3.1 Типы данных и операторы
1 из 15 шагов пройден
0 из 5 баллов получено

Типы данных и базовые операторы

Цель:

Познакомить учащихся с основными типами данных, операторами, условной логикой и циклами в Python, используя примеры из области робототехники.

Введение

Зачем программировать роботов?

Роботы — это машины, которые выполняют задачи автоматически.

Чтобы заставить робота двигаться, реагировать или принимать решения, нужен код.

Python — удобный язык для обучения и управления роботами.

На этом уроке мы узнаем:

- Какие данные используются в программах.
- Как выполнять вычисления.
- Как делать выбор (ветвление).
-

7 1

3.1 Типы данных и операторы
2 из 15 шагов пройдено
0 из 5 баллов получено

Типы данных в Python

```
int (целые числа):
```

```
age = 15
```

```
float (дробные числа):
```

```
speed = 0.5
```

```
str (строки):
```

```
name = "Робот-пылесос"
```

```
bool (логические значения):
```

```
is_moving = True
```

Пример в робототехнике:

```
distance = 10                # int
temperature = 25.7           # float
status = "в движении"        # str
obstacle_detected = False    # bool
```

Робот может **анализировать данные с датчиков** и принимать **решения на основе этих данных**.

Задача

Робот движется по местности. У него есть:

- Датчик расстояния
- Термометр
- Система обнаружения препятствий
- Статус движения

Пример датчика расстояния:

HC-SR04 — это популярный и недорогой ультразвуковой дальномер, который широко используется в электронике, особенно в проектах с микроконтроллерами, такими как **Arduino**, **Raspberry Pi** и другими. Он позволяет измерять расстояние до объекта с помощью ультразвука.

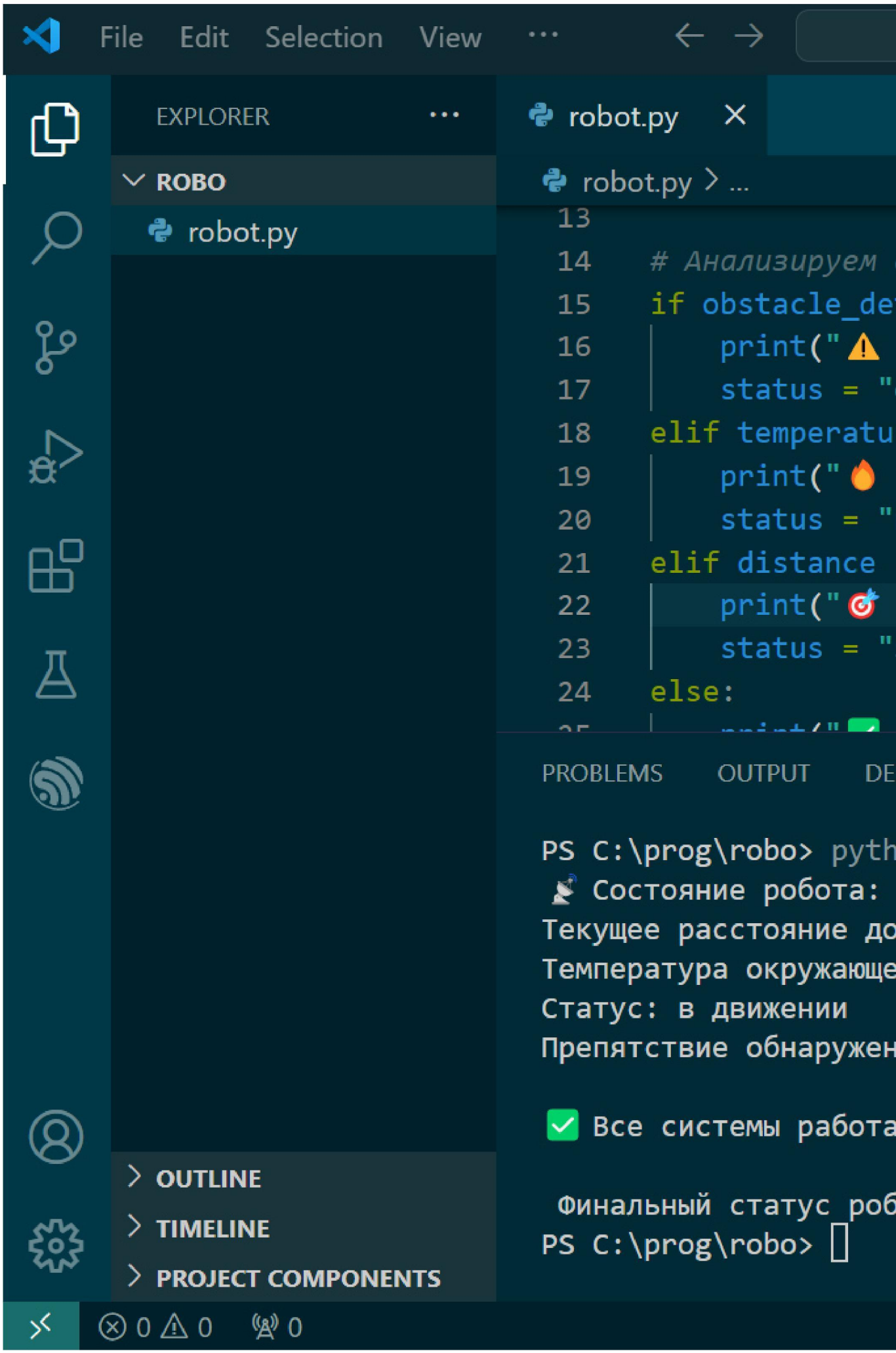
На основе полученных данных робот:

- Проверяет наличие препятствия
- Оценивает температурный режим
- Сообщает текущее состояние

```
# Данные с датчиков (можно менять для тестирования)

distance = 10            # расстояние до цели в метрах

temperature = 25.7       # температура окружающей среды

status = "в движении"    # статус робота

obstacle_detected = False  # обнаружено ли препятствие



# Выводим начальное состояние

print("📡 Состояние робота:")

print(f"Текущее расстояние до цели: {distance} м")

print(f"Температура окружающей среды: {temperature} °C")

print(f"Статус: {status}")
```

```
print(f"Препятствие обнаружено: {'Да' if obstacle_detected else 'Нет'}\n")


# Анализируем данные и принимаем решения
if obstacle_detected:
    print("⚠️ Обнаружено препятствие! Движение остановлено.")
    status = "остановлен"
elif temperature > 40:
    print("🔥 Температура слишком высока! Робот перегревается.")
    status = "неисправен"
elif distance < 1:
    print("🎯 Цель достигнута!")
    status = "задача выполнена"
else:
    print("✅ Все системы работают нормально. Продолжаю движение.")
    status = "в движении"


# Финальный статус
print(f"\n Финальный статус робота: {status}")
```

Запуск:

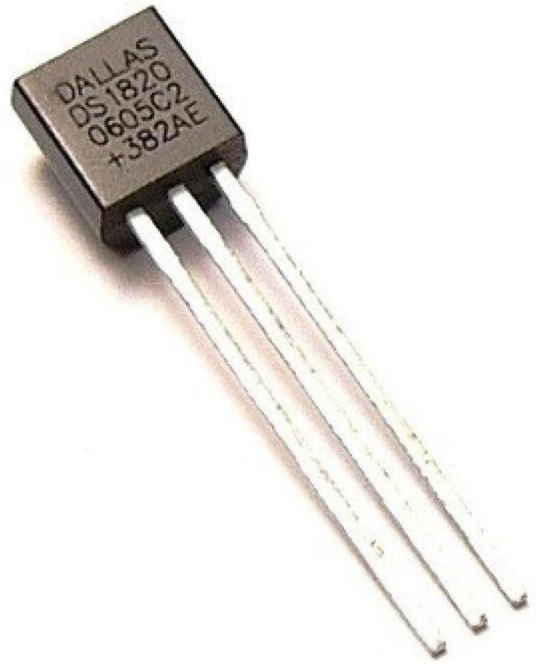

File Edit Selection View ···
EXPLORER ···
ROBO
robot.py
robot.py
robot.py > ...
13
14 # Анализируем
15 if obstacle_de
16 print("⚠️
17 status = "
18 elif temperatu
19 print("🔥
20 status = "
21 elif distance
22 print("🎯
23 status = "
24 else:
PROBLEMS OUTPUT DE
PS C:\prog\robo> pyth
📡 Состояние робота:
Текущее расстояние до
Температура окружающе
Статус: в движении
Препятствие обнаружен
✅ Все системы работа
Финальный статус роб
PS C:\prog\robo>
OUTLINE
TIMELINE
PROJECT COMPONENTS
0 0 0

Для справки, чтобы была связь с реальным миром и было понимание того, как получаются значения (расстояние, температура и т.д.), приводим пример кода на MicroPython (что это такое и зачем можно будет узнать далее в курсе), который считывает данные с ультразвукового датчика HC-SR04 и вычисляет расстояние до объекта:

```
from machine import Pin

import time


# === Настройка пинов ===

trig_pin = Pin(5, Pin.OUT)    # TRIG подключен к GPIO5

echo_pin = Pin(4, Pin.IN)     # ECHO подключен к GPIO4


def get_distance():

    # Отправляем короткий импульс на TRIG

    trig_pin.off()

    time.sleep_us(2)

    trig_pin.on()

    time.sleep_us(10)

    trig_pin.off()


    # Ждём, пока ECHO станет HIGH

    timeout = 100000  # максимальное время ожидания в микросекундах

    start_time = 0

    stop_time = 0


    while echo_pin.value() == 0:

        start_time = time.ticks_us()

        if time.ticks_diff(time.ticks_us(), start_time) > timeout:
```

```
            return None  # Таймаут

    while echo_pin.value() == 1:
        stop_time = time.ticks_us()
        if time.ticks_diff(time.ticks_us(), start_time) > timeout:
            return None  # Таймаут

    # Вычисляем длительность сигнала
    duration = time.ticks_diff(stop_time, start_time)

    # Расстояние в см: (время * скорость звука) / 2
    distance = (duration * 0.0343) / 2  # 0.0343 см/мкс — скорость звука при 20°C

    return distance

# === Основной цикл ===
while True:
    dist = get_distance()
    if dist is not None:
        print("Расстояние: {:.2f} см".format(dist))
    else:
        print("Ошибка измерения")
    time.sleep_ms(500)
```

Скорость звука зависит от температуры. При необходимости можно добавить датчик температуры (например, DS18B20), чтобы уточнить значение.



7 1

3.1 Типы данных и операторы

3 из 15 шагов пройдено

0 из 5 баллов получено

Переменные и операторы присваивания

Переменная — это ячейка памяти с именем.

Имя переменной может содержать буквы, цифры (но не в самом начале) и знаки нижнего подчёркивания.

Присваивание значения: = (знак равно).

Пример:

```
x = 5

y = x + 3
```

Пример (осмысленные названия):

```
motor_power = 70    # процент мощности мотора

sensor_value = 0    # начальное значение датчика
```

Система контроля мощности моторов робота

Цель:

Продемонстрировать, как робот может **управлять мощностью моторов**, основываясь на показаниях датчиков и ограничениях системы.

Что используется:

- Переменные с осмысленными именами
- Операторы присваивания (`=`)
- Арифметические операции (`+`, `-`, `*`, `/`)
- Условия (`if/else`)
- Вывод информации (`print()`)

Пример программы:

```
# Параметры робота

motor_power_left = 60      # текущая мощность левого мотора (%)

motor_power_right = 60     # текущая мощность правого мотора (%)

max_motor_power = 100      # максимальная допустимая мощность

min_motor_power = 0        # минимальная допустимая мощность


sensor_temperature = 45    # температура двигателя (°C)

overheat_threshold = 70    # порог перегрева


# Изменение мощности на основе команды пользователя

command = input("Введите команду (вперёд, назад, влево, вправо, стоп):
").lower()


if command == "вперёд":

    motor_power_left += 20

    motor_power_right += 20

elif command == "назад":

    motor_power_left -= 10
```

```
    motor_power_right -= 10
elif command == "влево":
    motor_power_left -= 15
    motor_power_right += 15
elif command == "вправо":
    motor_power_left += 15
    motor_power_right -= 15
elif command == "стоп":
    motor_power_left = 0
    motor_power_right = 0
else:
    print("Неизвестная команда. Моторы не изменены.")


# Ограничение мощности до допустимых значений
motor_power_left = max(min(motor_power_left, max_motor_power),
min_motor_power)
motor_power_right = max(min(motor_power_right, max_motor_power),
min_motor_power)


# Проверка на перегрев
if sensor_temperature > overheat_threshold:
    print("⚠ Перегрев! Все моторы остановлены.")
    motor_power_left = 0
    motor_power_right = 0
else:
    print(f"☑ Состояние моторов: Левый = {motor_power_left}%, Правый
= {motor_power_right}%")
```

Запуск:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\prog\robo> python robot.py
Введите команду (вперёд, назад, влево, вправо, стоп
✅ Состояние моторов: Левый = 80%, Правый = 80%
PS C:\prog\robo> python robot.py
Введите команду (вперёд, назад, влево, вправо, стоп
✅ Состояние моторов: Левый = 75%, Правый = 45%
PS C:\prog\robo> python robot.py
Введите команду (вперёд, назад, влево, вправо, стоп
✅ Состояние моторов: Левый = 0%, Правый = 0%
PS C:\prog\robo>
```

Программа имитирует **управление двумя моторами робота** (левым и правым) с учётом:

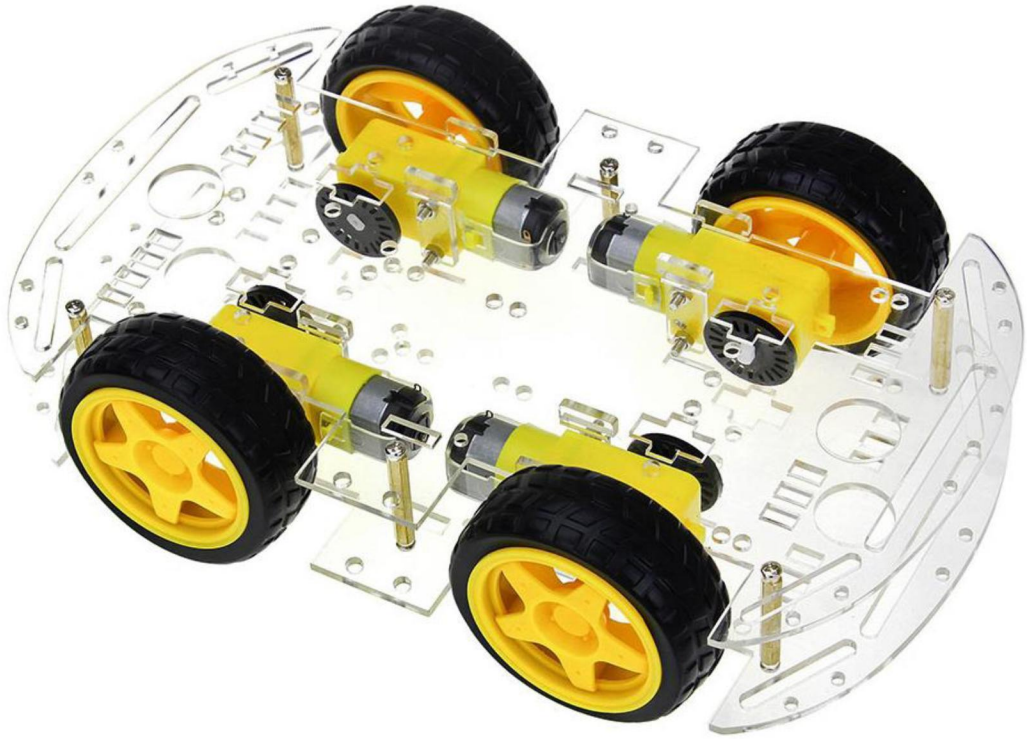
- Вводимой пользователем команды,
- Ограничений по максимальной и минимальной мощности,
- Проверки температуры двигателя на перегрев.

Пример моторов (это тип TT с пластиковым редуктором для колёсного робота):

TT-мотор — это популярный тип небольшого **двигателя постоянного тока (DC motor)**, часто используемый в робототехнике и DIY-проектах, особенно с Arduino,

ESP32, Raspberry Pi и другими контроллерами. Обычно TT-мотор используется как привод для колес в самодельных роботах.

К моторам крепятся колёса и всё вместе размещают на шасси:



---

7 1

3.1 Типы данных и операторы

4 из 15 шагов пройдено

0 из 5 баллов получено

Арифметические операторы

+ (сложение)

- (вычитание)

* (умножение)

/ (деление (дробное))

// (целочисленное деление)

% (остаток от деления)

** (возведение в степень)

Пример:

```
wheel_diameter = 10
```

```
pi = 3.14

circumference = pi * wheel_diameter  # длина окружности колеса
```

Расчёт пройденного пути робота

Цель программы:

Рассчитать, **сколько оборотов сделает колесо робота** и **какое расстояние он пройдёт**, зная:

- Диаметр колеса
- Количество оборотов
- Используя формулу длины окружности

Что используется:

- Арифметические операции (`*`, `/`, `**`)
- Переменные с осмысленными именами
- Форматированный вывод (`f-строки`)
- Ввод данных от пользователя

Пример программы:

```
# Параметры робота

wheel_diameter = 10      # диаметр колеса в см

pi = 3.14                # число π (приближённое значение)
```

```
rotations = 5            # количество оборотов колеса


# Рассчитываем длину окружности колеса

circumference = pi * wheel_diameter  # C = π * d


# Рассчитываем пройденное расстояние

distance = circumference * rotations


# Выводим результаты

print(f"Диаметр колеса: {wheel_diameter} см")

print(f"Число π: {pi}")

print(f"Количество оборотов: {rotations}")

print(f"Длина окружности колеса: {circumference:.2f} см")

print(f"Пройденное расстояние: {distance:.2f} см")
```

Запуск:

```
PS C:\prog\robo> python robot.py
Диаметр колеса: 10 см
Число π: 3.14
Количество оборотов: 5
Длина окружности колеса: 31.40 см
Пройденное расстояние: 157.00 см
PS C:\prog\robo>
```

Можно сделать программу более интерактивной, запросив данные у пользователя:

```
# Запрашиваем данные у пользователя

wheel_diameter = float(input("Введите диаметр колеса робота (в см):
"))

rotations = int(input("Введите количество оборотов колеса: "))
```

```
pi = 3.14


# Вычисляем длину окружности и пройденное расстояние
circumference = pi * wheel_diameter
distance = circumference * rotations


# Выводим результаты
print(f"\nДлина окружности колеса: {circumference:.2f} см")
print(f"Робот пройдёт расстояние: {distance:.2f} см")
```

Запуск:

```
PS C:\prog\robo> python robot.py
Введите диаметр колеса робота (в см): 40
Введите количество оборотов колеса: 34

Длина окружности колеса: 125.60 см
Робот пройдёт расстояние: 4270.40 см
PS C:\prog\robo>
```

Пример использования операторов:

Оператор	Код	Что делает
`+`	`a + b`	Сложение
`-`	`a - b`	Вычитание

Оператор	Код	Что делает
`*`	`a * b`	Умножение
`/`	`a / b`	Деление с дробной частью
`//`	`a // b`	Целочисленное деление
`%`	`a % b`	Остаток от деления
`**`	`a ** b`	Возведение в степень

---

7 1

3.1 Типы данных и операторы

5 из 15 шагов пройдено

0 из 5 баллов получено

Преобразование типов данных

Иногда нужно превращать одно значение в другое:

```
a = int("123")      # строка в число


b = str(456)        # число в строку


c = bool(1)         # любое число ≠ 0 → True


d = float(5)        # 5.0
```

Пример:

```
sensor_data = "25"

temp = int(sensor_data)

print("Температура:", temp)
```

Управление роботом на основе данных от пользователя

Цель программы

Показать, как важно **правильно преобразовывать данные**, например:

- Строка → Число (для расчётов)
- Число → Строка (для вывода)
- Значение → Логический тип (для принятия решений)

Что используется:

- Преобразование типов: `int()`, `str()`, `bool()`, `float()`
- Переменные
- Функции ввода/вывода: `input()`, `print()`
- Условия: `if/else`

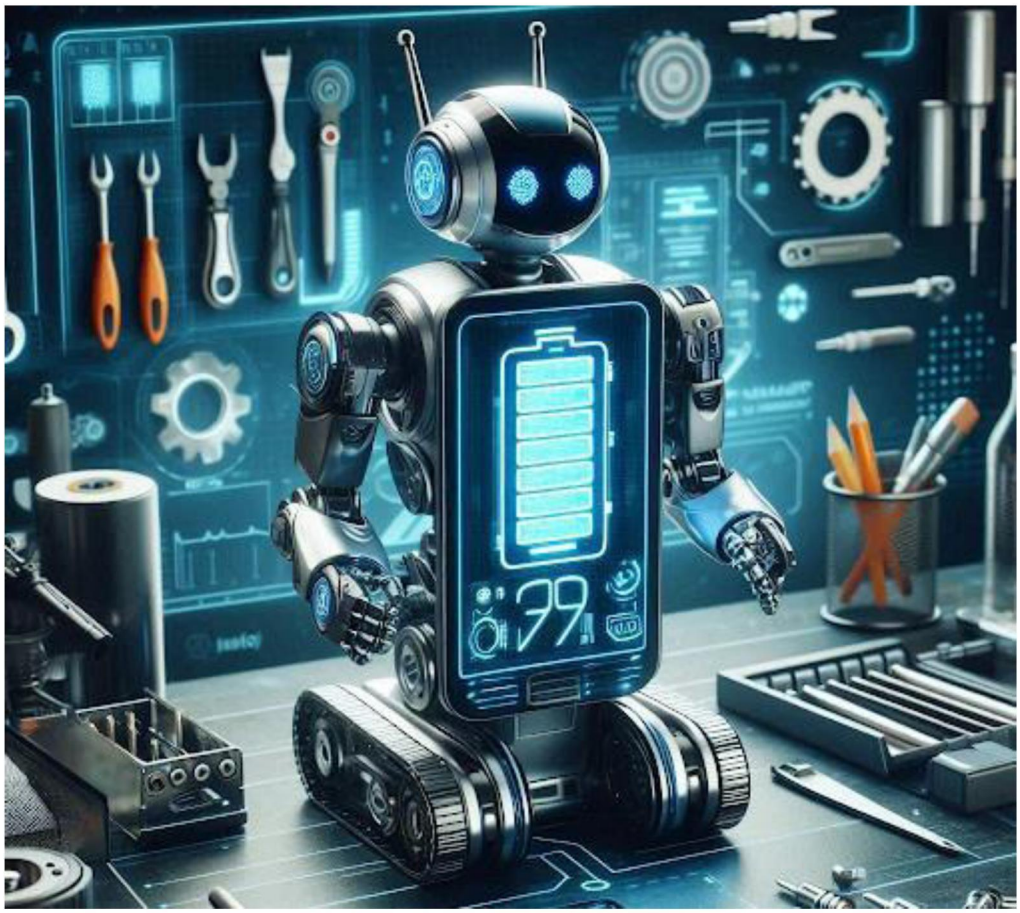
Пример программы:

```
# Получаем данные от пользователя (все значения изначально строки)

robot_name = input("Введите имя робота: ")

sensor_data = input("Введите показания датчика температуры (°C): ")

battery_level_str = input("Введите уровень заряда батареи (%): ")

is_activated_str = input("Робот активирован? (да/нет): ")



# Преобразуем типы данных

temperature = int(sensor_data)                    # строка → целое число

battery_level = float(battery_level_str)          # строка → вещественное
число

is_activated = bool(is_activated_str.lower() == "да")  # да → True,
нет → False
```

```
# Выводим информацию о состоянии робота
print("\n📊 Состояние робота:")
print("Имя робота: " + robot_name)
print("Температура: " + str(temperature) + " °C")
print("Заряд батареи: " + str(battery_level) + "%")
print("Активирован: " + str(is_activated))


# Проверяем возможность движения
if battery_level > 0 and is_activated:
    print(f"\n☑ {robot_name} готов к движению.")
elif battery_level <= 0:
    print(f"\n⚠ {robot_name} не может двигаться. Батарея разряжена.")
else:
    print(f"\n⛔ {robot_name} не активирован. Движение невозможно.")
```

Запуск:

```
PS C:\prog\robo> python robot.py
Введите имя робота: Линус
Введите показания датчика температуры (°C): 68
Введите уровень заряда батареи (%): 23
Робот активирован? (да/нет): да

📊 Состояние робота:
Имя робота: Линус
Температура: 68 °C
Заряд батареи: 23.0%
Активирован: True

✅ Линус готов к движению.
PS C:\prog\robo>
```



Объяснение преобразований:

Исходный тип	Целевой тип		Пример
			`"25"` → `25`
		`str(456)`	`45` → `"45"`
			`"45.7"` → `45.7`
	`bool`	`bool(... == "да")`	`"да"` → `True`, `"нет"` → `False`

7 1

3.1 Типы данных и операторы

6 из 15 шагов пройдено

0 из 5 баллов получено

Условные операторы

Условия позволяют роботу принимать решения.

```
if condition:
    # действие, если условие истинно
else:
    # действие, если ложно
```

Пример:

```
distance = 30
if distance < 10:
    print("Стоп! Препятствие рядом!")
else:
```