**Основы Python**

**Команды для Jupyter**

Выйти из режима редактирования ячейки: Esc

Создать ячейку ниже: b

Создать ячейку выше: a

Удалить ячейку: dd

Вырезать ячейки: x

Копировать ячейки: c

Вставить копированные ячейки: v

Изменить тип ячейки c "code" на "markdown" : m

Выделить несколько ячеек подряд: Shift + вверх или Shift + вниз

Объединить выделенные ячейки: Shift + m

**Выполнить ячейки:**

Ctrl + Enter — выполнение ячейки с сохранением фокуса

Shift + Enter — выполнение ячейки с переводом фокуса на следующую

простой комментарий

'''

я

многострочный

комментарий

'''

**Переменные**

> переменные могут менять значения

a = 10

b = 'peace'

c = 3.1415

print(f'{a}, {b}, {c}')

b = 'hello' изменили значение b

print(f'{a}, {b}, {c}')

**множественное присваивание значений переменных**

var, strings, pi = 20, 'hi', 3.1415

print(f'{var}, {strings}, {pi} ')

**Базовые операции**

print(28 + 18)

print(118 - 256)

print(25 / 5)

print(f'Деление нацело 9 // 4 = {9 // 4}')

print(f'Остаток от деления 9 % 4 = {9 % 4}')

print(f'Возведение в степень 3 \*\* 4 = {3 \*\* 4}')

`Задача`

Посчитать доход через 5 лет в 2 банках по вкладу, исходная сумма = 100000 рублей

initial\_sum = 100000 исх сумма

interest\_rate\_1 = 2 % годовых

interest\_rate\_2 = 3 % годовых

years = 5 длительность вклада

end\_sum\_1 = initial\_sum \* ((100 + interest\_rate\_1) / 100)\*\*years

end\_sum\_2 = initial\_sum \* ((100 + interest\_rate\_2) / 100)\*\*years

print(f'Сумма после 5 лет:\n - банк 1: {round(end\_sum\_1)} \n - банк 2: {round(end\_sum\_2)}')

**сокращенные записи выражений**

a = 10

a \*= 5 a = a \* 5

print(a)

a /= 2

print(a)

a += 100

print(a)

a -= 25

print(f'{a}')

**Типы данных**

для определения типа переменной можно использовать `type()`

**> Неизменяемые типы данных:**

Неизменяемые типы данных:

Строки (англ. "string", обозначаются как str)

my\_string = 'Hello'

type(my\_string)

**Целые числа (англ. "integer", обозначаются как int)**

years = 36

type(years)

**Числа с плавающей точкой (англ. "float", обозначаются как float)**

pi = 3.1415

type(pi)

**Кортежи (англ. "tuple", обозначаются как tuple)**

numbers\_1 = (1, 2, 3, 4, 5)

type(numbers\_1)

**Логический тип данных (англ. "boolean", обозначаются как bool)**

a = 28 > 15

type(a)

**Списки (англ. "list", обозначаются как list)**

my\_list = ['lucky', 'sirius86', 288]

type(my\_list)

**Словари (англ. "dictionary", обозначаются как dict)**

my\_dict = {

'lists': 'mutable',

'Strings': 'immutable',

'Numbers': 'immutable'

}

type(my\_dict)

**Логические операторы**

**оператор and**

Оператор and будет возвращать True, если в выражении И условие слева И условие справа истинно.

5 == 5 and 25 < 26

**оператор or**

Этот оператор будет возвращать True, если хотя бы одно из условий слева или справа истинное,и возвращать False, если оба условия ложные

print(type(3.14) == int or 28 == 28)

print(34 > 100 or 2 \*\* 3 == 9)

**Оператор not**

Этот унарный (т. е. применяется только к одному выражению,

стоящему справа от него) оператор изменяет логическое значение на противоположное, инвертирует его.

Например:

print(False or not False)

not True

**Приоритетность операторов**

1. Скобки ()

2. not

3. and

4. or

Операторы принадлежности

> оператор in

возвращает логическое значение True или False в зависимости от того присутствует ли значение в последовательности

animals = ['cat', 'dog', 'bird']

print('bird' in animals)

print('text1' in animals)

**> Оператор not in**

animals = ['cat', 'dog', 'bird']

'Text' not in animals

**> Коллекции: Строки**

Строка (string) — это неизменяемый упорядоченный тип данных, представляющий собой последовательность символов, заключенную в кавычки "".

Кавычки могут быть как одинарными, так и двойными.

language = 'Python'

name = 'Monti '

print(language)

print(name + language) сложение строк

умножение строк

target = 'learn '

print(target \* 5)

In[26]:

name = 'Neo'

reality = 'Matrix'

print(f'Wake up, {name}. The {reality} has you')

**> Коллекции: Списки**

\*\*Список (list)\*\* — это упорядоченная и изменяемая последовательность, позволяющая хранить внутри себя значения различных типов. Для создания списка в Python необходимо перечислить значения в квадратных скобках [].

\*Запоминает порядок, в котором элементы были добавлены.\*

my\_list = [1, 2, 3, 4, 5, 'Start'] объявление и заполнение списка

new\_list = [] пустой список

print(type(new\_list))

rint(f'Длина списка my\_list: {len(my\_list)}')

new\_list = [3.14, 28, 'Python'] элементы списка могут быть разного типа

print(\*new\_list)

In[35]:

сложение списков

my\_list + new\_list

> добавление элемента в конец списка:

`.append()`

In[38]:

my\_list.append('Java Script') добавление проходит при каждом запуске ячейки

In[39]:

my\_list

Удаление элементов списка

In[40]:

print(my\_list.pop()) удаление последнего элемента

In[41]:

print(my\_list.pop(1)) удаление второго элемента

In[42]:

print(my\_list.remove(5)) удаление элемента по его названию

my\_list

In[43]:

my\_list[3] вывести 4 элемент списка

> Кортежи (tuple)

Кортеж (tuple) — это упорядоченная неизменяемая коллекция. Кортеж обозначается в Python круглыми скобками ().

In[33]:

my\_typle = (1, 2, 3) создание кортежа

my\_typle

In[34]:

my\_typle\_one = ('Test', ) создание кортежа с 1 элементом

type(my\_typle\_one)

In[35]:

сложение кортежа, по факту создается новый объект

(25, 'Action') + my\_typle + my\_typle\_one

\*\*!!! Кортежи применяют когда требуются удобства списков, но в то же время важна неизменяемость данных.\*\*

Set (Множества)

`Множество (set)` — неупорядоченная и не индексируемая коллекция. Она особенна тем, что хранит только уникальные элементы. Множества обозначаются фигурными скобками `{}`

In[1]:

a = {2, 'a', 3.14} множество

print(f'{a}, {type(a)}')

In[4]:

b = set() объявить пустое множество.

print(f'{b}, {type(b)}')

In[6]:

a.add(9.88) добавим элемент в множество

print(a)

повторно добавить не получиться - в множестве только уникальные элементы

In[7]:

a.remove(9.88) удалить определенный элемент из множества

print(a)

при повторном удалении будет ошибка !!!!

In[9]:

Часто множества используются для удаления дублей.

color = ['blue', 'red', 'orange', 'red', 'blue', 'green', 'gray']

set(color)

> В множество можно добавлять только неизменяемые объекты! Это связано с его внутренним строением. Так, создать множество вида {2, "hello", []} не получится, т. к. третий элемент имеет тип list, а он изменяемый.

> Объекты типа set близки к "множествам" из математики. Можно брать пересечения множеств, объединение, симметричную разность:

- \*\*a.intersection(b)\*\* — возвращает множество, являющееся пересечением a и b

- \*\*a.union(b)\*\* — объединяет множества a и b

- \*\*a.symmetric\_difference(b)\*\* — возвращает множество элементов, которые не пересекаются в a и b

In[11]:

a = {'a', (1, 2), 3}

b = {3, (1, 2), 'unique'}

a.intersection(b)

In[12]:

a.union(b)

In[13]:

a.symmetric\_difference(b)

Коллекции: Словари

\*\*Словарь (dictionary)\*\* — ассоциативный тип данных, где каждый элемент является парой ключ-значение. Для создания словаря необходимо указать элементы внутри фигурных скобок {}.

In[18]:

объявляем словарь, пара ключ - значение

name\_to\_number = {

'Anton' : 9884561278,

'Maksim': 9567892545,

'Nikita': 9284736581

}

print(f'{type(name\_to\_number)}, {len(name\_to\_number)}') тип - словарь, длина 3 элемента

name\_to\_number

In[23]:

получить значение по ключу

name\_to\_number['Maksim']

In[24]:

метод выводит сообщение, если элемент словаря не найден

name\_to\_number.get('Aleksey', 'no info')

In[25]:

name\_to\_number['Mariya'] = 9037893479

In[28]:

проверка, есть ли ключ в словаре

print('Mariya' in name\_to\_number)

name\_to\_number

In[29]:

можно редактировать значение по ключу

name\_to\_number['Anton'] = 9684561278

name\_to\_number