

Warsztaty 8 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 nauczycieli, którzy brali udział w pierwszej edycji programu „Koduj z klasą”.

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

TOTO LOTEK

Czas realizacji: 2 * 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 ***Duży Lotek***, punkty 1.2.6 – 1.2.14, kod pełnego programu oraz ewentualne wersje pośrednie. Projektor, dostęp do internetu nie jest konieczny.

Realizacja:

Jako punkt wyjścia prosimy każdego o skopiowanie i uruchomienie *Małego Lotka*. Przypominamy podstawy programowania w Pythonie (zmienna, pobieranie i wyprowadzanie danych, instrukcja warunkowa). Następnie omawiamy założenia aplikacji *Duży lotek*: losowanie i zgadywanie wielu liczb i rozpoczynamy wspólne kodowanie wg materiału. W zależności od poziomu grupy dbamy o mniej lub bardziej samodzielne wykonywanie przewidzianych w materiale ćwiczeń.

Po ukończeniu można urządzić mini-konkurs, np. zgadnij 5 wylosowanych z 20 liczb.

Budując program **można** reżyserować błędy składniowe i logiczne, aby uczestnicy uczyli 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 (odstępy) i komentarze.

WYKRESY W PYTHONIE

Czas realizacji: 1 * 45 min.

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*). Wyjaśniamy i ćwiczymy w interpreterze charakterystyczne dla Pythona konstrukcje. Jeżeli wystarczy czasu, zachęcamy do samodzielnego sporządzenia wykresu funkcji kwadratowej bądź innej.

GRA ROBOTÓW (*Robot Game*)

Czas realizacji: 3 * 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. W grupach zaawansowanych warto pokazać klocki z zestawu B i omówić działanie wyrażień *zbiorów* i funkcji *lambda*.

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?

Bazy danych w Pythonie

Czas realizacji: 2*45 min.

Metody: równoległe kodowanie dwóch skryptów w edytorze, uruchamianie i testowanie wersji pośrednich, ćwiczenia z użyciem *interpretera Sqlite*.

Programy, materiały i środki: Python 2.7.x, biblioteka [sqlite3 DB-API](#) oraz framework [Peewee](#), edytor kodu, terminal, zalecany system Linux live *LxPupTahr*, wersja HTML scenariusza **SQL v. ORM** oraz **Interpreter Sqlite**, kody pełnych wersji obu skryptów. Projektor, dostęp do internetu lub scenariusz offline w wersji HTML dla każdego uczestnika.

Realizacja: Na początku pokazujemy przydatność poznawanych zagadnień: wszechobecność baz danych w projektowaniu aplikacji desktopowych i internetowych (tu odesłanie do materiałów prezentujących Flask i Django); obsługa bazy i podstawy języka SQL to treści nauczania informatyki w szkole ponadgimnazjalnej; zadania maturalne wymagają umiejętności projektowania i obsługi baz danych.

Na podstawie materiału równoległe budujemy oba skrypty metodą kopiuj-wklej. Wyjaśniamy podstawy składni SQL-a, z drugiej eksponując założenia i korzystanie z systemów ORM. Pokazujemy, jak ORM-y skracają i usprawniają wykonywanie operacji CRUD oraz wpisują się w paradygmat projektowania obiektowego. Uwaga: ORM-y nie zastępują znajomości SQL-a, zwłaszcza w zastosowaniach profesjonalnych, mają również swoje wady, np. narzuty w wydajności.

Interpreter Sqlite wykorzystujemy do pokazania struktury utworzonych tabel (polecenia *.table*, *.schema*), później można (warto) przećwiczyć w nim polecenia CRUD w SQL-u.