38 примеров Python для Cubroid

Источник: https://python.cubroid.com/js/snippets.js?v=20261103
Снимок источника: 11.11.2026. Примеры взяты из русского раздела действующей среды.
Открывать и выполнять: https://python.cubroid.com/ — выберите «Русский».
Это программы для встроенного модуля cubroid в Python Studio. В обычном Python
без среды производителя этот модуль не предоставляется.

Перед запуском подключите только нужные блоки своей группы. Для движения
подготовьте свободное место; перед касанием конструкции остановите программу.
Проверка редактора: синтаксис всех 38 файлов разобран Python, тексты сверены
с текущим источником. Работа на физическом наборе в рамках этой проверки
не испытывалась.

Задания с Teachable Machine требуют собственной обученной модели: замените
PASTE_TM_URL_HERE или PASTE_TM_AUDIO_URL_HERE её адресом. Названия классов
должны совпадать с Class 1 / Class 2 в примере либо быть изменены одновременно
в обучении и коде. Это параметры, а не непереведённые учебные инструкции.

Известное ограничение текущей среды: в ai.js распознавание речи задано для ko-KR,
а синтез речи — для ko-KR/en-US. В русском примере voice_motor подсказка русская,
проверка текста — английская. Примеры voice_motor, tts_basic, tts_sensor и ai_combo
сохранены для изучения алгоритмов; работу русской речи нельзя считать проверенной.
До исправления среды используйте задания с кнопкой, светом, камерой и моторами.

API и строковые параметры вроде red, happy, smile — рабочие идентификаторы.
Их перевод внутри вызова может нарушить программу. Пояснения и комментарии русские.

01. Вперёд/Назад

□□ Двигатель постоянного тока • dc_motor_basic.py

```
from cubroid import dc_motor, sleep

# motor1=левое колесо, motor2=правое колесо
# Вперёд 1с → Назад 1с
dc_motor.go()
sleep(1)
dc_motor.back()
sleep(1)
dc_motor.stop()
print("Готово!")
```

02. Езда в цикле

□□ Двигатель постоянного тока • dc_motor_loop.py

```
from cubroid import dc_motor, sleep

# Повторить 3 раза: вперёд → назад
for i in range(3):
    print(f"Цикл {i+1}")
    dc_motor.go()
    sleep(0.5)
    dc_motor.back()
    sleep(0.5)
dc_motor.stop()
```

03. Езда с поворотом

□□ Двигатель постоянного тока • dc_motor_turn.py

```
from cubroid import dc_motor, sleep

# Нарисовать квадрат: (вперёд → поворот на 90°) x4
# go() едет, пока не остановишь → используй sleep для задания времени движения
# right(сек) поворачивает указанное время и останавливается сам
for i in range(4):
    dc_motor.go()          # начать движение вперёд
    sleep(1)               # ехать прямо 1с
    dc_motor.right(1.5)    # ~1.5с поворот ≈ 90° (подстрой под робота)
dc_motor.stop()
print("Квадрат готов!")
```

04. Управление скоростью

□□ Двигатель постоянного тока • dc_motor_speed.py

```
from cubroid import dc_motor, sleep

# speed(left, right) — -255~255 (отрицательное=назад)
# Прямо: одинаковая скорость с обеих сторон
for spd in range(50, 256, 50):
    dc_motor.speed(spd, spd)
    print(f"Скорость: {spd}")
    sleep(0.5)

# Поворот налево: слева медленно, справа быстро
dc_motor.speed(50, 200)
sleep(1)
dc_motor.stop()
```

05. Отдельный двигатель

□□ Двигатель постоянного тока • dc_motor_individual.py

```
from cubroid import dc_motor1, dc_motor2, sleep

# dc_motor1 = левое колесо, dc_motor2 = правое колесо
# Управляй каждым двигателем независимо!

# Только левое вперёд
dc_motor1.go()
sleep(1)
dc_motor1.stop()

# Только правое вперёд
dc_motor2.go()
sleep(1)
dc_motor2.stop()

# Противоположно: вращение на месте
dc_motor1.go()
dc_motor2.back()
sleep(1)

# Отдельное управление скоростью (0~255: 0=стоп, 255=макс)
dc_motor1.speed(100)   # левое медленнее
dc_motor2.speed(200)   # правое быстрее
sleep(1)

dc_motor1.stop()
dc_motor2.stop()
print("Управление отдельными двигателями готово!")
```

06. Сменить выражение

 Светодиодная матрица • led_face.py

```
from cubroid import led, sleep

# Менять выражения лица
faces = ["happy", "sad", "angry", "surprise", "sleepy"]
for f in faces:
    led.face(f)
    print(f"Лицо: {f}")
    sleep(1)
led.off()
```

07. Буквы и цифры

 Светодиодная матрица • led_letter.py

```
from cubroid import led, sleep

# Показать буквы A~E
for ch in "ABCDE":
    led.letter(ch)
    print(f"Буква: {ch}")
    sleep(0.5)

# Обратный отсчёт чисел
for n in range(5, 0, -1):
    led.number(n)
    sleep(0.5)
led.off()
```

08. Бегущий текст

 Светодиодная матрица • led_text.py

```
from cubroid import led, sleep

# Прокрутить текст по буквам
led.text("HELLO", 0.5)
sleep(1)
led.text("12345", 0.3)
sleep(1)
led.off()
```

09. Галерея значков

 Светодиодная матрица • led_icon.py

```
from cubroid import led, sleep

# Показать разные значки
icons = ["heart", "star", "sun", "moon", "cloud",
         "rain", "snow", "music", "check", "x",
         "arrow_up", "arrow_down", "arrow_left", "arrow_right"]

for icon in icons:
    led.show(icon)
    print(f"Значок: {icon}")
    sleep(0.5)
led.off()
```

10. Гамма До Ре Ми

 Звук (Пьезозуммер) • sound_doremi.py

```
from cubroid import sound, sleep

# до ре ми фа соль ля си (октава 5)
notes = ["do", "re", "mi", "fa", "sol", "la", "si", "high_do"]
for n in notes:
    sound.play(n)
    print(f"♪ {n}")
    sleep(0.3)
sound.off()
```

11. Мелодия школьного звонка

 Звук (Пьезозуммер) • sound_melody.py

```
from cubroid import sound

# Мелодия школьного звонка (нота, доля)
sound.melody([
    ("sol", 1), ("sol", 1), ("la", 1), ("la", 1),
    ("sol", 1), ("sol", 1), ("mi", 2),
    ("sol", 1), ("sol", 1), ("mi", 1), ("mi", 1),
    ("re", 2), ("rest", 1), ("rest", 1),
], bpm=120)
```

12. Звуковые эффекты

 Звук (Пьезозуммер) • sound_fx.py

```
from cubroid import sound, sleep

# Сирена: чередовать две ноты
for i in range(5):
    sound.play("high_do")
    sleep(0.15)
    sound.play("mi")
    sleep(0.15)
sound.off()

sleep(0.5)

# Восходящий тон
for n in ["do", "re", "mi", "fa", "sol", "la", "si", "high_do"]:
    sound.play(n)
    sleep(0.1)
sound.off()
```

13. Светофор

 Главный светодиод • master_signal.py

```
from cubroid import master, sleep

# Светофор: Красный → Жёлтый → Зелёный
master.red()
sleep(2)
master.yellow()
sleep(1)
master.green()
sleep(2)
master.off()
```

14. Цвета радуги

 Главный светодиод • master_rainbow.py

```
from cubroid import master, sleep

# Радуга 7 цветов по порядку
master.rainbow(0.5)
master.off()
```

15. Мигание

 Главный светодиод • master_blink.py

```
from cubroid import master, sleep

# Мигать красным 5 раз
for i in range(5):
    master.red()
    sleep(0.3)
    master.off()
    sleep(0.3)
print("Мигание готово!")
```

16. Датчик приближения

 Датчики • sensor_proximity.py

```
from cubroid import dc_motor, proximity_sensor, master, sleep

# Датчик приближения: назад + красный при приближении
while True:
    if proximity_sensor.is_near():
        master.red()
        dc_motor.back()
        sleep(0.5)
        dc_motor.stop()
    else:
        master.green()
        dc_motor.go()
    sleep(0.1)
```

17. Датчик касания/света

 Датчики • sensor_touch.py

```
from cubroid import dc_motor, light_touch_sensor, sound, sleep

# Касание → вперёд + звук, вывести яркость
while True:
    if light_touch_sensor.is_pressed():
        sound.play("do")
        dc_motor.go()
        sleep(1)
        dc_motor.stop()
        sound.off()
    bright = light_touch_sensor.brightness()
    print(f"Яркость: {bright}%")
    sleep(0.2)
```

18. График данных датчика

 График данных • plotter_sensor.py

```
from cubroid import proximity_sensor, light_touch_sensor, sleep

# □ График данных: график датчиков в реальном времени!
# Нажми кнопку □ рядом с консолью, чтобы открыть график

plotter.show()
plotter.clear()

while True:
    bright = light_touch_sensor.brightness()
    touch = 100 if light_touch_sensor.is_pressed() else 0
    near = 100 if proximity_sensor.is_near() else 0

    plotter.add("Яркость", bright)
    plotter.add("Касание", touch)
    plotter.add("Приближение", near)

    sleep(0.2)
```

19. Создать свои действия

 Мои команды (def) • def_basic.py

```
from cubroid import dc_motor, sleep

# Создать свои собственные команды
def forward():
    dc_motor.go()
    sleep(1)
    dc_motor.stop()

def spin():
    dc_motor.speed(200, -200)   # левое вперёд, правое назад → вращение на месте
    sleep(0.5)
    dc_motor.stop()

# Использовать мои команды
forward()
spin()
forward()
print("Готово!")
```

20. Робот-танцор (def+цикл)

 Мои команды (def) • def_dance.py

```
from cubroid import dc_motor, led, sound, sleep

# Создать танцевальные движения как функции
def step_right():
    dc_motor.speed(200, 50)
    sleep(0.3)
    dc_motor.stop()

def step_left():
    dc_motor.speed(50, 200)
    sleep(0.3)
    dc_motor.stop()

def wiggle():
    for i in range(3):
        step_right()
        step_left()

def celebrate():
    led.face("love")
    sound.play("high_do")
    sleep(0.5)

# Время танцевать!
print(" Танец начинается!")
for i in range(2):
    print(f"Танец #{i+1}")
    wiggle()
    celebrate()

dc_motor.stop()
led.face("happy")
print(" Танец завершён!")
```

21. Робот-питомец

 Проекты • combo_pet.py

```
from cubroid import dc_motor, proximity_sensor, led, sound, master, sleep

# Робот-питомец: эмоциональный робот, реагирующий на датчик
led.face("happy")
master.green()
print(" Робот-питомец проснулся!")

while True:
    if proximity_sensor.is_near():
        # Радуется, когда кто-то рядом
        led.face("love")
        master.rgb(255, 100, 200)
        sound.play("sol")
        sleep(0.2)
        sound.play("high_do")
        sleep(0.3)

        # Приветствие (покачивание влево-вправо)
        dc_motor.speed(150, 50)
        sleep(0.3)
        dc_motor.speed(50, 150)
        sleep(0.3)
        dc_motor.stop()
    else:
        # Сонный, когда никого нет рядом
        led.face("sleepy")
        master.rgb(50, 50, 100)
    sleep(0.3)
```

22. Машина-викторина

 Проекты • combo_quiz.py

```
from cubroid import led, sound, master, light_touch_sensor, sleep

# Машина-викторина: отвечай Да/Нет касанием
questions = [
    "В1: 1+1=2?",
    "В2: 3>5?",
    "В3: Python — язык программирования?",
]
answers = [True, False, True]
score = 0

for i in range(len(questions)):
    print(f"\n{questions[i]}")
    led.text(f"Q{i+1}", 0.3)
    sleep(1)
    print("Касание=Да, Без касания=Нет (3 сек)")
    master.yellow()

    touched = False
    for t in range(30):
        if light_touch_sensor.is_pressed():
            touched = True
        sleep(0.1)

    correct = (touched == answers[i])
    if correct:
        score = score + 1
        master.green()
        sound.play("high_do")
        led.face("happy")
        print("Верно!")
    else:
        master.red()
        sound.play("do")
        led.face("sad")
        print("Неверно!")
    sleep(1)

led.off()
master.off()
print(f"\nИтоговый счёт: {score}/{len(questions)}")
```

23. TM Классификация → Двигатель

 Обучаемая машина • tm_basic.py

```
from cubroid import dc_motor, master, sleep
from cubroid.ai import camera, teachable

# 1. Запустить камеру
camera.start()
sleep(1)

# 2. Загрузить модель Teachable Machine
# Примечание: Введи URL в правой панели и нажми Загрузить
teachable.load("PASTE_TM_URL_HERE")
sleep(2)

# 3. Действовать по результату классификации
while True:
    result = teachable.predict()
    name = result["label"]
    score = result["score"]
    print(f"Класс: {name} ({score}%)")

    if name == "Class 1":
        dc_motor.go()
        master.green()
    elif name == "Class 2":
        dc_motor.back()
        master.red()
    else:
        dc_motor.stop()
        master.off()
    sleep(0.5)
```

24. Цвет → Двигатель

 Определение цвета • color_motor.py

```
from cubroid import dc_motor, master, sleep
from cubroid.ai import camera, color

# Запустить камеру (определение цвета автоматическое)
camera.start()
sleep(1)

# Управлять роботом по обнаруженному цвету
while True:
    c = color.name_en()
    print(f"Цвет: {c}")

    if c == "red":
        dc_motor.stop()
        master.red()
    elif c == "green":
        dc_motor.go()
        master.green()
    elif c == "blue":
        dc_motor.back()
        master.rgb(0, 0, 255)
    sleep(0.3)
```

25. Цвет → Светодиод

 Определение цвета • color_led.py

```
from cubroid import master, led, sleep
from cubroid.ai import camera, color

camera.start()
sleep(1)

# Показать обнаруженный цвет на главном светодиоде
while True:
    r, g, b = color.rgb()
    master.rgb(r, g, b)
    c = color.name_en()
    if c:
        led.letter(c[0].upper())
    sleep(0.3)
```

26. Голос → Двигатель

□□ Распознавание голоса • voice_motor.py

Русская речь в текущей версии среды требует исправления. Этот пример показывает алгоритм; см. пояснение перед запуском.

```
from cubroid import dc_motor, master, led, sleep
from cubroid.ai import voice

# Запустить распознавание голоса
voice.start()
print("Скажи: вперёд, назад, налево, направо")

while True:
    text = voice.text()
    if text:
        print(f"Услышано: {text}")
        if "go" in text or "forward" in text:
            dc_motor.go(1)        # вперёд 1с, авто-стоп
            master.green()
            led.face("happy")
        elif "back" in text or "reverse" in text:
            dc_motor.back(1)      # назад 1с, авто-стоп
            master.yellow()
            led.face("surprise")
        elif "left" in text:
            dc_motor.left()
            master.rgb(0, 0, 255)
        elif "right" in text:
            dc_motor.right()
            master.rgb(255, 0, 255)
        sleep(1)
    sleep(0.2)
```

27. Робот говорит

 Текст в речь • tts_basic.py

Русская речь в текущей версии среды требует исправления. Этот пример показывает алгоритм; см. пояснение перед запуском.

```
from cubroid.ai import tts
from cubroid import led, sleep

# Робот говорит (TTS) — say() ждёт окончания речи
led.face("happy")
tts.say("Привет! Я робот Куро.")
sleep(0.5)
tts.say("Давай учиться вместе сегодня!")
led.off()
```

28. Датчик → Речь

 Текст в речь • tts_sensor.py

Русская речь в текущей версии среды требует исправления. Этот пример показывает алгоритм; см. пояснение перед запуском.

```
from cubroid import proximity_sensor, master, sleep
from cubroid.ai import tts

# Датчик приближения + TTS
print("Поднеси руку поближе!")
count = 0

while True:
    if proximity_sensor.is_near():
        count = count + 1
        master.green()
        tts.say(f"Обнаружение {count}! Рад тебя видеть!")
        sleep(2)
        master.off()
    sleep(0.2)
```

29. Цвет+Голос+TTS

 Объединённый ИИ • ai_combo.py

Русская речь в текущей версии среды требует исправления. Этот пример показывает алгоритм; см. пояснение перед запуском.

```
from cubroid import dc_motor, master, sleep
from cubroid.ai import camera, color, voice, tts

# Цвет + Голос + TTS вместе
camera.start()
voice.start()
sleep(1)

print("ИИ-робот готов!")
tts.say("Привет! Покажи мне цвет или дай команду!")
sleep(0.5)

while True:
    text = voice.text()
    if text:
        print(f"Голос: {text}")
        if "forward" in text or "go" in text:
            tts.say("Еду вперёд!")
            dc_motor.go(1)         # 1с вперёд
        elif "back" in text or "reverse" in text:
            tts.say("Еду назад!")
            dc_motor.back(1)       # 1с назад
        elif "left" in text:
            tts.say("Поворачиваю налево!")
            dc_motor.left(1.5)     # ~1.5с поворот ≈ 90°
        elif "right" in text:
            tts.say("Поворачиваю направо!")
            dc_motor.right(1.5)    # ~1.5с поворот ≈ 90°

    c = color.name_en()
    if c == "red":
        master.red()
    elif c == "green":
        master.green()
    elif c == "blue":
        master.rgb(0, 0, 255)
    sleep(0.3)
```

30. Пальцы → Двигатель

 Определение руки • hand_motor.py

```
from cubroid import dc_motor, master, led, sleep
from cubroid.ai import camera, hand

camera.start()
hand.start()
sleep(2)

print("Управляй роботом своими пальцами!")

while True:
    fingers = hand.fingers()
    print(f"Пальцы: {fingers}")
    led.number(fingers)

    if fingers >= 4:
        dc_motor.go()
        master.green()
    elif fingers == 1:
        dc_motor.back()
        master.red()
    elif fingers == 0:
        dc_motor.stop()
        master.off()
    sleep(0.3)
```

31. Камень-ножницы-бумага

 Определение руки • hand_gesture.py

```
from cubroid import sound, master, led, sleep
from cubroid.ai import camera, hand
import random

camera.start()
hand.start()
sleep(2)

print("Играй в камень-ножницы-бумагу!")
print("Примечание: Покажи камере тыльную сторону руки!")
score = 0

while True:
    g = hand.gesture()
    if g in ["scissors", "rock", "paper"]:
        choices = ["scissors", "rock", "paper"]
        robot = random.choice(choices)

        print(f"Ты: {g} против Робота: {robot}")

        if g == robot:
            master.yellow()
            led.face("surprise")
            sound.play("mi")
            print("Ничья!")
        elif (g == "scissors" and robot == "paper") or (g == "rock" and robot == "scissors") or (g
== "paper" and robot == "rock"):
            score = score + 1
            master.green()
            led.face("happy")
            sound.play("high_do")
            print(f"Ты выиграл! (Счёт: {score})")
        else:
            master.red()
            led.face("sad")
            sound.play("do")
            print("Ты проиграл!")
        sleep(2)
    sleep(0.3)
```

32. Подсчёт лиц

 Определение лица • face_count.py

```
from cubroid import master, sound, led, sleep
from cubroid.ai import camera, face

camera.start()
face.start()
sleep(2)

print("Обнаружение лиц!")

while True:
    n = face.count()
    print(f"Обнаружено лиц: {n}")

    if n == 0:
        master.off()
        led.face("sleepy")
    elif n == 1:
        master.green()
        led.face("happy")
        sound.play("do")
    elif n >= 2:
        master.rgb(255, 100, 200)
        led.face("love")
        sound.play("sol")
    sleep(0.5)
```

33. Поза рук → Двигатель

 Определение позы • pose_arm.py

```
from cubroid import dc_motor, master, led, sleep
from cubroid.ai import camera, pose

camera.start()
pose.start()
sleep(2)

print("Подними руки, чтобы управлять роботом!")
print("Обе вверх: вперёд | Обе вниз: стоп")
print("Левая вверх: поворот налево | Правая вверх: поворот направо")

while True:
    left = pose.left_wrist()
    right = pose.right_wrist()
    nose = pose.nose()

    if left and right and nose:
        l_up = left["y"] < nose["y"]
        r_up = right["y"] < nose["y"]

        if l_up and r_up:
            dc_motor.go()
            master.green()
            led.face("happy")
        elif l_up:
            dc_motor.left()
            master.rgb(0, 0, 255)
        elif r_up:
            dc_motor.right()
            master.rgb(255, 0, 255)
        else:
            dc_motor.stop()
            master.off()
            led.face("sleepy")
    sleep(0.2)
```

34. Робот-танцор

 Определение позы • pose_dance.py

```
from cubroid import dc_motor, led, sound, master, sleep
from cubroid.ai import camera, pose

camera.start()
pose.start()
sleep(2)

print("Двигай телом и танцуй с роботом!")

prev_action = ""
while True:
    lw = pose.left_wrist()
    rw = pose.right_wrist()
    nose = pose.nose()

    if lw and rw and nose:
        l_up = lw["y"] < nose["y"]
        r_up = rw["y"] < nose["y"]

        if l_up and r_up and prev_action != "both":
            dc_motor.speed(200, 50)
            led.face("love")
            master.rainbow(0.2)
            sound.play("high_do")
            prev_action = "both"
        elif l_up and not r_up and prev_action != "left":
            dc_motor.speed(50, 200)
            led.face("happy")
            master.rgb(0, 0, 255)
            sound.play("sol")
            prev_action = "left"
        elif r_up and not l_up and prev_action != "right":
            dc_motor.speed(200, 50)
            led.face("happy")
            master.rgb(255, 0, 255)
            sound.play("mi")
            prev_action = "right"
        elif not l_up and not r_up and prev_action != "down":
            dc_motor.stop()
            led.face("sleepy")
            master.off()
            prev_action = "down"
    sleep(0.2)
```

35. Обнаружение → Двигатель

 Определение объектов • detect_basic.py

```
from cubroid import dc_motor, master, sleep
from cubroid.ai import camera, detect

camera.start()
detect.start()
sleep(2)

print("Обнаружение объектов!")

while True:
    obj = detect.best()
    if obj:
        name = obj["class"]
        score = obj["score"]
        print(f"Обнаружено: {name} ({score}%)")

        if name == "person":
            dc_motor.go()
            master.green()
        else:
            dc_motor.stop()
            master.yellow()
    else:
        dc_motor.stop()
        master.off()
    sleep(0.5)
```

36. Подсчёт объектов

 Определение объектов • detect_count.py

```
from cubroid import sound, master, led, sleep
from cubroid.ai import camera, detect

camera.start()
detect.start()
sleep(2)

print("Подсчёт видимых объектов!")

while True:
    items = detect.all()
    n = len(items)
    print(f"Обнаружено объектов: {n}")

    if n > 0 and n <= 9:
        led.number(n)
        master.green()
        if n >= 3:
            sound.play("high_do")
    else:
        led.face("sleepy")
        master.off()

    for item in items:
        name = item["class"]
        score = item["score"]
        print(f"  - {name} ({score}%)")
    sleep(1)
```

37. Выражение → Робот

 Определение выражения • fm_expression.py

```
from cubroid import led, master, sound, sleep
from cubroid.ai import camera, face_mesh

camera.start()
face_mesh.start()
sleep(2)

print("Сделай выражение лица!")
# Выражения: smile, big_smile, surprise, eyes_closed, neutral

while True:
    expr = face_mesh.expression()
    if expr:
        print(f"Выражение: {expr}")

        if expr == "smile" or expr == "big_smile":
            led.face("happy")
            master.green()
            sound.play("sol")
        elif expr == "surprise":
            led.face("surprise")
            master.yellow()
            sound.play("high_do")
        elif expr == "eyes_closed":
            led.face("sleepy")
            master.rgb(100, 0, 255)
            sound.play("mi")
        else:
            led.face("happy")
            master.off()
    sleep(0.5)
```

38. ТМ Аудио → Двигатель

 Классификация звука • audio_basic.py

```
from cubroid import dc_motor, master, led, sleep
from cubroid.ai import audio

# Загрузить модель Teachable Machine Audio
# Примечание: Введи URL в правой панели и нажми Загрузить
audio.load("PASTE_TM_AUDIO_URL_HERE")
sleep(2)

print("Издавай звуки!")

while True:
    result = audio.predict()
    name = result["label"]
    score = result["score"]
    print(f"Звук: {name} ({score}%)")

    if name == "Class 1":
        dc_motor.go()
        master.green()
        led.face("happy")
    elif name == "Class 2":
        dc_motor.back()
        master.red()
        led.face("surprise")
    else:
        dc_motor.stop()
        master.off()
        led.face("sleepy")
    sleep(0.5)
```