Scenariusz zajęć

Progr@muj w zespole

Scenariusz zajęć

Rozwijamy umiejętność współpracy w zespole w trakcie zadania programistycznego.

Obszar tematyczny: informatyka

Etap edukacyjny

Zajęcia zaplanowano dla uczniów klas II – III liceów ogólnokształcących, uczęszczających do klas z rozszerzonym programem z zakresu informatyki, a także dla uczniów klas II – IV techników, uczących się w zawodzie technik – informatyk lub technik programista. Nie ma również przeciwwskazań, abyście zastosowali scenariusz w odniesieniu do uczniów klas I szkół średnich pod warunkiem, że będą oni posiadali niezbędną wiedzę i umiejętności z zakresu programowania, minimalnie:

- podstawowa wiedza o konstrukcjach programistycznych
- znajomość jakiegokolwiek języka programowania tekstowego (C; Pascal; LOGO; BASIC; JavaScript; itp.) lub znajomość Python na poziomie podstawowym

Dodatkowo przydatna będzie znajomość języka angielskiego na poziomie średnim (konieczność czytania dokumentacji technicznej w języku angielskim, choć można korzystać z serwisów tłumaczących).

Informacje organizacyjne

Grupa uczestników (klasa) podzielona zostaje na grupy 4 – osobowe (ostatnia grupa między 2 a 5 os.) na podstawie przeprowadzonego na wcześniejszych zajęciach (lekcja "0") prostego testu predyspozycji i ról w zespole.

Poziom zaawansowania uczniów w zakresie znajomości języka Python: podstawowy. Zakładamy w naszych samouczkach podstawowy poziom znajomości zagadnień związanych z programowaniem w ww. języku, ale młodzież może do niego dojść samodzielnie, bazując na ogólnodostępnych kursach, np.:

- Kurs: https://it-szkola.edu.pl/kkurs,kurs,216 lub
- Kurs: https://it-szkola.edu.pl/kkurs,kurs,217 lub
- https://python.szkola.pl/szkolenia-python-edu/python101x-edu/

Treści nauczania

Tematyka i program zajęć odwołują się do podstawy programowej z rozszerzonego zakresu informatyki:

a) Programowanie i rozwiązywanie problemów z wykorzystaniem komputera oraz innych urządzeń cyfrowych: układanie i programowanie algorytmów, organizowanie, wyszukiwanie i udostępnianie informacji, posługiwanie się aplikacjami komputerowymi (cel kształcenia nr II);

b) Rozwijanie kompetencji społecznych, takich jak: komunikacja i współpraca w zespole, w tym w środowiskach wirtualnych, udział w projektach zespołowych oraz zarządzanie projektami (cel kształcenia nr IV).

Cele edukacyjne dla uczniów (efekty uczenia się)

Dzięki uczestnictwu w zajęciach w zakresie kompetencji programistycznych uczeń:

Wiedza

a. zna podstawowe zasady efektywnej komunikacji interpersonalnej;

b. wie jakie czynniki determinują sukces w pracy zespołu;

c. umie wykorzystać ogólnodostępne API;

d. wie jak posługiwać się środowiskiem programistycznym PyCharm;

e. potrafi odróżnić różne elementy języka programowania.

Umiejętności

a) potrafi sformułować klarowny komunikat i sprawdzić czy został on właściwie zrozumiany;

b) radzi sobie z sytuacją konfliktową w zespole;

c) potrafi korzystać z serwisu GitHub do współpracy z innymi programistami;

d) radzi sobie z systemem operacyjnym Linux;

e) realizuje obiektowe paradygmaty programowania.

Postawy / kompetencje społeczne

a) zachowuje się odpowiedzialnie w stosunku do pozostałych członków zespołu i wywiązuje się z własnych zobowiązań, mając na celu osiągniecie wspólnego celu zespołu;

b) reaguje ciekawością na odmienne opinie i poglądy, rozwiązując konflikty w trybie porozumienia;

c) traktuje inne osoby z szacunkiem dla ich odmienności;

d) komunikuje się w sposób klarowny, wykorzystując narzędzia efektywnej komunikacji interpersonalnej;

e) zyskuje świadomość swoich własnych zasobów: umiejętności, talentów i mocnych stron.

Czas trwania (liczba godzin): 7 h lekcyjnych (zajęcia szkolne) + 5 h zegarowych (praca po zajęciach)

- 1 x 45 zajęcia wprowadzające, wyjaśnienie zasad, jakie będą obowiązywać w trakcie zajęć oraz ról pełnionych przez nauczyciela i uczniów, zawarcie kontraktu z uczniami, pre-test kompetencji współpracy i komunikacji w zespole oraz test predyspozycji i ról w zespole.
- 5 x 45 minut zajęć stricte programistycznych
- 5 x 60 minut pracy samodzielnej uczniów (w ramach stworzonego zespołu - tu ilość czasu zależy od predyspozycji członków zespołu)
- 1 x 45 minut spotkania podsumowującego, końcowy test kompetencji współpracy i komunikacji w zespole

Metody i formy pracy

- metoda podawcza (wprowadzenie + filmy wspomagające)
- metoda projektu (praca w zespołach w czasie lekcji i pozalekcyjna)
- lekcja odwrócona (znaczna część materiału będzie przyswajana w domu, zaś praca nad projektem zespołowym w dużej mierze ma odbywać się podczas zajęć szkolnych)

Środki dydaktyczne

- filmiki (pigułki wiedzy)
- krótki przewodnik po zajęciach dla uczniów

- przykładowe kody źródłowe na platformie GitHub
- instrukcje dot. poszczególnych lekcji wspomagające nauczycieli
- przewodnik po kompetencjach współpracy i komunikacji w zespole dla nauczycieli
- testy wspomagające realizację zajęć (p. przebieg zajęć).

Wymagania techniczne

Do prowadzenia zajęć niezbędne są:

- *komputery z procesorem minimum Core 3, 4 GB RAM dla uczestnika zajęć; **nie można** przeprowadzić zajęć z wykorzystaniem tabletów/smartfonów*
- *system operacyjny: Linux lub MacOS lub MS – Windows*
- *zainstalowane oprogramowanie PyCharm Community oraz Python w wersji min. 3.8, LibreOffice, GIMP, Bluefish (wszystkie to Open Source dostępne na wszystkie systemy operacyjne)*
- *połączenie z siecią Internet łączem min. 100 MBit (w praktyce – dowolnym połączeniem zapewniającym płynne odtwarzanie wideo dla celów pokazu)*
- *projektor/monitor oraz nagłośnienie niezbędne do odtwarzania wybranych odcinków wideotutorialu przez nauczyciela w trakcie zajęć*

W celu unifikacji i ułatwienia pracy przygotowaliśmy kompletny system operacyjny Linux Ubuntu 20.04 LTS w wersji maszyny wirtualnej OVA dla środowiska Virtualbox. Więcej informacji znajduje się w instrukcjach dla nauczycieli.

Nasz projekt oparliśmy o programowanie w języku Python 3.8 z wykorzystaniem dodatkowych bibliotek (PySimpleGUI, requests), które są ogólnodostępne. Stworzona aplikacja będzie możliwa do uruchomienia tylko na komputerze z zainstalowanym minimum Python 3.6 . Dodatkowo wykorzystujemy też inne aplikacje, wszystkie na otwartych licencjach:

- edytor tekstów Libre Office Writer (https://libreoffice.org)
- edytor grafiki GIMP (https://gimp.org)
- edytor HTML Bluefish (https://bluefish.openoffice.nl)

Przebieg zajęć

Zaproponowany przez nas podział materiału można rozpisać na następujące jednostki dydaktyczne:

0. Lekcja w szkole - wprowadzenie, zasady pracy, testy wstępne,

1. Lekcja w szkole - ustawienia wstępne środowiska

* Praca samodzielna - maszyna wirtualna i podstawy aplikacji

2. Lekcja w szkole - podstawowe typy danych i konstrukcje programistyczne w Python

* Praca samodzielna - obsługa głównych elementów biblioteki PySimpleGUI

3. Lekcja w szkole - requests i API - słowniki i JSON

* Praca samodzielna - samodzielne testy dostępu do API

4. Lekcja w szkole - różne interfejsy aplikacji, `Commit/Push` do repozytorium

* Praca samodzielna - definiowanie funkcji w Python i dalsze przygotowywanie dokumentacji

5. Lekcja w szkole - praca z kluczami i wartościami słowników

* Praca samodzielna - końcowe tworzenie dokumentacji

6. Lekcja w szkole - podsumowanie projektów, wybór najlepszego projektu, post-testy itp.

Lekcja nr 0 (45 min.)

Lekcja ma na celu wprowadzenie uczniów w kontekst zajęć, ich cele, planowany przebieg, a także przekazanie źródeł, na podstawie których uczniowie powinni przygotować się do zajęć (przewodnik dla uczniów, materiały wideo), przeprowadzenie dwóch testów: prostego testu predyspozycji i ról w zespole służącego przygotowaniu podziału na zespoły (5 minut) oraz testu umiejętności współpracy i komunikacji w zespole, który stanowił będzie punkt odniesienia dla oceny nabycia kompetencji społecznych w wyniku przeprowadzonych zajęć (10-15 minut). Nauczyciel i uczniowie ustalają zasady pracy podczas zajęć, tzw. kontrakt (więcej informacji dot. zasad pracy, podziału, kontraktu w *Przewodniku dla nauczycieli*). Nauczyciel może również zaproponować, że zwycięskie/wszystkie zespoły, które ukończyły i od-

dały działające aplikacje wraz pozostałymi materiałami (strona internetowa etc.) otrzymają oceny celujące. Zawsze proponujemy tylko nagrodę, aby motywować uczniów do podjęcia wyzwania.

Pomiędzy lekcją 0 i lekcją 1 nauczyciel przeprowadza podział na zespoły na podstawie przeprowadzonego wcześniej testu predyspozycji, swojej znajomości poziomu zaawansowania uczniów oraz wg parytetu płci. Informacja o podziale na zespoły jest przekazywana uczniom co najmniej na 1 dzień przed rozpoczęciem właściwych zajęć. Następnie uczniowie kontaktują się ze sobą i wybierają temat pracy swojego zespołu.

Zespoły pracują nad wybranym przez siebie zadaniem samodzielnie. Członkowie zespołu dzielą pracę między siebie w taki sposób, że każdy z nich ma do wykonania swoje zadania i jest za nie odpowiedzialny. Podczas pracy zespołowej nauczyciel jest jedynie wsparciem i, w razie potrzeby, mediatorem.

Warto zaznaczyć, że celem projektu jest wspólna praca nad projektem, poznanie języka programowania, narzędzi edycyjnych; filmy podzielone są na te do wyświetlania w szkole (gdzie ważne jest wsparcie nauczyciela) i do obejrzenia samodzielnie (aby wyrobić umiejętności samodzielnej nauki).

Lekcja nr 1 (45 min.)

Rozpoczyna się od ankiety wśród uczniów - jakie systemy operacyjne znają, czy wiedzą, na które z nich dostępny jest Python, Libre Office, GIMP, Geany, czy znają różne licencje, w tym OpenSource. Przede wszystkim należy skupić się na instalacji odpowiednich aplikacji, dla ujednolicenia interfejsu proponujemy skorzystać z rozwiązania OVA dla VirtualBox. Jeśli nie korzystamy z OVA to w systemie Windows pobieramy instalatory ze stron internetowych:

- https://www.python.org/ftp/python/3.8.10/python-3.8.10-amd64.exe,
- https://www.libreoffice.org/download/download/,
- https://www.gimp.org/downloads/,
- https://geany.org/download/releases/,

Po instalacji aplikacji uczniowie tworzą pierwszy projekt w środowisku PyCharm i w nim wykonują pierwszy minimalny program - okno aplikacji z komunikatem, np. “Hej - witamy w zespole!”. W ten sposób poznają środowisko pracy i od razu widzą efekt swojej pracy. Przykładowy kod źródłowy znajduje się w repozytorium GitHub (https://github.com/abixadamj/Popoju-trze-Progr-mujemy) w katalogu spotkanie_1/00_start

Następnie tworzą swoje konta w serwisie GitHub i poznają różnego rodzaju licencje na materiały i oprogramowanie, zapoznają się też z podstawami pracy w edytorach wykorzystywanych w naszym projekcie.

W trakcie pracy samodzielnej uczniowie zapoznają się z filmami omawiającymi importowanie maszyny OVA i podstawy tworzenia dokumentacji, grafiki, stron internetowych oraz ustalają pomiędzy sobą szczegóły wykonania projektu (np. nazwa aplikacji, czy rodzaj informacji, z których chcą korzystać).

Lekcja nr 2 (45 min.)

Poświęcona jest omówieniu tych elementów i konstrukcji programistycznych, które będą niezbędne w dalszych etapach prac, a więc:

- podstawowych typów danych w Python i konwencji F-String;
- zaawansowanych typach danych: listach, słownikach;
- sposobach importowania modułów zewnętrznych;
- pętlach iteracyjnych;
- instrukcjach warunkowych.

Dodatkowo omawiamy:

- listy w edytorze tekstów
- warstwy w edytorze graficznym
- kontenery w edytorze HTML

W trakcie pracy samodzielnej uczniowie zapoznają się z filmami dotyczącymi zasad działania biblioteki PySimpleGUI oraz kolejnymi elementami dokumentów tekstowych, grafik czy stron WWW.:

- tworzenie layoutu aplikacji;
- sterowanie zdarzeniami;
- wyświetlaniem informacji;
- dodawaniem elementów sterujących (przycisków);
- dodawaniem obrazków;
- sposobami wprowadzania danych.

Na tym etapie mogą próbować tworzyć szkielet aplikacji, czyli tworzyć okna z elementami najważniejszymi z punktu widzenia funkcjonalności.

Osoby odpowiedzialne za dokumentację, grafikę i HTML mogą zacząć

tworzenie odpowiednich elementów, jak dokumentacja, strona WWW reklamowa, grafiki do projektu.

Lekcja nr 3 (45 min.)

Poświęcona jest na analizowanie systemu API, z którego zespoły wybierają elementy do swoich projektów. Uczniowie wykorzystują Python Console w PyCharm, sprawdzają dokumentację różnych przykładowych API. Uczą się wykorzystywać moduł requests do komunikacji z API. Dzięki temu otrzymują odpowiedzi w formacie JSON, które analizują wykorzystując pętle iteracyjne "for". Tworzą pliki PDF z dokumentacją czy PNG z własnymi grafikami. W tej lekcji niezbędne jest działające łącze internetowe.

W trakcie pracy samodzielnej uczniowie testują samodzielnie różne API, aby zapoznać się z rodzajami zwracanych informacji, testują błędy, jakie mogą wystąpić podczas używania API. Tworzą własne repozytorium w serwisie GitHub. Od kolejnej lekcji mogą swoją pracę wysyłać do serwisu GitHub.

Lekcja nr 4 (45 min.)

W trakcie tego spotkania poznajemy sposoby definiowania funkcji i przestrzenie w Python (ang. `namespace`); zapoznamy się ze skryptem odczytującym i prezentującym dane z przykładowego API. W drugiej części pracujemy ze zdalnym repozytorium GitHub.

- definiowanie funkcji
- koncepcja przestrzeni nazw w Python
- praca ze zdalnym repozytorium GitHub

W trakcie pracy samodzielnej uczniowie sprawdzają dane odczytywane z API za pomocą skryptu wysyłają też projekt lokalnie zapisany do repozytorium GitHub.

Lekcja nr 5 (45 min.)

Przeznaczona jest na utrwalenie wiedzy o słownikach, modyfikacji ich elementów. Możemy też wykorzystać ten czas dla uczniów na ostatnie zapytania. W trakcie pracy samodzielnej uczniowie kończą tworzenie aplikacji, dokumentacji i strony internetowej Tu nie przewidujemy już żadnych filmów. Dla zespołów najsłabszych przewidzieliśmy przykładowy kod aplikacji, przykładowy dokument tekstowy dokumentacji i stronę internetową opisującą aplikację.

Lekcja nr 6 (45 min.)

Przeznaczona jest na podsumowanie projektów; zespoły prezentują wyniki swojej pracy, mogą też wzajemnie komentować swoje aplikacje. Podczas tej lekcji przeprowadzamy też ponownie test umiejętności współpracy i komunikacji w zespole. Przy komentowaniu i sprawdzaniu prac możemy posłużyć się formatterem kodu Black (https://black.readthedocs.io/en/stable/) i sprawdzić, ile zmian wykona.

Sposób oceny osiągnięcia celów

- nauczyciel na zakończenie ocenia, czy dany zespół uczniowski zrealizował swój projekt, tj. osiągnął zamierzony cel, przygotowana prosta aplikacja działa poprawnie (zgodnie z intencją zespołu), a kod jest poprawny i dobrej jakości
- druga składowa oceny dotyczy poprawy umiejętności współpracy i komunikacji w zespole w stosunku do wartości bazowej (pre-test) - weryfikacja następuje z wykorzystaniem narzędzia udostępnionego nauczycielowi w ramach naszego modelu (test umiejętności współpracy i komunikacji w zespole) - wynik testu poprawiony o min. 30% w stosunku do wartości bazowej dzięki udziałowi w zajęciach uznaje się za w pełni satysfakcjonujący.

Oceny stopnia osiągnięcia celów nie należy mylić z ocenami szkolnymi - przed lekcją 0 nauczyciel powinien zdecydować, czy za wykonane zadania będą przyznawane oceny (rekomendujemy motywowanie przez nagrodę, np. ocenę celującą za ukończenie projektu, tj. osiągnięcie opisanego powyżej celu zajęć w dwóch wymiarach).

Materiały pomocnicze:

1) Materiały wspierające dla uczniów i nauczycieli z zakresu programowania:

- Kurs: https://it-szkola.edu.pl/kkurs,kurs,216 wraz z kodami w serwisie GitHub: https://github.com/klubmlodegoprogramisty/python/tree/main/poziom_podstawowy
- Kurs: https://it-szkola.edu.pl/kkurs,kurs,217 wraz z kodami w serwisie GitHub: https://github.com/klubmlodegoprogramisty/python/tree/main/poziom_sredniozaawansowany
- Kurs: https://python.szkola.pl/szkolenia-python-edu/python101x-edu/
- Dokument: https://kodujwpythonie.pl/Dodatek_A_GIT_A4_.pdf

Książki:

- A. Jurkiewicz, Python 3. Projekty dla początkujących i pasjonatów. Wyd. Helion, Warszawa 2022,
- Konrad Jagaciak, PYTHON, kurs programowania na prostych przykładach, ringier Axel Springer, Warszawa 2019,
- Sean McManus, Misja Python, Utwórz swoją kosmiczną grę! PWN, Warszawa 2019,
- Marek Luliński & Gniewomir Sarbicki, Python, C++, JavaScript. Zadania z programowania.Wyd.Helion, Warszawa 2018,
- Michał Wiszniewski, Python na start! Programowanie dla nastolatków. Wyd. Helion, Warszawa 2017,
- A. Jurkiewicz, K. Zadrzyński, K. Wasilkowska, M. Trojanowicz, Koduj w Pythonie. Tworzymy grę przygodową. Wyd. Fundacja Rozwoju Edukacji Cyfrowej, Gdańsk 2020.

2) Materiały wspierające dla uczniów i nauczycieli w zakresie komunikacji i współpracy w zespole:

- J. Kołodziejczyk, K. Salamon-Bobińska, N. Karaszewski, S. Bobula, Nauczanie kooperatywne w: Edukacja jako odpowiedź. Praca zbiorowa pod red G. Mazurkiewicz, Warszawa 2014;
- M. Spławska- Murmyło, A. Wawryszuk, Współpraca zespołowa z wykorzystaniem TIK; Materiały ORE:http:bc.ore.edu.pl. pobór 5/1/2022. Warszawa 2017;

Książki:

- M. Adams, Myślenie pytaniami. Wyd. Studio EMKA, Warszawa 2020,
- S. Stahl, Jak się dogadywać mimo różnic. Instrukcja obsługi typów osobowości. Wyd. OTWARTE, Kraków 2021,
- R. Gut, A. Gut, Pokolenie Y. Zarządzanie sobą na Zielonej Ścieżce. Wyd. Instytut Flashpoint, Wrocław 2016,
- J. Lamri, Kompetencje XXI wieku. Wyd. Wolters Kluwer, Warszawa 2021,
- T. Kozłowski, Zanim będzie za późno. O potrzebie rozwoju kompetencji społecznych w edukacji. Wys. Wolters Kluwer, Warszawa 2020,
- James M. Heidema. Carol A. MCKenzie, Budowanie zespołu z pasją. Dom Wydawniczy Rebis, Poznań 2006,
- M. Płócińska, H. Rylke, Czas współpracy i czas zmian. Wyd. WSiP, Warszawa 2002,
- E. Góralczyk, Umowa z klasą. Wyd. Fraszka Edukacyjna, Warszawa 2015

Autorzy scenariusza:

1) Adam Jurkiewicz
2) Rafał Kamiński
3) Konrad Kosieradzki
4) Elżbieta Piotrowska-Gromniak