

## Warsztaty 4 godz.

**Mów mi Python!** – czyli **programowanie w języku Python** w ramach projektu "Koduj z klasą" organizowanego przez Centrum Edukacji Obywatelskiej. Szczegóły pod adresem: <http://www.ceo.org.pl/pl/koduj>.

### Dla kogo, czyli co musi wiedzieć uczestnik

Dla każdego nauczyciela i ucznia, co oznacza, że materiał zawiera moduły o różnym stopniu trudności. Scenariusze zajęć oraz zakres przykładów można dostosować do poziomu uczestników.

### Cele, treści i metody

Cele projektu, spis wszystkich materiałów oraz zalecane metody ich realizacji dostępne są w dokumencie [Cele, materiały i metody](#). Umieszczono tam również listę oprogramowania wymaganego do realizacji wszystkich materiałów. **Podstawą szkoleń jest [wersja HTML](#)**. Wersje źródłowe dostępne są w repozytorium [Python101](#).

### Materiał zajęć

#### PODSTAWY PYTHONA

**Czas realizacji:** 1 \* 45 min.

**Metody:** kodowanie programu w edytorze od podstaw, wprowadzanie elementów języka w konsoli interpretera, ćwiczenia samodzielne w zależności od poziomu grupy.

**Programy, materiały i środki:** Python 2.7.x, edytor kodu, terminal, zalecany system Linux live *LxPupTahr*, wersja HTML scenariusza **Mały Lotek**, **punkty 1.2.1 – 1.2.5**, kod pełnego programu oraz ewentualne wersje pośrednie. Projektor, dostęp do internetu nie jest konieczny.

**Realizacja:** Na początku zapoznajemy użytkowników ze środowiskiem i narzędziami, tj. menedżer plików, edytor i jego konfiguracja, terminal znakowy, konsola Pythona, uruchamianie skryptu w terminalu, uruchamianie z edytora.

Omawiamy założenia aplikacji *Mały lotek*: losowanie pojedynczej liczby i próba jej odgadnięcia przez użytkownika. Następnie rozpoczynamy wspólne kodowanie wg materiału.

Po ukończeniu pierwszej części można urządzić mini-konkurs: zgadnij wylosowaną liczbę.

Budując program **można** reżyserować podstawowe błędy składniowe i logiczne, aby uczestnicy nauczyli się je dostrzegać i usuwać. Np.: próba użycia liczby pobranej od użytkownika bez przekształcenia jej na typ całkowity, niewłaściwe wcięcia, brak inkrementacji zmiennej iteracyjnej (nieskończona pętla), itp. Uczymy dobrych praktyk programowania: przejrzystość kodu (odstępny) i komentarze.

## WYKRESY W PYTHONIE

**Czas realizacji:** 1 \* 45 min.<sup>1</sup>

**Metody:** ćwiczenia w konsoli Pythona, wspólnie tworzenie i rozwijanie skryptów generujących wykresy, ćwiczenie samodzielne

**Programy, materiały i środki:** Python 2.7.x, biblioteka *Matplotlib*, edytor kodu, terminal, zalecany system Linux live *LxPupTahr*, wersja HTML scenariusza **Python kreśli**. Projektor, dostęp do internetu nie jest konieczny.

**Realizacja:** Zaczynamy od prostego przykładu w konsoli Pythona, z której cały czas korzystamy. Stopniowo kodujemy przykłady wykorzystując je do praktycznego (!) wprowadzenia wyrażień *listowych* zastępujących pętle *for*. Pokazujemy również mechanizmy związane z indeksowaniem list, m. in. notację wycinkową (ang. *slice*). Nie ma potrzeby ani czasu na dokładne wyjaśnienia tych technik. Celem ich użycia jest zaprezentowanie jednej z zalet Pythona: zwięzłości. Jeżeli wystarczy czasu, zachęcamy do samodzielnego sporządzenia wykresu funkcji kwadratowej.

## GRA ROBOTÓW (*Robot Game*)

**Czas realizacji:** 2 \* 45 min.

**Metody:** omówienie zasad gry, pokaz rozgrywki między przykładowymi robotami, kodowanie klasy robota z wykorzystaniem "klocków" (gotowego kodu), uruchamianie kolejnych walk.

**Programy, materiały i środki:** Python 2.7.x, biblioteka *rgkit*, przykładowe roboty z repozytorium *robotgame-bots* oraz skrypt *rgsimulator*, edytor kodu, terminal, zalecany system Linux live *LxPupTahr*, wersja HTML scenariusza **Gra robotów**, końcowy kod przykładowego robota w wersji A i B, koniecznie (!) kody wersji pośrednich. Projektor, dostęp do internetu lub scenariusz offline w wersji HTML dla każdego uczestnika.

**Realizacja:** Na początku omawiamy [przygotowanie środowiska testowego](#), czyli użycie *virtualenv*, instalację biblioteki *rgkit*, *rgbots* i *rgsimulator*, polecenie *rgrun*. Uwaga: jeżeli korzystać będziemy z systemu *LxPupTahr*, w katalogu *~/robot* mamy kompletne wirtualne środowisko pracy.

Podstawą jest zrozumienie reguł. Po wyjaśnieniu najważniejszych zasad gry, konstruujemy robota podstawowego w oparciu o materiał [Klocki 1](#). Kolejne implementowane zasady działania robota sprawdzamy w symulatorze, ucząc jednocześnie jego wykorzystania. W symulatorze reżyserujemy również przykładowe układy, wyjaśniając szczegółowe zasady rozgrywki. Później uruchomiamy "prawdziwe" walki, w tym z robotami open source (np. *stupid26.py*).

Dalej rozwijamy strategię działania robota w oparciu o funkcje – [Klocki 2A](#) i/lub zbiory – [Klocki 2B](#). W zależności od poziomu grupy można przećwiczyć: tylko wersję A, A następnie B, A i B równolegle z porównywaniem kodu. Uwaga: nie mamy czasu na wgłębianie się w szczegóły implementacji, takie np. jak wyrażenia zbiorów lub funkcje lambda.

---

1 Czas przeznaczony na tę część można zmniejszyć do 30 min., aby więcej czasu poświęcić grze robotów.

Wprowadzając kolejne zasady, wyjaśniamy odwołania do api biblioteki *rg* w dodawanych "klockach". Kolejne wersje robota zapisujemy w osobnych plikach, aby można je było konfrontować ze sobą.

Zachęcamy uczestników do analizy kodu i zachowań robotów: co nam dało wprowadzenie danej zasady? jak można zmienić kolejność ich stosowania w kodzie? jak zachowują się roboty open source? jak można ulepszyć działanie robota?