Temat: Naucz się tekstowego języka Python z Codey Rocky
- część 1 Zdarzenia

kl7-8 Szkoła Podstawowa

Przedmiot: informatyka

Autor: Sylwester Zasoński

Czas trwania: 1h lekcyjna

Cele ogólne:

- Rozwijanie kompetencji miękkich (umiejętność pracy zespołowej, logiczne, algorytmiczne myślenie)
- Wprowadzenie języka Python

Cele operacyjne:

Uczeń:

- posługuje się komputerem lub innym urządzeniem cyfrowym oraz urządzeniami zewnętrznymi przy wykonywaniu zadania
- uczeń zapoznaje się z zasadami składni języka Python
- uczeń potrafi pisać proste skrypty w języku Python

Metody:

praca indywidualna/zespołowa, wykład

Środki dydaktyczne:

1. Robot Codey Rocky + kabel/adapter do połączenia
2. Komputer z zainstalowaną aplikacją mBlock

http://www.mblock.cc/mblock-software/

Przebieg zajęć:

Celem lekcji jest poznanie języka tekstowego Python.

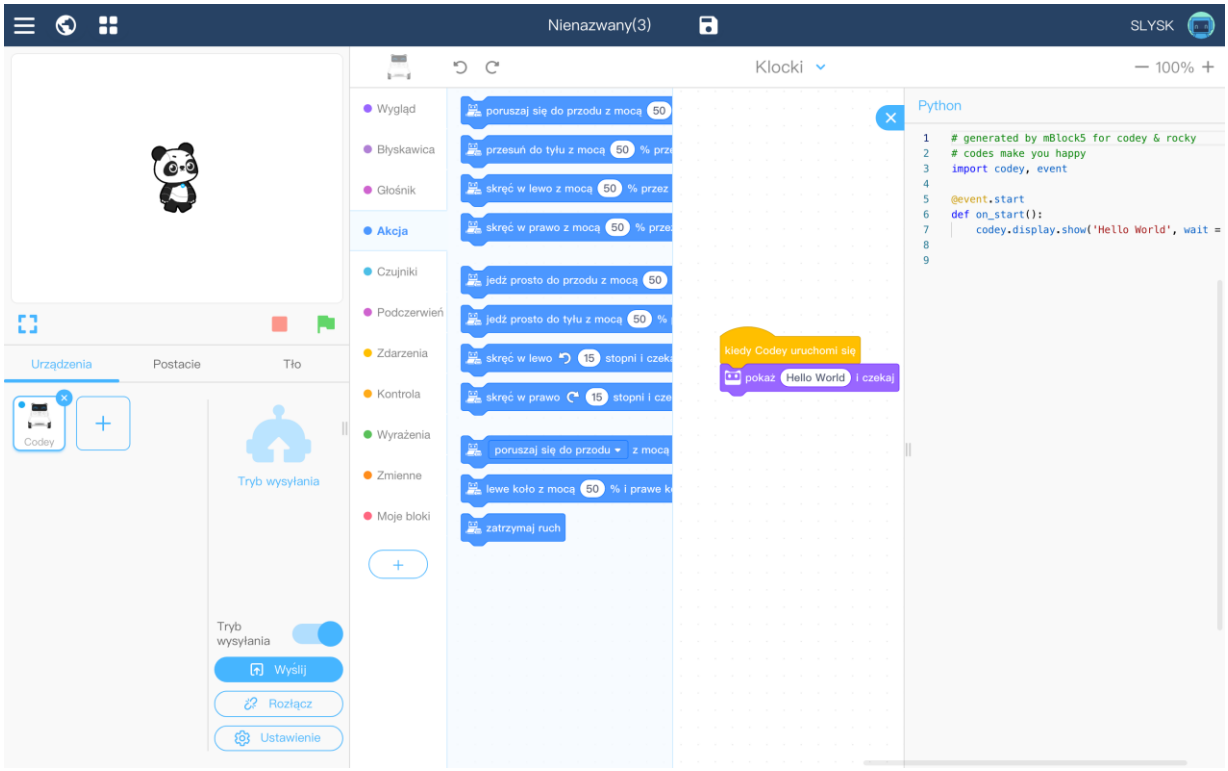
Python – język programowania wysokiego poziomu ogólnego przeznaczenia, o rozbudowanym pakiecie bibliotek standardowych, którego ideą przewodnią jest czytelność i klarowność kodu źródłowego. Jego składnia cechuje się przejrzystością i zwięzłością

Źródło definicji Wikipedia

W przypadku robota Codey Rocky mowa jest o języku microPython.

MicroPython to implementacja języka programowania Python 3, napisana w języku C, zoptymalizowana do pracy na mikrokontrolerze. MicroPython to pełny kompilator Pythona, który działa na sprzęcie mikrokontrolera. Codey Rocky oparty jest na mikrokontrolerze ESP32, a program mBlock5 służy jako kompilator.

MBlock5 (oparty na Scratch3.0) dostępny na komputer posiada unikalną funkcję podglądu kodu zapisanego w Scratch do języka Python (opcja dostępna w przypadku programowania robota Codey Rocky oraz duszków-obiektów na ekranie). Chcąc podejrzeć kod dla Codey Rocky, tryb wysyłania kodu musi być włączony.



Podgląd kodu będzie widoczny po kliknięciu zakładki w prawym górnym rogu.

Ale mBlock5 umożliwia również napisanie i wgranie do robota Codey Rocky kodu w języku Python. Wystarczy zmienić (zlokalizowaną w górnym prawym rogu) zakładkę Blocki na Python. Dzięki czemu możemy użyć aplikacji jako IDE do pisania kodów w Pythonie. Dzięki temu nauka podstaw Pythona może stać się łatwiejsza i przyjemniejsza.

Poniżej znajdziesz dostępne komendy pogrupowane na kategorie, podobnie jak w Scratch. Dodatkowo każdą komendę staram się opisać i wyjaśnić.

Zanim jednak przejdziesz do kategorii komend, ważna rzecz!

Kod koniecznie rozpocznij od zaimportowania potrzebnych bibliotek. Użyj do tego komendy import

Przede wszystkim potrzebne będą: codey – jeżeli chcesz używać tylko głowy, rocky - jeżeli chcesz użyć również podwozia, event – to biblioteka odpowiedzialna za zdarzenia, time – odpowiada za czas np. Ile sekund ma być wykonywana dane polecenie.

Pierwsza linijka powinna wyglądać tak:

```
import rocky, time, codey, event
```

Zaimportowane zostały 4 biblioteki.

Ponieważ kod rozpoczynamy zawsze od jednej z koment znajdujących się w kategorii **Zdarzenia** poniżej znajdziesz opisane komendy właśnie z tej kategorii.

Pamiętaj!
Napisany w Pythonie kod zadziała tylko wtedy, gdy będzie włączony **tryb przesyłania!**
Ten tryb polega na wysłaniu do pamięci robota kodu, dzięki czemu umożliwia robotowi pracę samodzielnie bez konieczności ponownego uruchamiania

komputera. W ten sposób zaprogramowane jest większość urządzeń, które używamy w życiu codziennym.

Skoro event to biblioteka odpowiedzialna za zdarzenia to każda komenda z kategori **Zdarzenia** będzie rozpoczynała się od @event. Po @event pojawi się kropka oddzielająca komendy. Potem zależnie od rodzaju komendy może być np. Start, shaked, button_a_pressed, tilted_left/right, ears_up/down, less/greater_than czy received.

Przykładowa linijka:

```
@event.start
```

W kolejnym wierszu zawsze pojawi się:

def on_ - po podkreślniku dodasz jedną z powyżej wymienionych koment, w tym przypadku start

Całe 2 linijki będą wyglądać następująco:

```
@event.start
def on_start():
```

Kod ten oznacza, że wszystko co zostanie umieszczone poniżej zostanie wykonane po uruchomieniu robota Codey Rocky, np.:

```
import codey, event

@event.start

def on_start():

    codey.display.show('hello')

    codey.speaker.play_melody('hello.wav')
```

Najpierw importujemy biblioteki, tu wystarczy Codey oraz event (1 linijka). Następnie ustalamy, kiedy kod zostanie wykonany (2,3 linijka). Tu po uruchomieniu urządzenia. Potem widzisz 2 linijki kodu odpowiedzialne za to co zostanie wykonane po uruchomieniu - wyświetlacz pokaże napis hello oraz robot odtworzy wgrany plik dźwiękowy o nazwie hello.wav To najprostszy przykład skryptu z użyciem kodu inicjującego.

W kategorii **Zdarzenia** znajdziesz jeszcze poniższe możliwości:

Jeżeli chcesz, aby skrypt został uruchomiony po potrząśnięciu robotem, użyj

```
@event.shaked
def on_shaked():
```

Jeżeli chcesz, aby skrypt został uruchomiony po naciśnięciu przycisku **A**, użyj

```
@event.button_a_pressed
def on_button_a_pressed():
```

> Pamiętaj!
> Możesz zmieniać przyciski wpisując **B** lub **C**

Jeżeli chcesz, aby skrypt został uruchomiony po przechyleniu robota w lewo

```
@event.tilted_left
def on_tilted_left():
```

Analogicznie będzie wyglądał skrypt przy przechyleniu w prawo, wystarczy podmienić **left** na **right**

Jeżeli chcesz, aby skrypt został uruchomiony po przechyleniu robota do przodu

```
@event.ears_up
def on_ears_up():
```

Analogicznie będzie wyglądał skrypt przy przechyleniu do tyłu, zmień tylko **up** na **down**

Jeżeli chcesz, aby skrypt został uruchomiony, gdy natężenie dźwięku będzie większe niż 10, wartość tą oczywiście możesz modyfikować

```
@event.greater_than(10, 'sound_sensor')
def on_greater_than():
```

Jeżeli chcesz, aby skrypt został uruchomiony, gdy wartość **timer** (czas) będzie większa niż 100, wartość tą oczywiście możesz modyfikować

```
@event.greater_than(100, 'timer')
```

```
def on_greater_than():
```

Jeżeli chcesz, aby skrypt został uruchomiony, gdy intensywność światła będzie mniejsza od określonej wartości (skala od 0 do 100)

```
@event.less_than(5, 'light_sensor')
def on_less_than():
```

Jeżeli chcesz, aby skrypt został uruchomiony, gdy robot otrzyma komunikat o nazwie 'komunikat1', nazwę komunikatu można dowolnie modyfikować.

```
@event.received('komunikat1')
def on_received():
```

W kategorii **Zdarzenia** znajduje się jeszcze komenda

```
codey.broadcast('komunikat1')
```

Jest to komenda odpowiedzialna za nadawanie komunikatu, nazwę komunikatu można dowolnie modyfikować, lecz nazwa nie może zawierać znaków specjalnych.

Przykładowy kod z użyciem komunikatów wygląda tak

```
import codey, event

@event.start
def on_start():
    codey.display.show('hello')
    codey.broadcast('start')

@event.received('start')
def on_received():
    codey.speaker.play_melody('hello.wav')
```

1. Importujemy biblioteki

2. Po uruchomieniu robot pokazuje napis ‘hello’ i nadaje komunikat ‘start’
3. Gdy otrzyma komunikat ‘start’ odtwarza dźwięk ‘hello.wav’

Mam nadzieje, że powyższe objaśnienia pomogły ci zrozumieć dostępne dla robota Codey Rocky komendy inicjujące skrypty. Zapraszam do lektury kolejnej części, w której omówię bloki związane z dźwiękiem i obrazem.