



## S2-S3: Python через команды бота

Повторяем основы Python простыми словами и сразу применяем их в боте 🐍

# Что у нас уже готово после S1

## Инструменты

Python, VS Code и Git уже установлены и готовы к работе

## Проект

Есть рабочий бот "Hello" с базовой настройкой

## Безопасность

Файлы .env и .gitignore настроены правильно

## GitHub

Репозиторий создан и готов для загрузки кода



Помним: файл .env с токеном никогда не отправляем в GitHub!

# Переменные: «коробки с ярлыком»

Переменная — это именованная коробка, где мы храним данные. Каждая коробка имеет ярлык (имя) и содержимое (значение).

```
name = "Алиса"  
age = 18  
is_student = True
```

💡 Попробуй сам:

- Создай переменные с твоим именем и возрастом
- Выведи их: `print(name, age)`



⊗ Осторожно: знак "=" это присваивание, а не математическое равенство!

# Типы данных: основная четверка

1

**int** — целые числа

Примеры: 5, -10, 0. Используем для подсчета, возраста, количества.

1.0

**float** — числа с точкой

Примеры: 3.14, -2.5, 0.0. Для измерений, расчетов с дробями.



**str** — строки (текст)


Примеры: "Привет", 'Python', "123". Всегда в кавычках!



**bool** — True/False

Только два значения: True или False. Для условий и проверок.

```
type(5)    # <class 'int'>
type("5")  # <class 'str'>
type(True) # <class 'bool'>
```

 `int("3.14")` выдаст ошибку — это не целое число!

# Строки: приводим в порядок

Часто пользователи пишут команды с лишними пробелами или в разном регистре. Python поможет это исправить!

```
text = " Привет, Мир "  
clean = text.strip().lower() # "привет, мир"  
  
# Еще полезные методы:  
text.upper() # "ПРИВЕТ, МИР"  
text.replace("Мир", "Python") # "Привет, Python"
```

 Попробуй сам:

- Очисти строку `"ECHO Привет ".strip().lower()`
- Что получится?



# Извлекаем команду и аргументы ⚡

Когда пользователь пишет `/echo Привет мир`, нам нужно разделить команду и текст после неё.

```
cmd = "/echo Привет мир"
parts = cmd.split(maxsplit=1)

command = parts[0] # "/echo"
arg = parts[1] if len(parts) > 1 else "" # "Привет мир"
```

01

## Разбиваем строку

`split(maxsplit=1)` делит только по первому пробелу

02

## Берем команду

Первая часть — это всегда команда

03

## Проверяем аргументы


Если есть вторая часть — это аргументы

❌ `split()` без `maxsplit` разобьет по всем пробелам — это не всегда нужно!

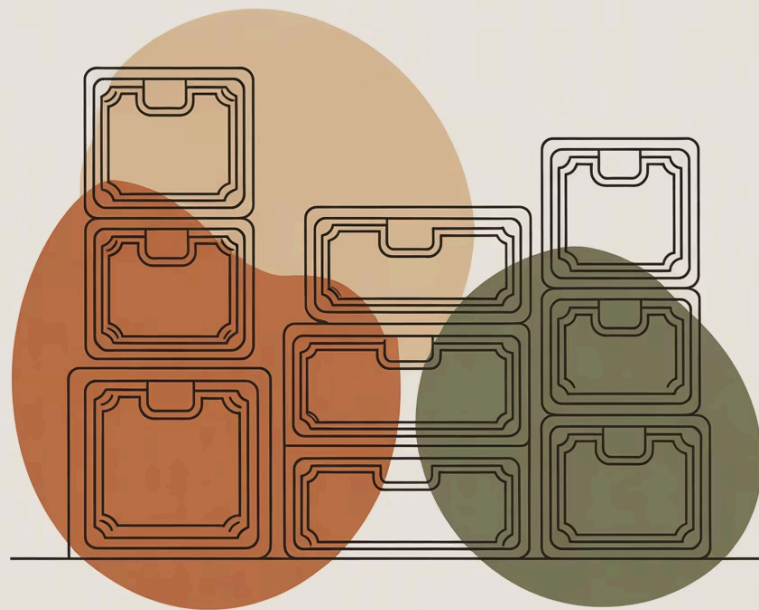
# Списки: коробка с отделениями

Список — это упорядоченная коллекция элементов. Можно хранить числа, строки, что угодно!

```
nums = [1, 2, 3]
nums.append(4)    # [1, 2, 3, 4]
nums[0]          # 1 (первый элемент)
len(nums)        # 4 (количество элементов)
```

 Попробуй сам:

- Создай список из строки: "а б в".split()
- Добавь новый элемент через append()





# Словари: подписанные ячейки



Словарь хранит пары "ключ → значение". Как телефонная книга: имя → номер.

```
user = {"name": "Алиса", "age": 18}
user["age"] = 19      # изменяем значение
user["city"] = "Москва" # добавляем новый ключ

print(user["name"])   # "Алиса"
```



**Попробуй сам:**

Создай словарь с информацией о себе: имя, возраст, город



**Осторожно:**

`user["x"]` упадет с ошибкой, если ключа нет. Сначала проверяй!



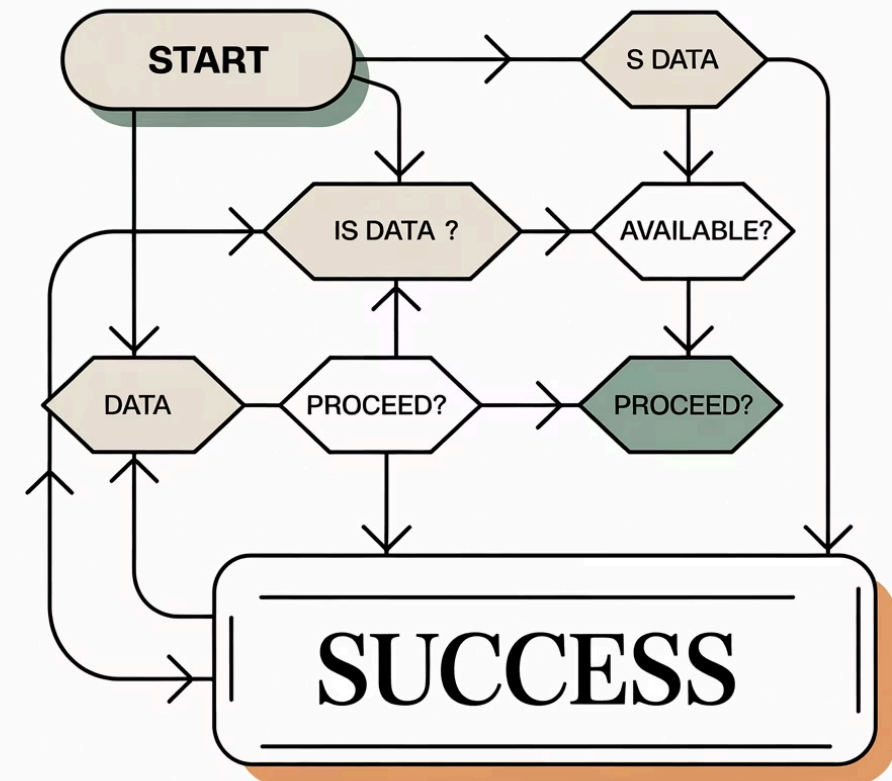
# Условия: если/иначе ?

Условия позволяют программе принимать решения в зависимости от ситуации.

```
age = 18

if age >= 18:
    print("Взрослый")
else:
    print("Не взрослый")
```

Операторы сравнения: ==, !=, <, >, <=, >=



💡 Попробуй сам:

Измени значение age на разные числа и посмотри, что выведется

# Команды бота: /about и /sum

Теперь применим все знания для создания полезных команд бота!

Команда /about — представляем бота:

```
@bot.message_handler(commands=['about'])
def about(message):
    bot.reply_to(message, "Я учебный бот: /start, /help, /sum")
```

Команда /sum — складываем числа:

```
@bot.message_handler(commands=['sum'])
def cmd_sum(message):
    parts = message.text.split()
    numbers = []

    for p in parts[1:]:
        if p.isdigit(): # только положительные целые
            numbers.append(int(p))

    if not numbers:
        bot.reply_to(message, "Напиши числа: /sum 2 3 10")
    else:
        bot.reply_to(message, f"Сумма: {sum(numbers)}")
```

1 Запусти `python main.py`

2 Проверь команды: /about, /sum 2 5 7

# Мини-квиз

Проверьте свои знания основных понятий Python, ответив на несколько быстрых вопросов!

1

Что делает метод `strip()` при работе со строками?

2

Какой тип данных используется для хранения целых чисел?

3

Как получить доступ к первому элементу списка в Python?

4

Что происходит, если вы пытаетесь получить значение по ключу, которого нет в словаре?

5

Когда выполняется блок кода после ключевого слова `else` в условной конструкции?



# МИНИ-КВИЗ

Проверьте свои знания основных понятий Python, ответив на несколько быстрых вопросов!

1

Что делает метод `strip()` при работе со строками?

Метод `strip()` удаляет начальные и конечные пробелы (или указанные символы) из строки.

2

Какой тип данных используется для хранения целых чисел?

Для хранения целых чисел в Python используется тип данных `int`.

3

Как получить доступ к первому элементу списка в Python?

Доступ к первому элементу списка осуществляется по индексу 0, например: `my_list[0]`.

4

Что происходит, если вы пытаетесь получить значение по ключу, которого нет в словаре?

При попытке получить значение по несуществующему ключу в словаре будет выдана ошибка `KeyError`.

5

Когда выполняется блок кода после ключевого слова `else` в условной конструкции?

Блок кода после `else` выполняется, если ни одно из условий `if` или `elif` не является истинным.

