 2 - C 3 - D 4 - E 5 -
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14
s[:9:2]
-15 -14 -13 -12 -11 -10 -9 -8 -7 -6 -5 -4 -3 -2 -1
A 1 - B 2 - C 3 - D 4 - E 5 -
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14
s[3::4]
-15 -14 -13 -12 -11 -10 -9 -8 -7 -6 -5 -4 -3 -2 -1
A 1 - B 2 - C 3 - D 4 - E 5 -
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14
s[::6]
-15 -14 -13 -12 -11 -10 -9 -8 -7 -6 -5 -4 -3 -2 -1
A 1 - B 2 - C 3 - D 4 - E 5 -
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14


s[::2]
-15 -14 -13 -12 -11 -10 -9 -8 -7 -6 -5 -4 -3 -2 -1
A B C D E F G H I J K L M N O
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14
s[::-2]
-15 -14 -13 -12 -11 -10 -9 -8 -7 -6 -5 -4 -3 -2 -1
A B C D E F G H I J K L M N O
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14
s[::4]
-15 -14 -13 -12 -11 -10 -9 -8 -7 -6 -5 -4 -3 -2 -1
A B C D E F G H I J K L M N O
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14
s[::-4]
-15 -14 -13 -12 -11 -10 -9 -8 -7 -6 -5 -4 -3 -2 -1
A B C D E F G H I J K L M N O
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14

Пример получения года из даты

- Есть дата в таком формате: 12-10-2022.
- Как извлечь из этой строки подстроку, в которую входит только год?

Example 23, 24

Методы работы со строками

- Метод – это встроенная функция, которая работает с объектом определенного типа данных. Существуют строковые методы, методы для работы с целыми числами, методы списков, словарей и так далее.
- Любой доступный метод можно вызвать следующим образом:

объект.имя_метода(аргументы)

```
msg = "Hello World!"

msg = msg.upper()
```

Основные методы для работы со строками

- `capitalize()` Переводит первый символ строки в верхний регистр, а все остальные в нижний
- `lower()` Преобразование строки к нижнему регистру
- `swapcase()` Переводит символы нижнего регистра в верхний, а верхнего – в нижний
- `title()` Первую букву каждого слова переводит в верхний регистр, а все остальные в нижний
- `upper()` Преобразование строки к верхнему регистру

Example 25

Основные методы для работы со строками

- **`find(sub[, start[, end]])`** Возвращает индекс первого вхождения подстроки sub
- `rfind(sub)` Возвращает индекс последнего вхождения подстроки sub
- `replace(sub, new)` Заменяет подстроку sub на новую подстроку new
- **`count(sub)`** Возвращает число вхождений подстроки sub
- `startswith(prefix)` Возвращает True, если строка начинается на указанный префикс
- `endswith(sufix)` Возвращает True, если строка заканчивается на указанный суффикс.

Example 26-27

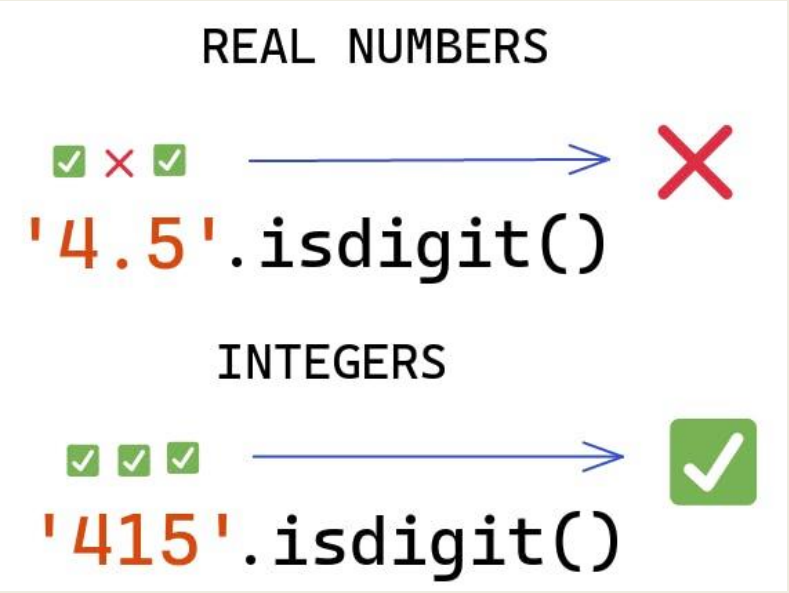
Обрезка и выравнивание

- `center(width, [fill])` Возвращает отцентрованную строку
- `ljust(width, [fill])` Делает длину строки не меньше width, по необходимости заполняя последние символы символом fill.
- `rjust(width, [fill])` Также есть метод rjust, выравнивающий по правому краю
- `zfill(width)` Делает длину строки не меньше width, по необходимости заполняя первые символы нулями
- **`strip([chars])`** Удаление пробельных символов в начале и в конце строки, либо символов из параметра chars, если он передан. Также есть команды lstrip и rstrip, удаляющие символы только слева или только справа

Example 28

Проверка типа строки

- `islower()` Состоит ли строка из символов в нижнем регистре
- `isupper()` Состоит ли строка из символов в верхнем регистре
- **`isalnum()`** Состоит ли строка из букв и цифр
- **`isalpha()`** Состоит ли строка из букв
- `isdigit()` Состоит ли строка из цифр



Example 29

Нарезка и склейка строк

- **`split()`** Разрезать строку (по пробельным символам)
- **`'[char]'.join()`** Склеивает все строки из переданного параметра, используя соединитель (возможно, пустой)
- `splitlines()` Нарезать большой кусок текста на строки
- `partition()` Ищет шаблон в строке. И разрезает строку на три части: строки до шаблона, самого шаблона, и строки после
- `maketrans(mask, shifr)` Создает таблицу замены для translate.
- `translate()` Используя maketrans таблицу, произвести посимвольную замену

Example 30
Example 31

Задача 7

Задача 7. Вы тайный агент. Вам пришла шифровка. Необходимо расшифровать.

Известно, что необходимо заменить в тексте все цифры 1 на слово 'раз', развернуть последовательность, заключенную между первым и последним вхождением буквы 'р' в обратном порядке.

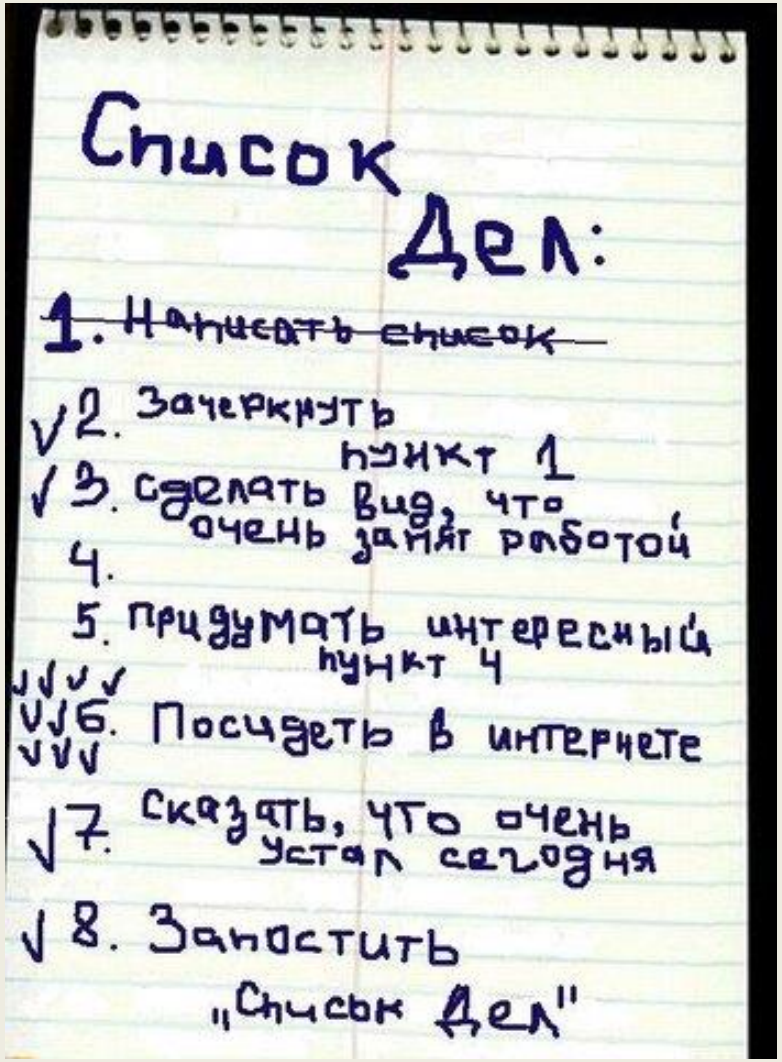
Составляем список

- Чтобы Python понял, что имеет дело со списком, нужно заключить все элементы в квадратные скобки ([]). Сами элементы разделяются запятыми.
- Мы можем создать пустой список

```
tasks = []
```

- Записать в него первую задачу

```
tasks = ['написать список’]
tasks[0] = 'зачеркнуть пункт 1'
```



Списки

- Список в Python может содержать элементы любого типа. Он вполне может содержать в себе другие списки в качестве элементов. Это называется вложенностью, а подобные списки – вложенными. Пусть сейчас наш список должен включать список на сегодня и на завтра.

```
tasks = [['посмотреть фильм', 'сдать зачет'],
         ['навестить маму', 'купить книгу']]
```

- Тогда каждый элемент будет иметь два индекса: номер во внешнем и номер во внутреннем списке.

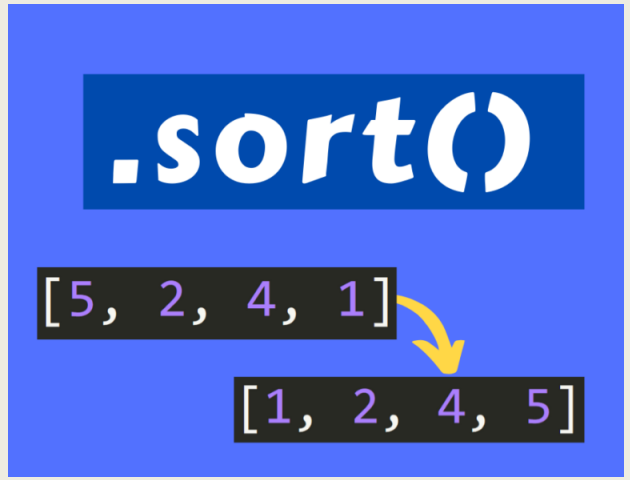
```
print(tasks[0][0])
```

```
посмотреть фильм
```

Example 33

Встроенные функции

- `min()` возвращает минимальный элемент.
- `max()` возвращает максимальный элемент.



- `sorted()` упорядочивает элементы списка (если указать sorted(s, key = s.count, reverse=True), то будет создан новый список, отсортированный в обратном порядке.

Example 33_1, 33_2

Встроенные функции

- Оператор «+» создает новый список, комбинируя списки, указанные в качестве операндов. А методы append() и extend() изменяют список.

- `pop()` возвращает последний элемент списка и удаляет его.

```
last_element = tasks.pop()
not_needed = tasks[1].pop(0)
```

- Если нам не нужен доступ к значению удаляемого элемента, можно воспользоваться функцией del.

```
del tasks[0]
tasks[0].remove('сдать зачет')
```

Example 33_3

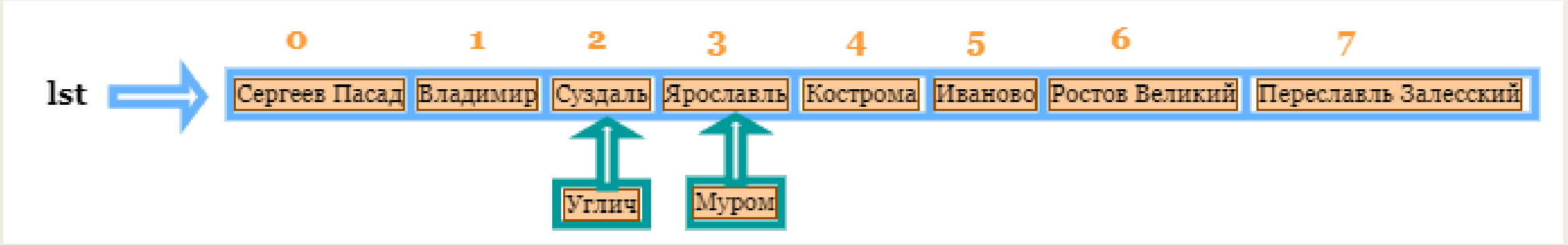
Встроенные функции

- `append()` Добавляет элемент в конец списка
- `insert()` Вставляет элемент в указанное место списка
- `remove()` Удаляет элемент по значению
- `pop()` Удаляет последний элемент, либо элемент с указанным индексом
- `clear()` Очищает список (удаляет все элементы)
- `copy()` Возвращает копию списка
- `count()` Возвращает число элементов с указанным значением
- `index()` Возвращает индекс первого найденного элемента
- `reverse()` Меняет порядок следования элементов на обратный
- `sort()` Сортирует элементы списка

Срезы для списков

```
lst = ['Сергиев Посад', 'Владимир', 'Суздаль', 'Ярославль', 'Кострома', 'Иваново', 'Ростов Великий', 'Переславль-Залесский']
```


```
lst[1:3]
```



Example 33_4

Словари

- Словарь – это неупорядоченная последовательность произвольных объектов с доступом по ключу.
- Пусть необходимо связать каждый город из маршрута с телефонным кодом. Например: Сергиев Посад – 49654, Владимир – 4922…
- dict1 = {'Сергиев Посад': 49654, 'Владимир': 4922, 'Углич': 48532, 'Муром': 49234, 'Кострома': 4942, 'Иваново': 4932, 'Ростов Великий': 48536, 'Переславль-Залесский': 48535}
  Можно создать словарь при помощи слияния двух списков, причем лишние значения будут проигнорированы.

Example 34, 35

Методы работы со словарем

- `clear()` Удаляет все элементы из словаря
- `copy()` Возвращает копию словаря
- `get(key [,d])` Возвращает значение ключа key. Если key не существует, возвращает d (по умолчанию None)
- `items()` Возвращает новый объект элементов словаря в формате (ключ, значение)
- `keys()` Возвращает новый объект с ключами словаря
- `pop(key [,d])` Удаляет элемент с ключом key и возвращает его значение или d, если key не найден. Если d не было обозначено и key не найден, вызывает ошибку KeyError

Методы работы со словарем

- `popitem()` Удаляет и возвращает последнюю пару (ключ, значение). Вызывает ошибку KeyError, если словарь пустой
- `setdefault(key [,d])` Если ключ key есть в словаре, возвращает соответствующее ему значение. Если нет, добавляет в словарь элемент с ключом key и значением d и возвращает d (по умолчанию None)
- `update([other])` Обновляет словарь имеющимися парами ключ-значение из другого словаря, перезаписывая существующие ключи
- `values()` Возвращает новый объект со значениями словаря

Example 36

Забегая вперед

- Очень часто необходимо перебирать значения, хранящиеся в словарях. Для этого используется следующая конструкция.

```
for <переменная-ключ> in <словарь>:

<действие>
```

Example 37

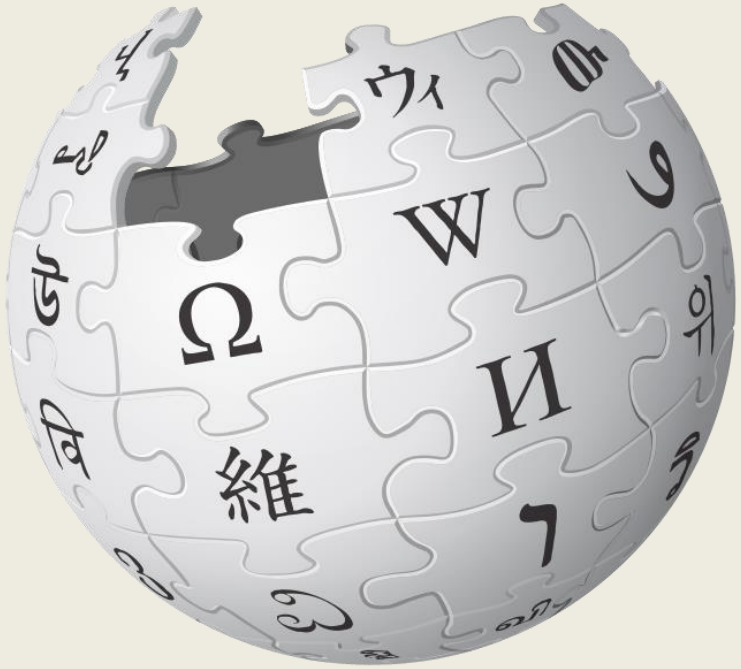
Полезные модули и библиотеки

- В составе Python есть множество полезных модулей и библиотек, функции из которых можно использовать в своей программе.
- Мы уже говорили о модуле math, содержащем математические методы. Сейчас нам может потребоваться функция генерации случайных значений.
- Набор таких функций реализован в модуле random. Например, если выполнить
- import random, то можно использовать функцию randint(start, end), генерирующую целое число в заданном диапазоне, или функцию randrange(n), генерирующую случайное целое число со значением от 0 до n - 1.

Задача 8

- *Задача 8*. Составим свою мини-Википедию. Представим, что у нас есть список знаменитостей, о каждом из которых имеется информация, например, год и страна рождения, страна проживания, сфера деятельности, самое большое достижение и так далее, причем не у всех набор информации одинаков.

- Напишите программу, которая по запросу пользователя из известного списка будет выводить имеющуюся о человеке информацию.



Задача 9

- *Задача 9*. Вам поручили написать часть модуля RPG игры с простой системой прокачки персонажа. На старте персонаж обладает минимальным списком сопротивлений. По ходу игры ему случайным образом выпадают заклинания, позволяющие пополнить их список.
- Что использовать?
  - список магических заклинаний ограничен (**кортеж**).
  - заклинание связно с получением нового резиста (**словарь**).
  - то что имеет и получает персонаж в одном месте (**список**).

Практическая работа

1) Напишите программу, которая принимает на вход строку, преобразует ее в список и определяет, сколько слов в списке содержат заданную пользователем букву, причем только один раз.
2) Разработайте программу, которая использует информацию о ряде государств в Европе и Азии: название, столица, часть света, численность населения, площадь территории. Программа должна определять количество государств, расположенных в заданной части света, общую площадь государств в заданной части света, государство с минимальной численностью населения.

Контрольные вопросы

1) Что определяет тип переменной?
2) Python является языком с неявной сильной динамической типизацией. Что это означает?
3) Перечислите встроенные типы данных в Python.
4) Что такое множество? Какие операции существуют над множествами в Python?
5) Что такое кортеж? Какие операции над кортежами существуют в Python?
6) Что такое словарь? Какие операции над словарями существуют в Python?
7) Что такое список в Python? Опишите процесс создания списка.
8) Что такое изменяемые и неизменяемые объекты?
9) Какие существуют операции над строками в языке Python?
10) Определите приоритет операций и найдите значение выражения

- 7 * 3**2 + 4 / 2 - (8 + 2 *4)/2