Micro:bit – pierwsze kroki z Pythonem

Autorzy: Dawid Osuchowski, Filip Kłębczyk

Ściągawka – przydatne funkcje

W tym dokumencie znajduje się zbiór wszystkich używanych podczas zajęć funkcji oraz elementów, wraz z krótkimi opisami ich działania.

Przygotowania przed zajęciami:

1. Oprogramowanie – instalacja na wszystkich komputerach edytora Mu, sprawdzenie czy edytor się uruchamia i czy jest możliwe sfleszowanie (zapisanie pamięci flash) płytki Micro:bit. Dla wygody, by nie usuwać programu demonstracyjnego na wszystkich płytkach, do celów testowych najlepiej wykorzystywać tylko jedną płytkę.
2. Sprzęt – sprawdzenie, czy po podłączeniu płytek Micro:bit uruchamia się program demonstracyjny. Jeśli został już nadpisany innym programem (podczas testów lub wcześniejszych zajęć), należy go przywrócić zgodnie z instrukcją znajdującą się na końcu scenariusza pierwszej lekcji.

Lekcja 1

Wyrażenia/funkcje używane podczas lekcji:

- `from microbit import *` – używamy na początku pliku z kodem źródłowym, dzięki niemu dostajemy dostęp do wszystkich funkcji płytki Micro:bit
- `display.scroll("Ahoj, przygodo!")` – wypisuje na ekran “Ahoj, przygodo!” w postaci przewijanego ciągu znaków na ekran
- `display.show("Ahoj, przygodo!")` – wypisuje na ekran “Ahoj, przygodo!”, w postaci
  sekwencji znaków, występujących jeden po drugim
- `Image` – klasa używana do tworzenia obrazków, ma zdefiniowaną dużą liczbę obrazków standardowych:
  - Image.HEART
  - Image.HEART_SMALL
  - Image.CLOCK11
  - Image.CLOCK10

- Image.HAPPY
- Image.SMILE
- Image.SAD
- Image.CONFUSED
- Image.ANGRY
- Image.ASLEEP
- Image.SURPRISED
- Image.SILLY
- Image.FABULOUS
- Image.MEH
- Image.YES
- Image.NO
- Image.CLOCK12
- Image.ARROW_S
- Image.ARROW_SW
- Image.ARROW_W
- Image.ARROW_NW
- Image.TRIANGLE
- Image.TRIANGLE_LEFT
- Image.CHESSBOARD
- Image.DIAMOND
- Image.DIAMOND_SMALL
- Image.SQUARE
- Image.SQUARE_SMALL
- Image.RABBIT
- Image.COW
- Image.MUSIC_CROTCHET
- Image.MUSIC_QUAVER
- Image.MUSIC_QUAVERS
- Image.PITCHFORK
- Image.CLOCK9
- Image.CLOCK8
- Image.CLOCK7
- Image.CLOCK6
- Image.CLOCK5
- Image.CLOCK4
- Image.CLOCK3
- Image.CLOCK2
- Image.CLOCK1
- Image.ARROW_N
- Image.ARROW_NE
- Image.ARROW_E
- Image.ARROW_SE
- Image.XMAS
- Image.PACMAN
- Image.TARGET
- Image.TSHIRT
- Image.ROLLERSKATE
- Image.DUCK
- Image.HOUSE
- Image.TORTOISE
- Image.BUTTERFLY
- Image.STICKFIGURE
- Image.GHOST
- Image.SWORD
- Image.GIRAFFE
- Image.SKULL
- Image.UMBRELLA
- Image.SNAKE

- box = Image("99999:90009:90009:90009:99999") – zdefiniowanie własnego obrazka oraz przypisanie go do zmiennej box

Lekcja 2

- while True: – pętla nieskończona
- sleep(500) – funkcja usypia program na 500ms, zmieniając wartość wydłużamy lub skracamy czas uśpienia
- delay = 500 – przypisanie do zmiennej delay wartości 500

Lekcja 3

W kodzie możemy odwołać się do przycisków za pomocą obiektów button_a oraz button_b. Każdy z obiektów posiada metody (funkcje wywoływane na rzecz obiektu):

- is_pressed() – zwraca wartość True, jeżeli przycisk jest wciśnięty dokładnie w momencie wywołania metody, False w przeciwnym wypadku
- was_pressed() – zwraca wartość True lub False, aby wskazać, czy przycisk był wciśnięty od momentu rozpoczęcia działaniu programu lub ostatniego wywołania tej metody.
- get_presses() – zwraca liczbę naciśnięć przycisku do momentu wywołania tej metody, True i False w Pythonie to wartości logiczne zwracane najczęściej przez funkcje lub w wyniku porównań i używane w pętlach lub instrukcjach warunkowych.

Lekcja 4

- image_list = [Image.HAPPY, Image.SAD, Image.ANGRY, Image.CONFUSED] – zdefiniowanie listy image_list i wypełnienie jej obrazkami
- image_list[0] – dostanie się do wartości elementu przechowywanego w liście image_list na indeksie 0 (listy indeksowane są od 0, więc 0 jest pierwszym elementem listy)
- display.show(image_list[0]) – wyświetlenie obrazka będącego pierwszym elementem listy image_list
- if button_b.was_pressed(): – instrukcja warunkowa, która wykona się, jeżeli wciśnięty był przycisk B
- index += 1 – inkrementacja (zwiększenie wartości o jeden) zmiennej index równoważne do operacji index = index + 1
- len(image_list) – funkcja zwracająca liczbę elementów przechowywanych w liście image_list
- if index == len(image_list): – instrukcja warunkowa, porównująca wartość zmiennej index, z liczbą elementów przechowywanych w liście image_list
- elif button_a.was_pressed(): – instrukcja warunkowa, która wykona się, jeżeli wcześniej żadna instrukcja z tego samego bloku nie została uruchomiona oraz wciśnięty był przycisk A

- index -= 1 – dekrementacja (zmniejszenie wartości o jeden) zmiennej index, równoważne do operacji index = index - 1

Lekcja 5

- pin0.write_digital(1) – ustawienie wartości 1 (+3V) na pinie 0 płytki Micro:bit
- pin0.write_digital(0) – ustawienie wartości 0 (0V) na pinie 0 płytki Micro:bit
- import music – zaimportowanie (dołączenie do programu) biblioteki music, udostępniającej funkcje związane z odtwarzaniem dźwięków
- music.pitch(440) – zagranie dźwięku o wysokości tonu 440
- music.stop() – zaprzestanie gry dźwięku
- import random – zaimportowanie (dołączenie do programu) biblioteki random, która udostępnia funkcje pozwalające między innymi na np. otrzymanie pseudolosowych wartości
- random.randint(1, 10) – zwraca pseudolosową liczbę całkowitą z zakresu 1 do 10
- display.scroll(beats_per_minute, delay=80, wait=False, loop=True) – wyświetlanie tekstu zawartego w zmiennej beats_per_minute w trybie powtarzającym się (loop=True), parametr delay ustawia szybkość wyświetlania tekstu na 80. Funkcja jest w trybie nieblokującym (wait=False), co oznacza, że kolejne instrukcje są wykonywane i program nie zatrzymuje się na niej.

Lekcja 6

- accelerometer.current_gesture() – funkcja zwracająca aktualny gest w postaci ciągu znaków, odczytywany przez akcelerometr na płytce Micro:bit
- if gesture == "right": – sprawdzenie za pomocą instrukcji warunkowej, czy gest przechowywany w zmiennej gesture to "right"
- Gesty rozpoznawane przez akcelerometr na płytce Micro:bit to:
    - "up"
    - "down"
    - "left"
    - "right"
    - "face up"
    - "face down"
    - "freefall"

- "3g"
- "6g"
- "8g"
- "shake"

Lekcja 7

- import radio – zaimportowanie (dołączenie do programu) biblioteki radio, która udostępnia funkcje pozwalające na wysyłanie oraz odbieranie sygnałów radiowych
- radio.config(channel=1) – ustawienie radia na kanał 1
- radio.on() – włączenie radia
- radio.send("Hello!") – wysłanie danych poprzez radio w postaci ciągu znaków
- text = radio.receive() – odebranie danych i zapisanie ich do zmiennej text
- radio.off() – wyłączenie radia (zwalnia dodatkową pamięć oraz zmniejsza zużycie energii)
- repr(images_list[index]) – zakodowanie obrazu do postaci, która umożliwia jego przesłanie przez radio
- eval(image) – odkodowanie obrazu do postaci, która umożliwia jego wyświetlenie

Lekcja 8

- for signal in code: – pętla przyjmująca kolejne znaki z ciągu znaków znajdujących się w zmiennej code
- music.BLUES – melodia w stylu bluesa przechowywana w bibliotece music
- W bibliotece music mamy dostęp do różnych predefiniowanych melodii, ich spis możemy znaleźć na stronie: https://microbit-micropython.readthedocs.io/en/latest/music.html#built-in-melodies
- music.play(songs[song_number], wait=False, loop=True) – rozpoczęcie gry melodii z listy songs z indeksu song_number, w trybie powtarzającym się (loop=True). Funkcja jest w trybie nieblokującym (wait=False) co oznacza, że kolejne instrukcje są wykonywane.