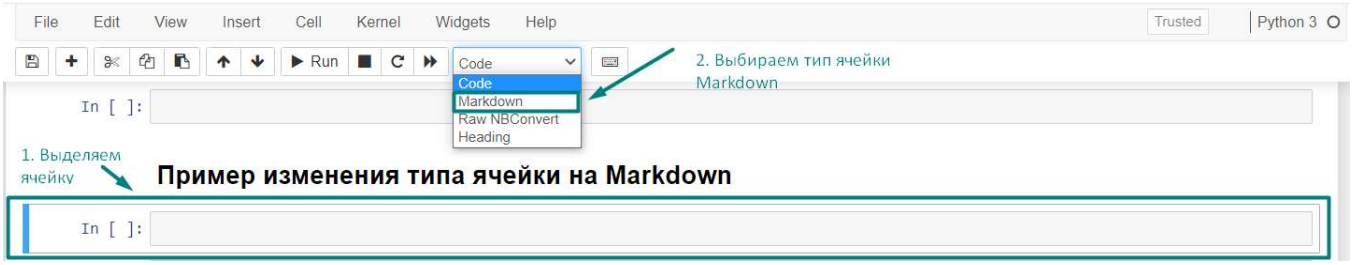
День №1

Введение в Python

Мы находимся в Jupyter ноутбуке. Ячейки кода выделены серым цветом. В них можно писать код и выполнять его пошагово. Эта особенность сделала jupyter ноутбуки основным инструментов для анализа данных.
Помимо ячеек с кодом, jupyter notebook позволяет записывать любой текст с помощью облеченного языка разметки **"Markdown"**. Для этого нужно выделить ячейку с кодом и с помощью панели меню выбрать тип ячейки "Markdown".



Кроме этого, ячейки markdown позволяют использовать HTML (язык гипертекстовой разметки) для стилизации ячейки.

Подробнее об особенностях markdown читайте в документации (https://docs.github.com/en/get-started/writing-on-github/getting-started-with-writing-and-formatting-on-github/basic-writing-and-formatting-syntax)

Пишем первую программу на Python

Когда программисты начинают изучать новый язык, мы всегда пишем простую программу из одной команды, которая выводит сообщение "Hello, world!". Это простая программа, которая показывает, что все необходимые базовые составляющие языка программирования установились корректно и загрузились успешно.

Напишете в ячейке ниже команду **print('Hello, world!')** и выполните ячейку

Ввод [2]:

```
1 print('Hello, world!')
```

```
Hello, world!
```

Переменные

Переменные – объекты, в которых хранится какая-либо информация (числа, текст и т.д). Они нужны для того, чтобы хранить различные данные во время выполнения кода

2. Правила работы с переменными

- Названия переменных может содержать только буквы, цифры или нижнее подчеркивание. Название может начинаться с буквы или нижнего подчеркивания, но не с цифры

- Использовать пробелы в названии нельзя (пробелы можно заменить нижним подчеркиванием)
- Нельзя использовать зарезервированные ключевые слова языка Python (например, "and")
- Не рекомендуется использовать длинные названия переменных

Ввод [7]:

```
# Создаем переменные name с различными названиями, присвоив им различные значения
var_1 = 5
print(var_1)

_var_2 = 10
print(_var_2)

asfaljshdlsahdljhafljahflksahflkhsaf = 100
print(asfaljshdlsahdljhafljahflksahflkhsaf)

# 10_var = 10
# print(10_var)

```

```
5
10
100
```

3. Ошибки при работе с переменными

Существует типовая ошибка в использовании переменных, с которой вы рано или поздно столкнетесь. Посмотрите на код ниже и скажите, почему он вызывает ошибку?

Ввод [11]:

```
# Этот код с ошибкой
message = 'Учение - свет!'
print(message)
```

```
Учение - свет!
```

Давайте посмотрим на вид ошибки - **NameError**. Зеленая пунктирная линии показывает нам, в какой части нашего кода произошла ошибка. В описании ошибки мы видим текст "name 'message' is not defined", что означает, что название "message" не определено.

Почему происходит эта ошибка?

4. Простейшая арифметика

Ячейки jupyter ноутбука поддерживают выполнение простейших арифметических операций. При этом даже необязательно объявлять переменные. Ниже вы найдете примеры таких операций.

Ввод [12]:

```
# Сложение
4 + 10
```

Out[12]:

```
14
```

Ввод [13]:

```
# Возведение в степень (двойная звездочка, а не ^, как в Excel)
10**5
```

Out[13]:

```
100000
```

Ввод [14]:

```
# Деление
100/10
```

Out[14]:

```
10.0
```

Ввод [15]:

```
# Деление №2
9//4
```

Out[15]:

```
2
```

Ввод [16]:

```
# Деление №3
9%4
```

Out[16]:

```
1
```

5. Типы данных

Вы наверняка заметили, что мы присваивали переменным 2 вида значений: текст и целые числа. И текст, и целые числа являются базовыми основными типами данных в Python. Базовых типов данных в Python целых 4:

1. Строка (string), сокращенно str
2. Целые числа (integer), сокращенно int
3. Дробные числа (float), сокращенно float
4. Булевые значения (boolean), сокращенно bool

Создадим 4 переменные, в которых отразим каждый из перечисленных выше типов переменных, и с помощью функции **type()** напечатаем типы созданных переменных

Ввод [20]:

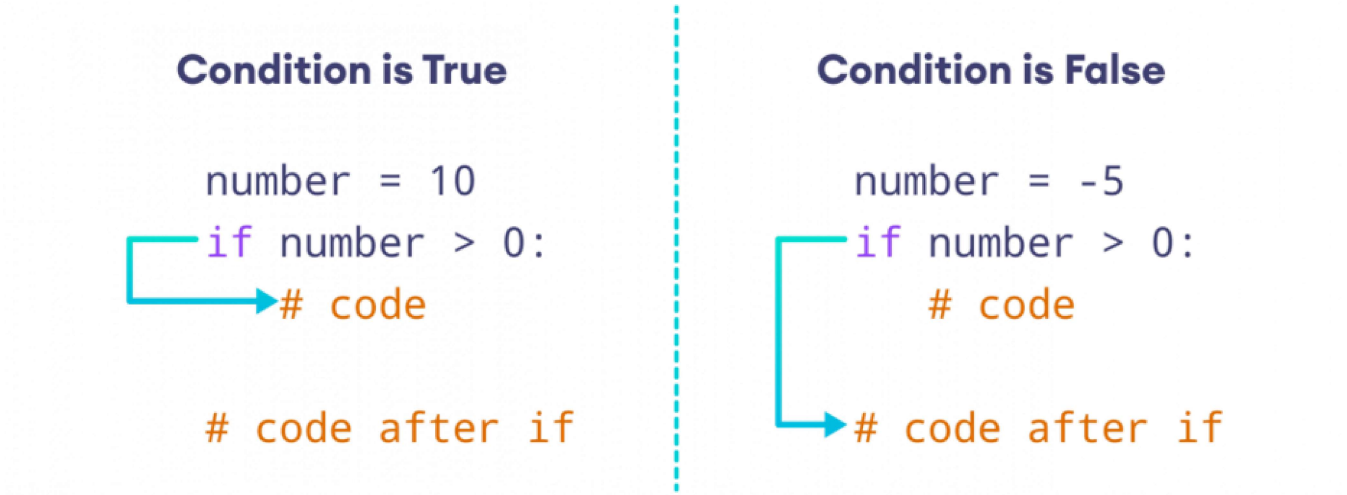
```
# Создаем переменные var, присваиваем им значения разных типов данных и выводим на п
var = 'Text'
print(type(var))

var_2 = 10
print(type(var_2))

var_3 = 8.2
print(type(var_3))

var_4 = True
print(type(var_4))
```

```
<class 'str'>
<class 'int'>
<class 'float'>
<class 'bool'>
```

6. Условия (if...else)

Условия в любом языке программирование один из основных инструментов принятия решений. В python в основе условий лежит инструкция if-elif-else (её ещё иногда называют оператором ветвления). Проще говоря, она выбирает, какое действие следует выполнить, в зависимости от значения переменных в момент проверки условия.

До настоящего момента мы писали с вами код без каких-либо отступов. Конструкция условий в python - первая, в которой применяются отступы.



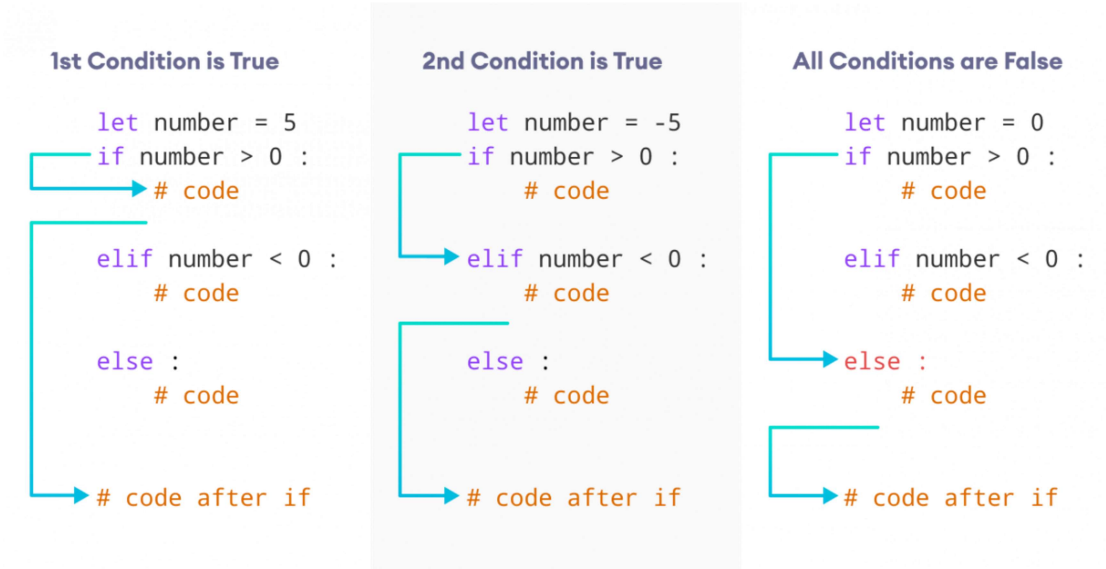
При создании условия можно применять логические проверки и операторы сравнения.

Ввод [25]:

```
# Создадим переменную x, присвоим ей значение 10 и создадим простое условие,
# которое будет нам печать значение переменной,
# если оно больше 10

x = 11
print('var', x)
print(x > 10)
if x > 10:
    print(x)
```

```
var 11
True
11
```

Операторы if…else позволяют задать 2 условия. Если нам необходимо использовать доп. условие, конструкция принимает вид **if…elif…else**.



Ввод [29]:

```
# Повторим предыдущий код, добавив в него дополнительное условие elif
y = 5

if y > 5:
    print('y > 5')
elif y == 5:
    print('y = 5')
else:
    print('y < 5')
```

```
y = 5
```

Условия могут быть вложенными. Представьте, что пользователь назвал вам 2 числа. Если первое число будет больше 10, то нам необходимо перейти к проверке второго числа. Если второе число будет больше 1, то возведем его в квадрат и выведем на экран. Если меньше - выведем сообщение, что первое число меньше 10.

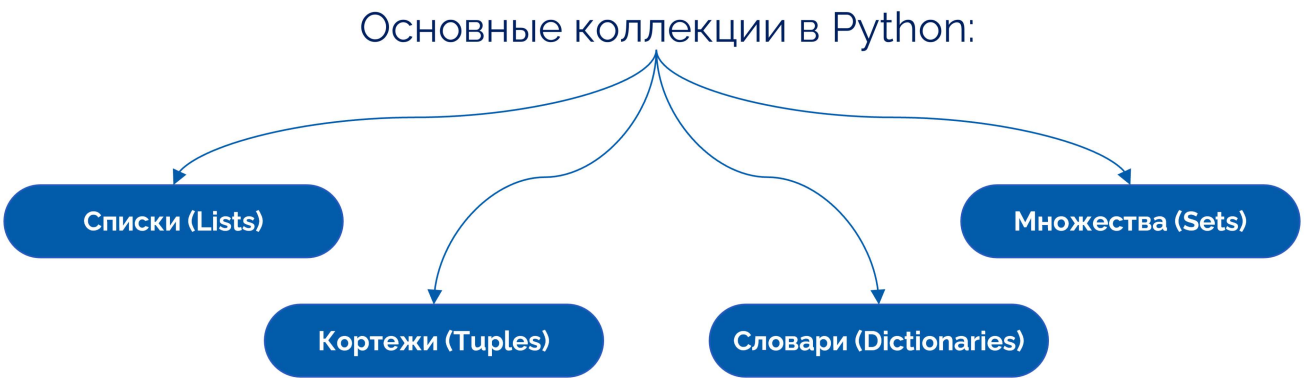
Ввод [32]:

```
number_1 = 11
number_2 = 15

if number_1 > 10:
    if number_2 > 1:
        print(number_2**2)
    else:
        print('number_1 < 10')
```

```
225
```

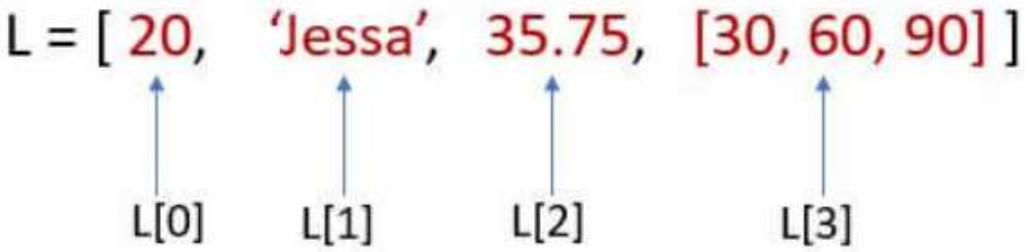
7. Контейнеры (коллекции) в Python

Контейнеры (коллекции) в Python – особые структуры данных, которые позволяют хранить в себе информацию в различном виде.



8. Списки (Lists)

В Python список является типом данных (коллекцией), который позволяет хранить в нем объекты различных типов: числа, строки, другие списки и т.д. Фактически список - это контейдер для объектов различных типов. Списки являются одним из самых популярных типом данных в Python.



- ✓ **Ordered**: Maintain the order of the data insertion.
- ✓ **Changeable**: List is mutable and we can modify items.
- ✓ **Heterogeneous**: List can contain data of different types
- ✓ **Contains duplicate**: Allows duplicates data

Создается список с помощью квадратных скобок **[ ]** или функции **list( )**. Создадим пустой список и проверим, какой тип данных ему присвоен

Ввод [33]:

```
# Создадим пустой список и посмотрим, какой тип данных присвоил ему Python
list_1 = []
print(list_1)
print(type(list_1))
```

```
[]
<class 'list'>
```

Ввод [36]:

```
# Создадим список с различными числовыми значениями и
# проверим длину этого списка с помощью функции len()
my_list = [0, 2, 10, 20, 20, 25]
print(my_list)
print(len(my_list))

print(my_list[0])
```

```
[0, 2, 10, 20, 20, 25]
6
0
```

Ввод [76]:

```
# Создадим пустой список, а затем последовательно добавим в него значения от 1 до 5
my_list = [0, 2, 10, 20, 20, 25]
my_list.append(200)
print(my_list)
```

```
[0, 2, 10, 20, 20, 25, 200]
```

Список является упорядоченным объектом и последовательностью, поэтому мы можем обращаться к элементам списка по индексам, а также применять срезы.

Ввод [85]:

```
# Создадим лист my_list со значениями от 0 до 10 и попробуй применить несколько срез
# 1. Выведем первый элемент
print(my_list[0])
# 2. Выведем второй элемент
print(my_list[1])
# 3. Выведем первые 2 элемента
print(my_list[0:3])
# 4. Выведем элементы с 4 по 6
# 5. Выведем последний элемент
print(my_list[-1])
# 6. Выведем элементы с шагом 3
my_list[::3]
# 7. Выведем элемент, который выходит за границы списка
my_list[100]
```

```
0
2
[0, 2, 10]
200

---------------------------------------------------------------------------
-
IndexError                                Traceback (most recent call las
t)
<ipython-input-85-bad7efb9107b> in <module>
     12 my_list[::3]
     13 # 7. Выведем элемент, который выходит за границы списка
---> 14 my_list[100]

IndexError: list index out of range
```

Ввод [16]:

```
# Попробуем заменить один из элементов созданного списка на другое значение
pass
```

Для управления элементами списки имеют целый ряд методов. Некоторые из них:

- **append(item):** добавляет элемент item в конец списка
- **insert(index, item):** добавляет элемент item в список по индексу index
- **extend(items):** добавляет набор элементов items в конец списка
- **remove(item):** удаляет элемент item. Удаляется только первое вхождение элемента. Если элемент не найден, генерирует исключение ValueError

- **clear():** удаление всех элементов из списка
- **index(item):** возвращает индекс элемента item. Если элемент не найден, генерирует исключение ValueError
- **pop([index]):** удаляет и возвращает элемент по индексу index. Если индекс не передан, то просто удаляет последний элемент.
- **count(item):** возвращает количество вхождений элемента item в список
- **sort([key]):** сортирует элементы. По умолчанию сортирует по возрастанию. Но с помощью параметра key мы можем передать функцию сортировки.
- **reverse():** расставляет все элементы в списке в обратном порядке
- **copy():** копирует список

Кроме того, Python предоставляет ряд встроенных функций для работы со списками:

- **len(list):** возвращает длину списка
- **sorted(list):** возвращает отсортированный список
- **min(list):** возвращает наименьший элемент списка
- **max(list):** возвращает наибольший элемент списка

9. Давайте попрактикуемся!

1. Со дня рождения Жени прошло уже 121 день, 9 часов, 43 минуты и 40 секунд. Он решил узнать, на сколько секунд за этот период он постарел.

Ввод [86]:

```
## calc birthday
40+43*60+9*60**2+121*60**2*24
```

Out[86]:

```
10489420
```

2. Дан список из целых чисел произвольной длины. Необходимо выводить на печать True, если первый или последний элемент этого списка равен 4, в противном случае выводить False. Использовать циклы и условия нельзя.

Ввод [88]:

```
my_list = [0, 2, 4, 2, 3, 7, 9, 10, 100, 4]
my_list[0]==4 or my_list[-1]==4
```

Out[88]:

```
True
```

3. Выведите все элементы списка с четными индексами (то есть A[0], A[2], A[4], ...)

Ввод [90]:

```
print(my_list)
my_list[::2]
```

```
[0, 2, 4, 2, 3, 7, 9, 10, 100, 4]
```

Out[90]:

```
[0, 4, 3, 9, 100]
```

4. Анатолий - футбольный эксперт. Он знает, сколько каждая из команд одержала побед, сыграла вничью и потерпела поражений. Для составления итоговой таблицы с результатами ему нужен калькулятор

очков, которая набрала команда. Создайте такой калькулятор.

Ввод [91]:

```
wi = 10
lo = 2
eq = 5

point = wi*3 + lo * 0 + eq *1
print(point)
```

```
35
```

5. Владелец автосалона поддержанных автомобилей покупает автомобили только марки Ford. Он знает, что название автомобилей для одного года выпуска не повторяются. Он хочет создать простой справочник авто с короткими названиями. Например, модель Mustang 2018 года он планирует зашифровать как Mu-018. Напишите код, который будет принимать название авто и год выпуска и возвращать закодированное название автомобиля.

Ввод [96]:

```
name = 'Mustang'
year = 2018

code_name = name[:2] + '-' + str(year)[-3:]
print(code_name)
```

```
Mu-018
```