Mów mi Python! – warsztaty 4 godz.

– 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.

Po co, czyli cele

Celem projektu jest zachęcanie nauczycieli i uczniów do programowania z wykorzystaniem języka Python. Przygotowane materiały prezentują zarówno zalety języka, jak i podstawowe pojęcia związane z tworzeniem programów i algorytmiką.

Ogólnym celem projektu jest propagowanie myślenia komputacyjnego, natomiast praktycznym rezultatem szkoleń ma być wyposażenie uczestników w minimum wiedzy i umiejętności umożliwiających samodzielne kodowanie w Pythonie.

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.

Oprogramowanie

1. **Interpreter *Pythona*** w wersji **2.7.x**.[1]
2. **System operacyjny:**
   a) ***Linux*** w wersji *live USB* (polecamy LxPupTahr) lub *desktop*, np. (X)Ubuntu lub Debian. Python jest domyślnym składnikiem systemu.
   b) lub ***Windows** 7 / 8 / 10*. Interpreter Pythona należy doinstalować.
3. **Edytor kodu**, np. *Geany*, *PyCharm*, *Sublime Text*, *Atom* (działają w obu systemach).
4. **Narzędzia dodatkowe**: *pip*, *virtualenv*, *git*.
5. **Biblioteki i frameworki Pythona** wykorzystywane w przykładach: *Matplotlib*, *Pygame*, *Peewee*, *SQLAlchemy*, *Flask*, *Django*, *Rgkit*, *RobotGame-bots*, *Rgsimulator*.

Uwaga: w ramach projektu przygotowano specjalną dystrybucję systemu Linux *LxPupTahr* przeznaczoną do instalacji na kluczach USB w trybie *live*. System zawiera wszystkie wymagane narzędzia i biblioteki Pythona, umożliwia realizację wszystkich scenariuszy oraz zapis plików tworzonych przez uczestników szkoleń.

Co, czyli treści

Spis wszystkich materiałów przygotowanych w ramach projektu dostępny jest w dokumencie: .

Wersje źródłowe: https://github.com/koduj-z-klasa/python101. Wersja HTML: http://python101.readthedocs.org lub http://python101.rtfd.org.

1 Wykorzystanie wersji 3.x interpretera również jest możliwe, ale wymaga poprawiania kodu.

(rb)

Jak, a więc metody

Cechy języka Python przedstawiane są na przykładach, których realizacja może przyjąć różne formy w zależności od dostępnego czasu. Zasada ogólna jest prosta: im więcej mamy czasu, tym więcej metod aktywizujących (kodowanie, testowanie, ćwiczenia, konsola Pythona, konsola Django itp.); im mniej, tym więcej metod podających (pokaz, wyjaśnienia najważniejszych fragmentów kodu, kopiuj-wklej). W niektórych materiałach (np. Robot Game, gry w Pygame) po skopiowaniu i wklejeniu kodu warto stosować zasadę uruchom-zmodyfikuj-uruchom.

1. Prezentacja, czyli uruchamianie gotowych przykładów wraz z omówieniem najważniejszych fragmentów kodu.
2. Wspólne budowanie programów od podstaw: kodowanie w edytorze, wklejanie bardziej skomplikowanych fragmentów kodu.
3. Ćwiczenia w interpreterze Pythona – niezbędne m. in. podczas wyjaśnianiu elementów języka oraz konstrukcji wykorzystywanych w przykładach.
4. Ćwiczenia i zadania wykonywane samodzielnie przez uczestników.

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 po zakomentowaniu instrukcji drukujących losowane liczby można urządzić mini-konkurs, najlepiej z nagrodami :-)

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 (odstępy) i komentarze.

Wykresy w Pythonie

Czas realizacji: 1 * 45 min.[2]

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 *wyrażeń 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.

2 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?

(rb)