UNIWERSYTET EKONOMICZNY W KATOWICACH

KIERUNEK INFORMATYKA

**ANNA PISZING**

**147698 (nr albumu)**

Projekt gry edukacyjnej wspomagającej uczenie się

**języka Python**

**Educational video game project about computer programming in Python**

Praca licencjacka

napisana pod kierunkiem

prof. dr hab. Jerzego Gołuchowskiego

KATOWICE 2025

**OŚWIADCZENIE PROMOTORA**

Oświadczam, że niniejsza praca została przygotowana pod moim kierunkiem i stwierdzam, że spełnia wymogi stawiane pracom dyplomowym.

Pracę akceptuję

…………………………………… ……………………………………

(data) (podpis promotora)

...................................................... Katowice, dnia...............................

Imię i nazwisko

......................................................

Kierunek

......................................................

Nr albumu

**OŚWIADCZENIE**

Mając świadomość odpowiedzialności prawnej oświadczam, że złożona praca licencjacka/inżynierska pt.: …………….......................................................................................

…………………………………………………………………………………………………..………..

została napisana przeze mnie samodzielnie.

Równocześnie oświadczam, że praca ta nie narusza praw autorskich w rozumieniu ustawy z dnia 4 lutego 1994 roku o prawie autorskim i prawach pokrewnych oraz dóbr osobistych chronionych prawem.

Ponadto praca nie zawiera informacji i danych uzyskanych w sposób niedozwolony i nie była wcześniej przedmiotem innych procedur związanych z uzyskaniem dyplomów lub tytułów zawodowych uczelni wyższej.

Wyrażam zgodę na nieodpłatne udostępnienie mojej pracy w celu oceny jej oryginalności przez Jednolity System Antyplagiatowy prowadzony przez ministra właściwego ds. szkolnictwa wyższego oraz przechowywania jej w Ogólnopolskim Repozytorium Prac Dyplomowych oraz wewnętrznej bazie prac dyplomowych Uniwersytetu Ekonomicznego w Katowicach. Oświadczam, że poinformowano mnie o zasadach dotyczących oceny oryginalności pracy dyplomowej przez Jednolity System Antyplagiatowy.

Oświadczam także, że ostateczna wersja pracy przesłana przeze mnie drogą elektroniczną jest zgodna z plikiem poddanym ocenie w Jednolitym Systemie Antyplagiatowym.

Jednocześnie oświadczam, że jest mi znany przepis art. 233 § 1 Kodeksu karnego określający odpowiedzialność za składanie fałszywych zeznań.

......................................................

(podpis składającego oświadczenie)

**Spis treści**

[Wstęp 7](#_Toc211550761)

[Rozdział 1. Gry komputerowe i grywalizacja w procesie uczenia się 9](#_Toc211550762)

[1.1. Gry komputerowe i grywalizacja 9](#_Toc211550763)

[1.2. Wybrane gry komputerowe w kształceniu programowania 10](#_Toc211550764)

[1.3. Użyteczność grywalizacji w realizowaniu zadań edukacyjnych 12](#_Toc211550765)

[1.4 Studium wykonalności (Feasibility study) gry wspomagającej uczenie się programowania w języku Python 13](#_Toc211550766)

[1.5. Analiza zapotrzebowania 14](#_Toc211550767)

[1.6. Ocena czynników ekonomicznych 14](#_Toc211550768)

[Rozdział 2. Projekt gry wspomagającej uczenie się języka Python 14](#_Toc211550769)

[2.1 Założenia projektu gry 14](#_Toc211550770)

[2.2. Elementy grywalizacji 16](#_Toc211550771)

[**2.2.1. System punktacji** 16](#_Toc211550772)

[**2.2.2. Poziomy trudności i progresja** 16](#_Toc211550773)

[**2.2.3. System osiągnięć i trofeów** 17](#_Toc211550774)

[**2.2.4. Tablica wyników i rywalizacja** 17](#_Toc211550775)

[**2.2.5. Podpowiedzi i informacja zwrotna** 17](#_Toc211550776)

[**2.2.6. Interfejs użytkownika** 17](#_Toc211550777)

[**2.2.7. Elastyczność i rozwój treści** 18](#_Toc211550778)

[2.3. Struktura gry 18](#_Toc211550779)

[**2.3.1. Etap inicjalizacji (Initialization)** 19](#_Toc211550780)

[**2.3.2. Główna pętla programu (Main Loop)** 20](#_Toc211550781)

[**2.3.3. Menu główne i nawigacja (Menu and Navigation)** 20](#_Toc211550782)

[**2.3.4. Obsługa użytkownika i informacja zwrotna (User Feedback and Input)** 20](#_Toc211550783)

[**2.3.5. Przepływ wyzwań (Challenge Flow)** 21](#_Toc211550784)

[**2.3.6. Przepływ lekcji (Lesson Flow)** 21](#_Toc211550785)

[**2.3.7. Zapis stanu gry (Game State Saving)** 21](#_Toc211550786)

[**2.3.8. Zakończenie wykonywania gry** 21](#_Toc211550787)

[2.4. Materiały dydaktyczne wspierające uczenie się języka Python 22](#_Toc211550788)

[2.5. Projekt szkolenia zawartość: teksty i materiały szkolące 23](#_Toc211550789)

[Rozdział 3 Eksperymentalna implementacja gry w środowisku Pygame 25](#_Toc211550790)

[3.1. Inicjalizacja środowiska 26](#_Toc211550791)

[**3.1.1. Inicjalizacja środowiska i konfiguracja** 26](#_Toc211550792)

[**3.1.2. Struktura danych dydaktycznych** 26](#_Toc211550793)

[3.2. Interfejs graficzny 27](#_Toc211550794)

[**3.2.1. Menu główne** 27](#_Toc211550795)

[**3.2.2. Ekran wyników (High Scores)** 30](#_Toc211550796)

[**3.2.3. Ekran lekcji (Learn Page)** 30](#_Toc211550797)

[**3.2.4. Ekran zadania kodowego (Code Challenge)** 31](#_Toc211550798)

[**3.2.5. Ekran zadań interaktywnych** 32](#_Toc211550799)

[**3.2.6. Ekran wprowadzania imienia (Name Input)** 33](#_Toc211550800)

[**3.2.7. Podsumowanie sekcji graficznej** 33](#_Toc211550801)

[3.3. Logika gry i ocena odpowiedzi 34](#_Toc211550802)

[**3.3.1. Sprawdzanie kodu użytkownika** 34](#_Toc211550803)

[**3.3.2. Zadania interaktywne** 34](#_Toc211550804)

[**3.3.3. System punktacji** 34](#_Toc211550805)

[**3.3.4. Zapis postępu i wyników** 34](#_Toc211550806)

[**3.3.5. Główna pętla programu** 34](#_Toc211550807)

[**3.3.6. Mechanizmy pomocnicze** 35](#_Toc211550808)

[Zakończenie 35](#_Toc211550809)

[Literatura 36](#_Toc211550810)

[**Pozycje zwarte i artykuły naukowe** 36](#_Toc211550811)

[**Netografia** 37](#_Toc211550812)

[Spis fragmentów kodu 38](#_Toc211550813)

[Spis rysunków 38](#_Toc211550814)

# Wstęp

Współczesne technologie cyfrowe w coraz większym stopniu przenikają do edukacji, oferując nowe możliwości nauki poprzez interaktywne narzędzia. Jednym z kluczowych problemów edukacyjnych jest efektywne nauczanie podstaw programowania i logiki, szczególnie wśród dzieci i młodzieży. Tradycyjne metody dydaktyczne nie zawsze skutecznie angażują uczniów i motywują ich do nauki. W odpowiedzi na to wyzwanie, coraz większą popularność zyskują gry edukacyjne, które łączą naukę z zabawą, ułatwiając przyswajanie skomplikowanych zagadnień w przystępny sposób.

Problemem badawczym niniejszej pracy jest wykorzystanie edukacyjnej gry komputerowej do nauki podstaw programowania i logiki. W kontekście rosnącego zapotrzebowania na kompetencje cyfrowe i umiejętności programistyczne, taka gra może stanowić innowacyjne wsparcie w procesie edukacyjnym. Aktualność problematyki wynika zarówno z dynamicznego rozwoju technologii, jak i konieczności dostosowania metod nauczania do współczesnych realiów.

Głównym celem pracy jest stworzenie prostej gry komputerowej w Pythonie przy użyciu biblioteki Pygame, która będzie wspierać naukę podstaw programowania w sposób angażujący i przyjazny dla dzieci oraz młodzieży.

Cele szczegółowe obejmują:

* analizę istniejących gier edukacyjnych i metod nauki programowania,
* zaprojektowanie i implementację mechanik gry sprzyjających nauce logicznego myślenia,
* ocenę skuteczności gry jako narzędzia dydaktycznego.

W ramach pracy sformułowano następujące pytania badawcze:

* W jaki sposób wykorzystanie gier komputerowych może wspierać naukę podstaw programowania i logiki?
* Jakie mechaniki rozgrywki najlepiej wspomagają przyswajanie wiedzy z zakresu programowania przez dzieci i młodzież?
* Jakie narzędzia w silniku Pygame są najbardziej efektywne w tworzeniu gier edukacyjnych?

Zakres pracy obejmuje analizę teoretycznych podstaw edukacyjnych gier komputerowych oraz projekt i implementację prototypu gry w Pygame. Przedmiotem badań jest wykorzystanie interaktywnej gry komputerowej do nauki podstaw programowania i logiki. Zakres czasowy obejmuje analizę aktualnych trendów w edukacji cyfrowej oraz prace projektowe i implementacyjne przeprowadzone w okresie pisania pracy. Zakres przestrzenny odnosi się do rynku gier edukacyjnych oraz ich potencjalnych zastosowań w placówkach edukacyjnych i samodzielnej nauce.

W pracy zastosowano metody badawcze obejmujące: analizę literatury dotyczącej gier edukacyjnych i metod nauczania programowania, projektowanie i implementację prototypu gry w Pygame, testy i ewaluację skuteczności gry w kontekście edukacyjnym.

Podczas pracy korzystano z literatury dotyczącej teorii gier edukacyjnych, metod nauki programowania, a także dokumentacji technicznej Pythona i Pygame. Wykorzystano również artykuły naukowe oraz publikacje branżowe opisujące aktualne trendy w edukacji cyfrowej. Takie podejście umożliwiło stworzenie solidnych podstaw teoretycznych dla realizowanego projektu. Literatura obejmuje wyłącznie te pozycje, które były przywołane w artykule. Spis literatury powinien być uporządkowany alfabetycznie (zaczynając od nazwiska autora, potem inicjał imienia), a w odniesieniu do aktów normatywnych tej samej rangi chronologicznie. Spis jest numerowany. Spis literatury zawiera wykaz wykorzystanej bibliografii i netografii. Dodatkowo w bibliografii należy wyróżnić publikacje zwarte, artykuły naukowe, raporty, akty normatywne. Netografia obejmuje wykorzystane i opisane źródła internetowe ze wskazaniem daty dostępu.

Praca dostarcza wartościowych wniosków dotyczących wykorzystania gier komputerowych w edukacji oraz możliwości ich implementacji w praktyce.

# Rozdział 1. Gry komputerowe i grywalizacja w procesie uczenia się

## Gry komputerowe i grywalizacja

Gry komputerowe stanowią jedno z najważniejszych zjawisk współczesnej kultury cyfrowej. Początkowo traktowano je wyłącznie jako formę rozrywki. Z czasem zaczęto dostrzegać ich potencjał edukacyjny, społeczny i kulturowy. Współcześnie gry pełnią różnorodne funkcje i stają się narzędziem oddziaływania na jednostkę oraz społeczeństwo.

Zasadniczym elementem gier komputerowych jest interaktywność. Użytkownik nie tylko odbiera treść, ale także aktywnie uczestniczy w procesie rozgrywki. Interakcja umożliwia kształtowanie przebiegu zdarzeń oraz rozwijanie indywidualnych strategii działania (Juul, 2005). To odróżnia gry od tradycyjnych form przekazu medialnego.

W literaturze przedmiotu wskazuje się, że gry stanowią system reguł. Gracz podejmuje decyzje w ramach określonych zasad, które wyznaczają możliwe działania (Salen i Zimmerman, 2004). Reguły te nadają rozgrywce strukturę i pozwalają na osiąganie wyznaczonych celów. Bez jasno określonych zasad gra traci sens i staje się chaotyczna.

Nie bez znaczenia pozostaje również aspekt narracyjny. Gry często tworzą światy fikcyjne, w których gracz odgrywa określoną rolę (Juul, 2005). Narracja sprzyja zaangażowaniu emocjonalnemu oraz identyfikacji z bohaterem. Dzięki temu rozgrywka zyskuje dodatkową głębię i staje się bardziej atrakcyjna.

Podkreśla się także znaczenie gier jako medium społecznego. Współczesne gry komputerowe coraz częściej umożliwiają rozgrywkę wieloosobową (Filiciak, 2006). Gracze współpracują lub rywalizują w środowiskach sieciowych. Powstają w ten sposób społeczności graczy, które wykraczają poza ramy samej rozrywki.

Analizując istotę gier komputerowych, należy zauważyć ich rosnące zastosowanie poza obszarem czystej zabawy. Gry wspierają proces dydaktyczny, rozwijają kompetencje poznawcze i społeczne, a także służą treningowi zawodowemu. Stanowią więc narzędzie, które łączy rozrywkę z nauką i praktyką życia codziennego.

Grywalizacja, czyli zastosowanie mechanizmów znanych z gier w innych dziedzinach, znajduje szerokie zastosowanie w edukacji. Pojęcie to odnosi się do wykorzystania mechanizmów znanych z gier w środowiskach niezwiązanych bezpośrednio z rozrywką. Dzięki temu proces uczenia staje się bardziej atrakcyjny, a uczestnicy zyskują dodatkową motywację do działania (Zichermann i Cunningham, 2012).

Grywalizacja w edukacji programistycznej może wspierać rozwój umiejętności, które wymagają systematyczności i wytrwałości. Podstawowymi elementami grywalizacji są punkty, poziomy, rankingi oraz nagrody. Wykorzystanie takich rozwiązań pozwala na zwiększenie zaangażowania użytkowników i utrzymanie ich uwagi przez dłuższy czas. (Deterding. i in. (2011)

Zastosowanie mechanizmów nagradzania sprawia, że uczestnicy częściej podejmują próby rozwiązania zadania i szybciej reagują na popełnione błędy. W konsekwencji proces nauki staje się dynamiczny i bardziej efektywny.

Istotnym aspektem grywalizacji jest rywalizacja między uczestnikami. Porównywanie własnych wyników z osiągnięciami innych osób może stanowić silne źródło motywacji (Werbach i Hunter, 2012). Należy jednak pamiętać, że nie wszyscy gracze reagują w ten sam sposób na rywalizację. W związku z tym rekomenduje się, aby system grywalizacyjny uwzględniał zarówno współzawodnictwo, jak i możliwość samodzielnego rozwoju.

Grywalizacja pełni również funkcję wspierającą w budowaniu poczucia postępu czego. Widoczny rozwój umiejętności zachęca do dalszego wysiłku i pozwala na utrzymanie wysokiego poziomu satysfakcji z nauki (Kapp, 2012). Kolejne odblokowywane poziomy, misje lub osiągnięcia pełnią rolę wyraźnych sygnałów informujących o pokonywaniu trudności. Dzięki temu uczestnicy procesu edukacyjnego mają świadomość wartości swojego zaangażowania.

Zastosowanie grywalizacji w edukacji programistycznej, szczególnie w nauce języka Python, stwarza dodatkowe możliwości. Wprowadzenie systemu punktów za rozwiązanie zadań, nagród za odkrycie nowych funkcji języka czy rankingów może pozytywnie wpłynąć na postępy w nauce (Domínguez i in. 2013). Dobrze zaprojektowany system grywalizacji nie tylko wspiera zdobywanie wiedzy, ale także kształtuje umiejętności miękkie, takie jak wytrwałość, współpraca czy kreatywne rozwiązywanie problemów.

## 1.2. Wybrane gry komputerowe w kształceniu programowania

Gry edukacyjne stanowią narzędzie wspierające proces uczenia się poprzez zabawę. Łączą elementy rozrywki z celami dydaktycznymi, co pozwala na przyswajanie wiedzy w angażujący sposób. W przypadku nauki programowania mogą pełnić rolę środka motywującego do podejmowania regularnych ćwiczeń i rozwiązywania problemów.

W literaturze podkreśla się, że gry edukacyjne mogą być wykorzystywane na różnych etapach kształcenia. Ich zadaniem jest nie tylko wspomaganie rozwoju umiejętności technicznych, lecz także wspieranie myślenia logicznego i analitycznego. Dzięki temu uczą rozwiązywania zadań krok po kroku i rozwijają umiejętności krytycznego myślenia.

Gry programistyczne przybierają różne formy, od prostych aplikacji przeglądarkowych po rozbudowane symulatory i platformy edukacyjne. Często wykorzystują zadania polegające na sterowaniu postacią za pomocą kodu, rozwiązywaniu łamigłówek logicznych lub tworzeniu prostych programów. Rozwiązania tego typu rozwijają umiejętność praktycznego stosowania składni języka i uczą przewidywania działania kodu.

Ważnym aspektem gier uczących programowania jest interaktywność. Umożliwia ona natychmiastowe otrzymywanie informacji zwrotnej o poprawności rozwiązania. Dzięki temu użytkownik uczy się na bieżąco, może analizować błędy i wprowadzać poprawki. Takie podejście wspiera proces uczenia się przez eksperymentowanie.

Gry edukacyjne uczące programowania często zawierają elementy rywalizacji i współpracy. Rywalizacja motywuje do podejmowania kolejnych prób oraz poprawiania wyników. Z kolei współpraca sprzyja wymianie doświadczeń i rozwijaniu umiejętności pracy zespołowej, które są istotne w praktyce programistycznej.

Istotne jest także stopniowe zwiększanie poziomu trudności. Gry tego typu zazwyczaj rozpoczynają się od prostych zadań, aby stopniowo wprowadzać bardziej złożone problemy. Takie rozwiązanie pozwala na przyswajanie wiedzy krok po kroku i dostosowanie tempa nauki do możliwości gracza.

W badaniach nad skutecznością gier edukacyjnych wskazuje się, że ich wykorzystanie w nauce programowania może zwiększać motywację, zaangażowanie oraz trwałość nabytej wiedzy. Istotne znaczenie ma również atrakcyjna forma wizualna i narracyjna, które wzmacniają zainteresowanie odbiorcy.

Wśród znanych gier edukacyjnych uczących programowania często wymienia się CodeCombat. Gra ta umożliwia sterowanie postacią za pomocą rzeczywistego kodu pisanego w językach programowania, w tym w języku Python (CodeCombat, 2025). Wykorzystano w niej mechanikę gry fabularnej, gdzie gracz rozwija bohatera, walczy z przeciwnikami i wykonuje zadania. Dzięki temu nauka kodowania łączy się z rozbudowaną narracją oraz elementami gry przygodowej. Rozwiązanie to może stanowić inspirację przy projektowaniu gry, w której kod staje się narzędziem interakcji z wirtualnym światem.

Kolejnym przykładem jest LightBot. W tej grze gracz kieruje robotem, który wykonuje określone czynności na planszy. Rozgrywka polega na układaniu sekwencji komend, które robot musi poprawnie wykonać. Gra koncentruje się na nauce podstawowych pojęć programistycznych, takich jak pętle, funkcje i warunki. Prostota zasad oraz przejrzysta forma wizualna stanowią wartościowy wzorzec dla tworzenia gry, która stopniowo wprowadza kolejne elementy programowania bez nadmiernego obciążenia gracza.

Inspirującym rozwiązaniem jest również Scratch. Choć nie jest to klasyczna gra, lecz środowisko programistyczne, często wykorzystuje się je w formie edukacyjnej zabawy. Użytkownik buduje programy z klocków symbolizujących instrukcje, a efekt jest natychmiast widoczny na ekranie. Scratch kładzie nacisk na twórczość, interaktywność oraz naukę przez eksperymentowanie. W projekcie gry można wykorzystać podobną ideę natychmiastowej informacji zwrotnej i przyjaznego wizualnego środowiska.

Za inspirację mogą posłużyć także gry takie jak Human Resource Machine (). Gra ta prezentuje zadania przypominające pracę w biurze, które należy rozwiązać poprzez tworzenie prostych algorytmów. Dzięki temu nauka programowania zostaje powiązana z rozwiązywaniem problemów logicznych osadzonych w humorystycznym kontekście. Podejście to pokazuje, jak ważna może być atrakcyjna narracja, która ułatwia przyswajanie złożonych zagadnień.

Dokonując analizy powyższych przykładów, można wskazać kilka cech szczególnie istotnych w kontekście projektowania własnej gry edukacyjnej. Należy zwrócić uwagę na stopniowanie trudności, atrakcyjną formę wizualną, zastosowanie narracji oraz natychmiastową informację zwrotną. Ważne jest również wprowadzenie elementów zabawy i wyzwań, które podtrzymują motywację gracza. Własny projekt może czerpać inspirację z tych rozwiązań, dostosowując je do specyfiki języka Python i przyjętych założeń dydaktycznych.

## 1.3. Użyteczność grywalizacji w realizowaniu zadań edukacyjnych

W celu zwiększenia zaangażowania użytkowników oraz utrzymania ich motywacji na wysokim poziomie w projektowanej grze edukacyjnej zastosowano mechanizmy grywalizacji. Grywalizacja polega na wykorzystaniu elementów znanych z gier w kontekstach niezwiązanych z rozrywką, w tym w edukacji (Zhan, 2022). Dzięki temu proces nauki staje się bardziej atrakcyjny, a użytkownicy chętniej podejmują kolejne wyzwania.

Istotnym elementem stało się także wprowadzenie rywalizacji i współpracy między graczami. Użytkownicy mogą porównywać swoje wyniki z innymi uczestnikami za pośrednictwem tablicy liderów. Ponadto przewidziano możliwość wprowadzenia trybu zespołowego, w którym gracze mogliby wspólnie rozwiązywać zadania i dzielić się postępami. Tego rodzaju rozwiązanie zwiększa zaangażowanie oraz sprzyja wymianie wiedzy i doświadczeń (Tech & Learning.).

Na etapie projektowania gry przyjęto założenie, że mechanizmy grywalizacji powinny wspierać proces nauki, a nie odciągać od celów edukacyjnych. Zadbano o to, by każdy element rozgrywki był powiązany z nauką programowania w języku Python, a nie stanowił jedynie atrakcyjnego dodatku. W literaturze wskazuje się, że nieumiejętne zastosowanie grywalizacji może prowadzić do efektów negatywnych, takich jak spadek motywacji lub poczucie nadmiernej rywalizacji (Almeida i in., 2023).

Zastosowanie grywalizacji w edukacyjnej grze programistycznej przyczynia się do poprawy skuteczności nauczania. Użytkownicy wykazują większe zaangażowanie, częściej powracają do aplikacji oraz osiągają lepsze wyniki w rozwiązywaniu zadań. Potwierdzają to badania nad efektywnością grywalizowanych kursów języka Python, które wskazują na pozytywny wpływ punktów, poziomów i osiągnięć na proces przyswajania wiedzy (Zhan, 2022). W rezultacie nauka programowania staje się bardziej efektywna i satysfakcjonująca.

## 1.4 Studium wykonalności (Feasibility study) gry wspomagającej uczenie się programowania w języku Python

Na etapie planowania projektu dokonano analizy wykonalności przedsięwzięcia. Studium wykonalności miało na celu określenie czy przygotowanie gry edukacyjnej jest możliwe z perspektywy technicznej, organizacyjnej i dydaktycznej. Ustalono główne ograniczenia oraz potencjalne możliwości rozwoju projektu.

W pierwszej kolejności przeanalizowano aspekty techniczne. Oceniono, że możliwe jest wykorzystanie popularnych bibliotek wspierających interaktywne środowiska programistyczne, co zapewnia stabilność działania.

Następnie rozważono aspekty finansowe i organizacyjne. Wskazano, że projekt nie wymaga zakupu specjalistycznych licencji ani komercyjnych narzędzi. Zastosowano technologie opensource, co znacząco obniża koszty realizacji. Stwierdzono, że prototyp może zostać opracowany przy ograniczonych zasobach finansowych i kadrowych.

Analizie poddano także kwestie dydaktyczne. Zbadano, czy zaproponowana forma gry wspiera proces uczenia się programowania w języku Python. Ustalono, że interaktywne środowisko z natychmiastową informacją zwrotną umożliwia szybsze utrwalanie wiedzy. Podkreślono, że stopniowe zwiększanie poziomu trudności sprzyja długotrwałej motywacji.

Przeprowadzono również analizę ryzyka. Zidentyfikowano zagrożenia związane z możliwymi trudnościami technicznymi oraz ograniczeniami czasowymi. Uznano jednak, że dzięki modularnej strukturze gry ryzyko można ograniczyć poprzez wprowadzanie zmian w kolejnych iteracjach projektu.

Na podstawie przeprowadzonych analiz stwierdzono, że realizacja projektu jest możliwa i uzasadniona. Gra edukacyjna wspomagająca naukę programowania w języku Python charakteryzuje się niskimi kosztami wdrożenia, wysoką dostępnością oraz potencjałem dydaktycznym. Studium wykonalności potwierdziło zasadność kontynuacji prac nad prototypem.

## 1.5. Analiza zapotrzebowania

Przeprowadzono również analizę zapotrzebowania na tego typu narzędzie edukacyjne. W wyniku przeglądu literatury oraz dostępnych produktów stwierdzono, że istnieje zapotrzebowanie na proste, angażujące gry edukacyjne wspierające naukę programowania. Istniejące rozwiązania często są zbyt zaawansowane lub niedostosowane do początkujących użytkowników czego przykładem jest gra komputerowa JOY OF PROGRAMMING - Software Engineering Simulator, która wymagają podstawową znajomość programowania. Przyjęto zatem, że opracowanie gry opartej na jasnych zasadach, prostym interfejsie oraz dostosowanej do poziomu początkujących użytkowników może wypełnić lukę na rynku.

## 1.6. Ocena czynników ekonomicznych

Ocenie poddano także czynniki ekonomiczne. Ze względu na wykorzystanie bezpłatnych narzędzi i brak konieczności zakupu licencji lub sprzętu, koszty realizacji projektu uznano za pomijalne. Jedynym wymaganym zasobem jest czas, który został odpowiednio zaplanowany w harmonogramie realizacji pracy licencjackiej.

Na podstawie przeprowadzonej analizy stwierdzono, że projekt gry edukacyjnej uczącej programowania w języku Python jest wykonalny. Istnieją odpowiednie zasoby techniczne, organizacyjne i merytoryczne pozwalające na jego realizację w ramach pracy dyplomowej.

# Rozdział 2. Projekt gry wspomagającej uczenie się języka Python

## 2.1 Założenia projektu gry

Na etapie projektowania prototypu gry edukacyjnej przyjęto kilka kluczowych założeń. Ich celem było zapewnienie spójności koncepcji oraz dostosowanie rozwiązania do potrzeb odbiorców. Wskazano elementy techniczne, dydaktyczne i estetyczne, które uznano za niezbędne dla realizacji założonych celów.

Podstawowym założeniem było stworzenie narzędzia wspierającego naukę programowania w języku Python. W tym celu przewidziano wprowadzenie mechanizmów umożliwiających stopniowe zwiększanie poziomu trudności zadań. Dzięki temu użytkownik ma możliwość rozwijania umiejętności krok po kroku.

Celem projektowanego szkolenia z języka Python jest opracowanie podstaw merytorycznych i dydaktycznych, które posłużą jako fundament dla prototypu gry edukacyjnej. Założono, że gra ma wspierać naukę programowania na poziomie początkującym. Szczególny nacisk położono na rozwijanie umiejętności logicznego myślenia, rozwiązywania problemów oraz stosowania składni i konstrukcji języka Python w praktyce.

W kontekście nauki programowania w języku Python założono, że wdrożenie odpowiednich elementów grywalizacji może znacząco zwiększyć motywację i zaangażowanie graczy.

W kontekście gry uczącej języka Python wprowadzono system punktów i poziomów trudności, który nagradza użytkowników za poprawne rozwiązanie zadań (Swacha, 2023). Zdobyte punkty sumują się i umożliwiają awans na kolejne poziomy, co wspiera poczucie postępu i wzmacnia motywację do dalszej nauki. Badania wskazują, że tego rodzaju mechanizmy pozytywnie wpływają na wyniki edukacyjne oraz zwiększają wytrwałość uczących się (Garcia i Revano, 2021).

Dodatkowo zastosowano system osiągnięć, które przyznawane są za wykonanie określonych działań, takich jak ukończenie zestawu zadań bez błędów lub rozwiązanie trudniejszych zadań w krótkim czasie. Mechanizm ten pełni funkcję nagrody i zwiększa satysfakcję z nauki, jednocześnie podtrzymując zainteresowanie użytkownika nawet przy bardziej wymagających zadaniach (Swacha, 2023). Dzięki zastosowaniu osiągnięć nauka staje się bardziej urozmaicona i zyskuje element rywalizacji z samym sobą.

Przyjęto, że Dodatkową formą motywacji mogą być trofea, które nagradzają gracza za osiągnięcia, takie jak rozwiązanie określonej liczby zadań czy osiągnięcie perfekcyjnego wyniku. Ważnym aspektem grywalizacji jest stopniowe zwiększanie poziomu trudności. W edukacyjnej grze do nauki Pythona kolejne poziomy powinny wprowadzać coraz bardziej zaawansowane koncepcje, takie jak pętle, instrukcje warunkowe czy funkcje.

Kolejnym założeniem było zastosowanie mechaniki gry, która motywuje do systematycznej nauki. Uwzględniono elementy rywalizacji, system nagród oraz czytelny sposób prezentacji postępów. Przyjęto, że mechanizmy te zwiększą zaangażowanie i utrwalą zdobywaną wiedzę.

Gra jest wielopoziomowa. Przejście do kolejnego poziomu wymagałoby poprawnego rozwiązania wszystkich lub większości wyzwań. W ten sposób użytkownik systematycznie zdobywałby nowe umiejętności programistyczne. Aby uniknąć frustracji u początkujących graczy, warto wdrożyć system podpowiedzi. W przypadku trudniejszych zadań użytkownik mógłby skorzystać z podpowiedzi w formie wskazówki dotyczącej rozwiązania problemu lub skróconego wyjaśnienia teoretycznego.

Przykład prostego zadania edukacyjnego przedstawia fragment kodu:

|  |
| --- |
| "type": "code",  "description": "Challenge 1:\n Create a variable x with value 5.",  "test": lambda env: get\_case\_insensitive(env, "x") == 5,  "hint": "Try: x = 5" |

Fragment kodu 1 Przykładowe zadanie z odpowiedzią i podpowiedzią

Na etapie projektowania przyjęto również założenie, że gra powinna być uniwersalna pod względem środowiska uruchomieniowego. Takie rozwiązanie zwiększa dostępność narzędzia i ułatwia jego dystrybucję.

Ważnym założeniem było połączenie teorii z praktyką. Postanowiono, że użytkownik będzie miał możliwość natychmiastowego sprawdzenia efektów napisanych fragmentów kodu. Dzięki temu proces uczenia się zostaje uatrakcyjniony i zyskuje charakter interaktywny.

Przyjęto także założenie o elastyczności treści. Gra została zaprojektowana w taki sposób, aby można było dodawać nowe zadania, moduły i poziomy trudności. Umożliwia to dalszy rozwój projektu oraz dostosowanie do różnych grup odbiorców.

## 2.2. Elementy grywalizacji

Wdrożenie elementów grywalizacji w edukacyjnej grze do nauki języka Python znacząco zwiększa zaangażowanie użytkowników oraz skuteczność procesu nauki. System punktacji, poziomy trudności, osiągnięcia i tablica wyników stanowią kluczowe elementy, które sprawiają, że nauka programowania staje się bardziej angażująca i satysfakcjonująca.

### **2.2.1. System punktacji**

Jednym z podstawowych mechanizmów grywalizacji jest system punktacji. W projektowanej grze użytkownik otrzymuje punkty za poprawne rozwiązanie zadań programistycznych. Liczba zdobytych punktów zależy od stopnia trudności zadania oraz czasu potrzebnego na jego ukończenie.

Przykładowo, za szybkie i poprawne rozwiązanie użytkownik otrzymuje premię punktową, natomiast za wielokrotne próby, mniejszą liczbę punktów. Takie rozwiązanie zachęca do dokładności i systematyczności.

Punkty są zapisywane w profilu gracza i sumują się w ramach kolejnych etapów, co umożliwia śledzenie postępów i porównywanie wyników z innymi użytkownikami.

### **2.2.2. Poziomy trudności i progresja**

Gra została zaprojektowana jako wielopoziomowy system nauki. Każdy poziom (ang. level) wprowadza nowe pojęcia języka Python, od podstawowych konstrukcji, takich jak zmienne i listy, po bardziej zaawansowane, jak funkcje, pętle i instrukcje warunkowe.

Po ukończeniu zadań na danym poziomie użytkownik odblokowuje kolejne etapy:

* Beginner (początkujący) - zmienne, typy danych, podstawowe operatory,
* Intermediate (średniozaawansowany) - listy, pętle, instrukcje warunkowe,
* Advanced (zaawansowany) - funkcje, zbiory, słowniki, złożone problemy logiczne,
* Interactive (rozszerzony) - zadania z otwartą formą odpowiedzi i interaktywnym testowaniem kodu.

Każdy etap rozpoczyna się krótkim wprowadzeniem teoretycznym, po którym gracz wykonuje zestaw praktycznych ćwiczeń.

### **2.2.3. System osiągnięć i trofeów**

Kolejnym istotnym elementem grywalizacji jest system osiągnięć (achievements), który nagradza gracza za określone działania, np.:

* rozwiązanie wszystkich zadań bez błędów,
* ukończenie poziomu w krótkim czasie,
* napisanie szczególnie optymalnego rozwiązania.

Za zdobycie osiągnięć użytkownik otrzymuje trofea, wirtualne medale lub odznaki, które są widoczne w profilu. Takie rozwiązanie wzmacnia poczucie sukcesu, wspiera motywację wewnętrzną i promuje rywalizację z samym sobą.

### **2.2.4. Tablica wyników i rywalizacja**

Dodatkową formą motywacji jest tablica wyników (High Scores), na której prezentowane są najlepsze rezultaty użytkowników. Ranking można filtrować według poziomu trudności lub liczby ukończonych zadań.

Wyniki zapisywane są w lokalnym pliku gry, co pozwala graczowi obserwować własny postęp. Rozważane jest również wprowadzenie wersji sieciowej umożliwiającej porównywanie wyników z innymi użytkownikami.

### **2.2.5. Podpowiedzi i informacja zwrotna**

Aby uniknąć frustracji u początkujących graczy, gra oferuje system podpowiedzi. W przypadku błędu użytkownik otrzymuje krótką sugestię w stylu:

*„Sprawdź nazwę zmiennej — Python rozróżnia wielkość liter.”*

lub

*„Spróbuj użyć pętli for zamiast powtarzania kodu.”*

Natychmiastowa informacja zwrotna jest jednym z kluczowych czynników dydaktycznych. Program automatycznie sprawdza poprawność wprowadzonego kodu i przekazuje komunikat o sukcesie lub błędzie wraz z wyjaśnieniem. Dzięki temu gracz uczy się poprzez eksperymentowanie i samodzielne rozwiązywanie problemów.

### **2.2.6. Interfejs użytkownika**

Istotnym elementem projektu jest intuicyjny interfejs, zaprojektowany z myślą o osobach początkujących. Menu główne umożliwia szybki dostęp do:

* trybu nauki (Learn Python),
* wyboru poziomu trudności,
* przeglądu wyników (High Scores),
* resetowania postępu.

Układ graficzny został opracowany w oparciu o prostą kolorystykę, czytelne czcionki i logiczny podział na sekcje, co pozwala użytkownikowi skoncentrować się na nauce, a nie na obsłudze aplikacji.

### **2.2.7. Elastyczność i rozwój treści**

Gra została zaprojektowana w sposób modułowy, co umożliwia łatwe dodawanie nowych zadań, poziomów i tematów nauczania. Dzięki temu aplikacja może być rozwijana w przyszłości, np. o moduły dotyczące innych języków programowania lub zagadnień algorytmicznych.

Zastosowana struktura kodu i danych umożliwia modyfikowanie treści bez konieczności zmiany logiki działania programu, co znacząco ułatwia utrzymanie i rozbudowę projektu.

## 2.3. Struktura gry

Projektowana gra edukacyjna została zaprojektowana w formie modularnej, czyli że w łatwy sposób jest zarządzany każdy z oddzielnych składowych systemu oraz ich dalszy rozwój. Struktura aplikacji opiera się na zrównaniu na poziomie funkcjonalnym, które składa się z moduły odpowiedzialne na wypełnianie funkcji obsługi interfejsu użytkownika, zarządzania postępem, obsługi zdarzeń oraz realizacji zadań dydaktycznych.

Graficzny schemat przedstawiony na rysunku 2.1 przedstawia przepływ informacji pomiędzy podstawowymi składowymi gry oraz logiczne powiązania między modułami.

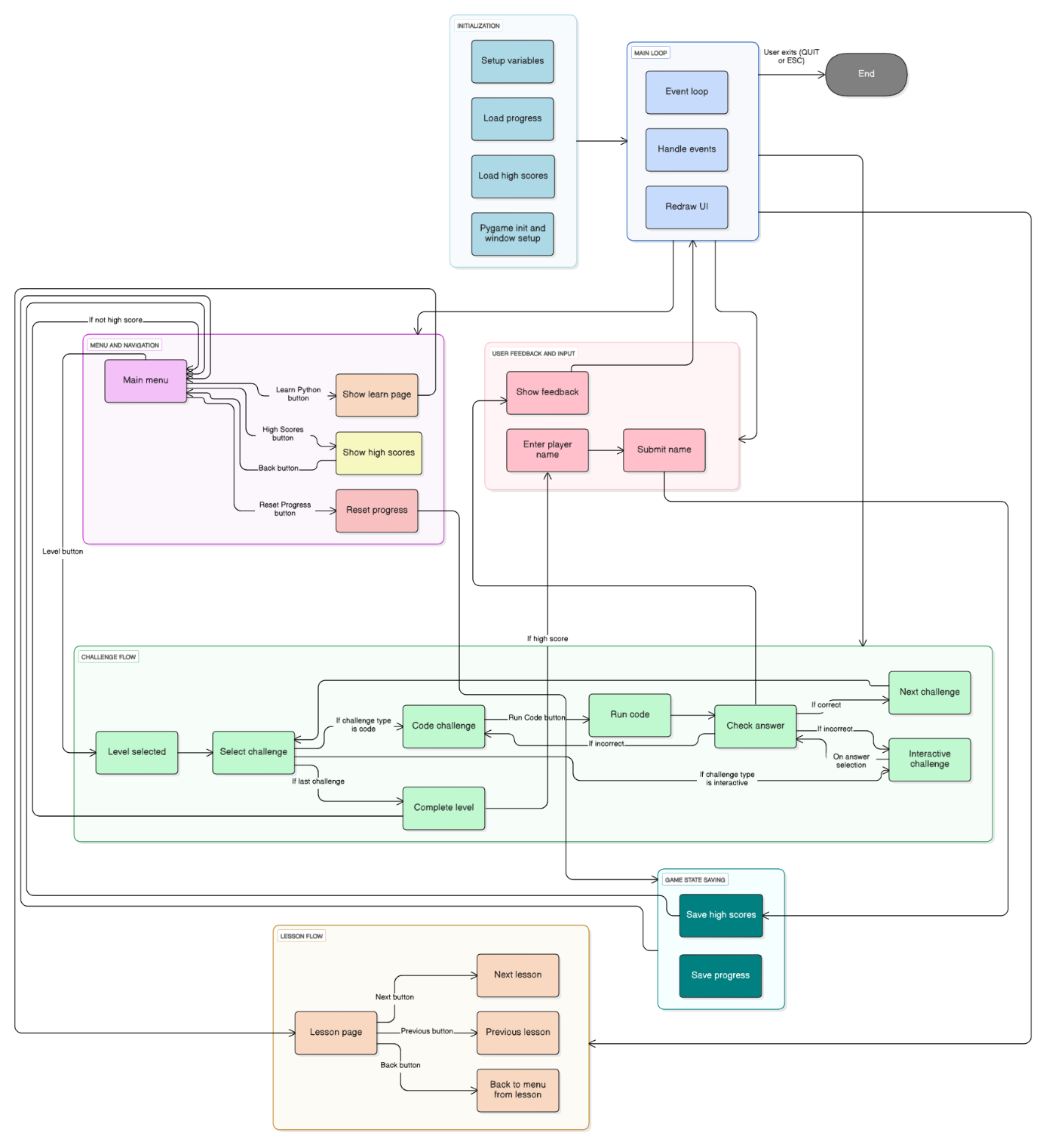


Diagram 1 Schemat blokowy struktury gry edukacyjnej wspomagającej naukę języka Python.

### **2.3.1. Etap inicjalizacji (Initialization)**

Pierwszym etapem działania programu jest inicjalizacja, obejmująca przygotowanie środowiska gry. W tym etapie następuje:

* wczytanie zmiennych globalnych oraz ustawień,
* odczyt postępu użytkownika z pliku,
* załadunek wyników z tablicy rekordów,
* inicjacja biblioteki Pygame i konfiguracja okna gry.

Ten proces pełni funkcję punktu wejścia do włączenia głównej pętli programu.

### **2.3.2. Główna pętla programu (Main Loop)**

Najważniejszym komponentem gry jest pętla główna, odpowiedzialna za stałą obsługę zdarzeń, reagowanie na działanie użytkownika oraz odświeżanie interfejsu. Z jej pomocą realizowane są trzy podstawowe funkcje:

* Obsługa zdarzeń (Event Loop) – reagowanie na kliknięcia, wprowadzanie kodu oraz wybór opcji w menu.
* Przetwarzanie logiki gry (Handle Events) – uruchamianie odpowiednich modułów (np. wyboru poziomu, sprawdzania wyników, zapisu postępu).
* Aktualizacja interfejsu (Redraw UI) – odświeżanie elementów graficznych i wyświetlanie aktualnych danych.

Użytkownik może wysłać grę w dowolnym momencie, co powoduje przejście do bloku End.

### **2.3.3. Menu główne i nawigacja (Menu and Navigation)**

Menu główne jest punktem centralnym interakcji z użytkownikiem. Składa się on z przycisków posiadających:

* rozpoczęcie nauki (Learn Python),
* wyświetlenie tablicy wyników (High Scores),
* resetowanie postępu (Reset Progress),
* powrót do poprzednich ekranów.

Moduł menu odpowiedzialny jest za prowadzenie użytkownika do wybranych sekcji, zapewniając spójność interfejsu i logikę nawigacji.

### **2.3.4. Obsługa użytkownika i informacja zwrotna (User Feedback and Input)**

Po rozwiązaniu problemu program emituje informację zwrotną (feedback) na temat prawidłowość wprowadzonego kodu. W przypadku poprawnego wyniku gracz może wprowadzić swoje imię i przesłać wynik na tablicę najlepszych wyników. W sytuacji błędu gry emituje komunikat i oferuje wskazówkę w postaci podpowiedzi.

Taki model interakcji realizuje zasadę nauki przez doświadczenie i bezpośrednią korektę błędów.

### **2.3.5. Przepływ wyzwań (Challenge Flow)**

Moduł Challenge Flow tworzy główną część dydaktyczną gry. Po dokonaniu przez użytkownika wyboru poziomu następuje:

* Wybór wyzwania (Select Challenge) – do zadań przyporządkowany jest pewien tematyczny zakres.
* Wykonanie kodu (Run Code) – zawodnik wpisuje rozwiązanie, które jest automatycznie uruchamiane.
* Sprawdzenie wyniku (Check Answer) – system porównuje wynik z poprawnym rozwiązaniem.
* Przejście dalej (Next Challenge / Complete Level) – w razie poprawnego rozwiązania zawodnik przechodzi do następnego zadania lub etapu.

W przypadku popełnionych błędów użytkownik otrzymuje instrukcje, w związku z czym można dalej uczyć się w sposób interaktywny.

### **2.3.6. Przepływ lekcji (Lesson Flow)**

W module Lesson Flow gracz może się kontaktować z teoretycznymi materiałami. Każda lekcja jest po krótkim wyjaśnieniu pojęć, z przykładami kodu. Gracz może:

* przesuwać się między lekcjami (Next / Previous Lesson),
* wracać do menu głównego (Back to Menu).

Takie zagęszczenie łączy praktykę z teorią, będąc intro do programistycznych zagadnień.

### **2.3.7. Zapis stanu gry (Game State Saving)**

Aby zapewnić ciągłość nauki, gra automatycznie zapisuje postęp użytkownika. Ta mechanismy obejmują:

* zapis liczby ukończonych poziomów,
* zapis całkowitej liczby punktów,
* updatę listy najlepszych wyników.

Te dane są czytaną podczas następnego uruchomienia oprogramowania, pozwalając użytkownikowi na kontynuację nauki od tego miejsca, w którym je rozwinął.

### **2.3.8. Zakończenie wykonywania gry**

Po zakończeniu grau lub wyjściu ze środowiska (np. poprzez naciśnięcie klawisza ESC) aplikacja przejdzie do bloku End, które zakończy pętlę główną i zwolni zasoby systemowe.

## 2.4. Materiały dydaktyczne wspierające uczenie się języka Python

W procesie tworzenia gry edukacyjnej uczącej programowania w języku Python dokonano analizy dostępnych materiałów dydaktycznych. Uwzględniono zarówno zasoby drukowane, jak i cyfrowe, które mogą wspierać graczy w procesie uczenia się. Przyjęto założenie, że dobór materiałów powinien być dostosowany do poziomu początkującego użytkownika, który nie posiada wcześniejszego doświadczenia z programowaniem.

Podstawową grupę materiałów stanowią podręczniki i książki poświęcone językowi Python. Publikacje tego typu przedstawiają podstawowe konstrukcje języka, omawiają składnię i prezentują przykłady programów (Dawson, 2020), (Matthes, 2023). Tego rodzaju źródła mogą pełnić funkcję uzupełniającą dla treści zawartych w grze, umożliwiając graczowi samodzielne pogłębianie wiedzy.

Ważną kategorię materiałów stanowią kursy internetowe oraz platformy elearningowe. Zawierają one lekcje w formie tekstowej, graficznej lub wideo, co pozwala na zróżnicowaną prezentację zagadnień. Wiele z nich oferuje ćwiczenia interaktywne, które umożliwiają sprawdzenie zdobytej wiedzy w praktyce (Python Academy), (Pystart). Tego rodzaju rozwiązania mogą wspierać graczy w utrwalaniu treści poznanych w grze.

Zastosowanie w grze mogą znaleźć także oficjalne dokumentacje języka Python. Dokumentacja zawiera szczegółowe opisy funkcji, bibliotek i modułów, a także przykłady użycia (Python Software Foundation). Choć tego typu materiał jest często zbyt obszerny i trudny dla początkujących, może stanowić cenne źródło informacji dla graczy bardziej zaawansowanych.

Nie bez znaczenia pozostają materiały w formie zadań i zestawów ćwiczeń. Zadania krok po kroku prowadzą do rozwiązania problemu programistycznego, rozwijając umiejętność logicznego myślenia. W grze edukacyjnej można wykorzystać podobne zadania jako element fabularny, np. w postaci misji do wykonania. Dzięki temu nauka staje się bardziej atrakcyjna i motywująca.

Szczególną rolę pełnią materiały wizualne i interaktywne, takie jak infografiki, diagramy czy quizy. Ułatwiają one przyswajanie trudnych zagadnień oraz wspierają zapamiętywanie kluczowych informacji. Włączenie tego typu materiałów do gry pozwala na połączenie treści edukacyjnych z elementami zabawy (Python 101).

Dokonano również analizy forów dyskusyjnych i społeczności internetowych skupionych wokół języka Python. Uczestnictwo w takich przestrzeniach umożliwia wymianę doświadczeń, zadawanie pytań i uzyskiwanie pomocy (Python.org). Gracz, który dzięki grze nabędzie podstawowe kompetencje, może korzystać z tego rodzaju wsparcia w dalszej nauce.

Podsumowując, dostępne materiały dydaktyczne oferują szerokie możliwości wykorzystania w procesie nauki programowania w języku Python. Ich właściwy dobór i integracja z treściami gry edukacyjnej mogą zwiększyć skuteczność procesu kształcenia. W ten sposób gra pełni rolę narzędzia inicjującego zainteresowanie, a dodatkowe źródła wspierają rozwój samodzielnych umiejętności programistycznych.

## 2.5. Projekt szkolenia zawartość: teksty i materiały szkolące

Przyjęto, że szkolenie będzie miało strukturę modułową. Każdy moduł odpowiada za naukę wybranego zagadnienia programistycznego, takiego jak zmienne, instrukcje warunkowe, pętle, funkcje czy słowniki. Moduły zostały zaprojektowane w sposób progresywny, co oznacza, że wiedza i umiejętności nabyte w jednym etapie stanowią podstawę do realizacji kolejnych. Dzięki temu możliwe jest stopniowe wprowadzanie gracza w świat programowania bez konieczności wcześniejszego przygotowania informatycznego.

W ramach szkolenia zaplanowano wykorzystanie interaktywnych zadań programistycznych, quizów sprawdzających wiedzę oraz krótkich materiałów wyjaśniających kluczowe pojęcia. Wyniki będą zapisywane i użytkownik będzie miał

możliwość powtórzenia poprzednich modułów. Uznano, że takie podejście sprzyja utrwaleniu wiedzy i zwiększa zaangażowanie użytkownika.

Do przygotowania treści merytorycznych wykorzystano dokumentację języka Python, ogólnodostępne podręczniki do nauki programowania np. Python kurs dla nauczycieli i studentów oraz doświadczenia dydaktyczne z kursów online, takich jak Codeacademy, Coders Lab czy W3Schools. Zawartość szkolenia dostosowano do możliwości technicznych planowanej gry oraz do poziomu trudności, który nie będzie zniechęcał początkujących użytkowników.

Do realizacji szkolenia wykorzystano szereg narzędzi dydaktycznych i technologicznych. Jako środowisko programistyczne wybrano edytor Visual Studio Code ze względu na jego dostępność, funkcjonalność oraz intuicyjny interfejs. W zakresie narzędzi programistycznych wstępnie przyjęto, że gra edukacyjna zostanie stworzona w środowisku Pygame. Umożliwia tworzenie rozbudowanych interfejsów użytkownika oraz wspiera integrację z zewnętrznymi bibliotekami.

Zbadano możliwość zastosowania języka Python w środowisku Unity do określonych zadań, który nie jest podstawowym językiem do tworzenia gier. Szukano również innego silnika, na którym może powstać gra m.in. Godot używającego własnego języka skryptowego o nazwie GDScript, który jest językiem programowania wysokiego poziomu bardzo podobnego do Pythona czy narzędzie Panda3D.

Szkolenie dostarcza gotowej struktury modułów, które mogą zostać odwzorowane jako poziomy w grze. Każdy poziom będzie odpowiadał jednemu zagadnieniu programistycznemu. Elementy szkoleniowe, takie jak quizy, instrukcje czy zadania kodowe, zostaną przekształcone w interaktywne komponenty gry. Dzięki temu możliwe będzie zachowanie spójności między treścią edukacyjną a mechaniką gry.

Zorganizowanie szkolenia w formie zrozumiałej, czytelnej i atrakcyjnej wizualnie przyczyni się do zwiększenia motywacji użytkowników. Wstępny projekt zakłada, że szkolenie będzie dostępne w języku polskim, co dodatkowo zwiększy jego przystępność dla uczniów szkół średnich oraz studentów. Uwzględniono również możliwość późniejszego rozszerzenia gry o nowe moduły np. o programowanie obiektowe.

Podsumowując, projekt szkolenia z języka Python stanowi fundament merytoryczny dla tworzonego prototypu gry edukacyjnej. Odpowiednio dobrana struktura szkolenia, zastosowane narzędzia oraz przemyślany sposób organizacji treści dydaktycznych pozwalają na efektywne połączenie nauki programowania z elementami rozrywki.

# Rozdział 3 Eksperymentalna implementacja gry w środowisku Pygame

Do implementacji gry edukacyjnej przyjęto bibliotekę Pygame, która umożliwia tworzenie aplikacji multimedialnych z wykorzystaniem języka Python. Zaletą tego rozwiązania jest dostępność wielu gotowych funkcji obsługujących grafikę dwuwymiarową, dźwięk oraz zdarzenia użytkownika. Wybór Pygame został uzasadniony prostotą integracji z językiem Python, co ułatwia tworzenie prototypów i późniejsze modyfikacje kodu.

Podstawowym założeniem implementacji było zbudowanie gry o modularnej strukturze kodu. Wydzielono osobne moduły odpowiedzialne za obsługę grafiki, logiki gry oraz interfejsu użytkownika. Takie rozwiązanie pozwala na łatwiejsze testowanie i rozwijanie poszczególnych elementów systemu. Zastosowano również jednolite standardy nazewnictwa oraz komentarzy w kodzie, co zwiększa jego czytelność.

Przyjęto, że gra powinna działać w rozdzielczości dostosowanej do większości ekranów komputerowych. W tym celu wykorzystano skalowanie obiektów graficznych, które umożliwia prawidłowe wyświetlanie niezależnie od parametrów sprzętu. Uwzględniono także konieczność utrzymania płynności działania gry poprzez zastosowanie ograniczenia liczby klatek na sekundę.

W ramach implementacji dokonano wyboru mechanizmów obsługi zdarzeń.

Zdecydowano się na wykorzystanie wbudowanego w Pygame systemu kolejkowania zdarzeń. Rozwiązanie to umożliwia przechwytywanie akcji użytkownika, takich jak ruch myszką czy wciśnięcie klawisza. Dzięki temu możliwe było zaimplementowanie intuicyjnego sterowania i szybkiej reakcji programu. Przyjęto, że interfejs użytkownika powinien być prosty i przejrzysty. Zaprojektowano ekrany startowe, menu oraz okna dialogowe, które prowadzą gracza przez kolejne etapy rozgrywki. Uwzględniono również elementy graficzne wspierające proces nauki programowania, np. podpowiedzi i komunikaty zwrotne.

Podczas implementacji gry przyjęto także założenie dotyczące systemu poziomów. Każdy poziom miał wprowadzać nową wiedzę z zakresu języka Python, a także weryfikować jej opanowanie poprzez zadania praktyczne.

## 3.1. Inicjalizacja środowiska

Celem implementacji było stworzenie w pełni interaktywnej aplikacji, która w angażujący sposób wspomaga naukę podstaw programowania.

Program został podzielony na logiczne moduły:

* inicjalizacja środowiska,
* definicje danych dydaktycznych (lekcje i zadania),
* interfejs użytkownika (UI),
* logika gry i ocena rozwiązań,
* mechanizmy zapisu postępu i wyników,
* pętla główna programu.

### **3.1.1. Inicjalizacja środowiska i konfiguracja**

Gra uruchamiana jest w oknie o rozdzielczości 900×700 pikseli.  
Już w momencie startu inicjalizowane są wszystkie zasoby: czcionki, kolory, stałe interfejsu oraz struktury danych.

Zdefiniowano paletę kolorów (tło, przyciski, teksty, ramki) oraz styl graficzny nawiązujący do estetyki edytorów kodu.  
Dzięki temu użytkownik uczy się w środowisku wizualnie zbliżonym do realnych narzędzi programistycznych.

### **3.1.2. Struktura danych dydaktycznych**

Gra wykorzystuje dwa główne zbiory danych:

* levels – zestawy zadań pogrupowanych według poziomów trudności: *Beginner*, *Intermediate*, *Advanced*, *Interactive*;
* lessons – lekcje teoretyczne prezentujące składnię i przykłady kodu.

Każde zadanie w levels opisane jest przez:

* typ ("code", "multiple\_choice", "fill\_in\_the\_blank", "drag\_drop"),
* treść ("description"),
* funkcję testującą poprawność ("test"),
* podpowiedź ("hint").

Przykład zadania:

|  |
| --- |
| "type": "code",  "description": "Challenge 1:\n Create a variable x with value 5.",  "test": lambda env: get\_case\_insensitive(env, "x") == 5,  "hint": "Try: x = 5" |

Fragment kodu 2 Fragment słownika levels

Każda lekcja w lessons zawiera tytuł, opis teoretyczny, przykład kodu i jego wynik:

|  |
| --- |
| "title": "Variables",  "description": "Variables are containers for storing data values...",  "example": ["age = 21", "name = 'Alice'", "print(age)"],  "output": "21 \nAlice" |

Fragment kodu 3 Fragment listy lessons

Dzięki temu możliwe jest przeplatanie nauki teoretycznej z praktycznymi zadaniami.

## 3.2. Interfejs graficzny

Interfejs graficzny gry został zaprojektowany w sposób prosty i czytelny, tak aby użytkownik mógł w intuicyjny sposób poruszać się pomiędzy poszczególnymi ekranami aplikacji. Całość oparto na bibliotece Pygame, która umożliwia tworzenie interaktywnych elementów graficznych oraz reagowanie na zdarzenia generowane przez użytkownika.

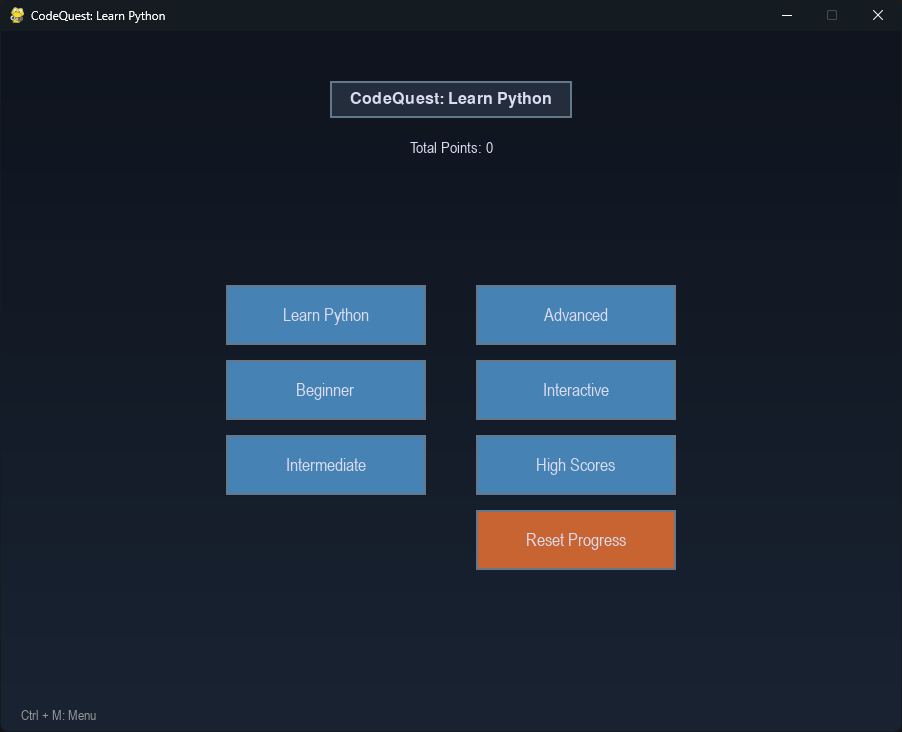
Układ wizualny gry opiera się na jednolitej kolorystyce (ciemne tło, kontrastowe przyciski) oraz prostych kształtach o zaokrąglonych krawędziach. Wszystkie ekrany gry posiadają stały element, dolny pasek pomocy, przypominający o klawiszach skrótu (np. „Ctrl + M: Menu”).

### **3.2.1. Menu główne**

Menu główne jest pierwszym ekranem widocznym po uruchomieniu gry. Zawiera tytuł projektu, bieżącą liczbę zdobytych punktów oraz zestaw przycisków prowadzących do poszczególnych sekcji aplikacji.

Użytkownik ma do wyboru:

* Learn Python – przejście do sekcji lekcji teoretycznych,
* Beginner / Intermediate / Advanced / Interactive – rozpoczęcie nauki na wybranym poziomie,
* High Scores – wyświetlenie tablicy najlepszych wyników,
* Reset Progress – wyczyszczenie zapisanych danych i rozpoczęcie gry od nowa.



Rysunek 1 Menu główne programu

Dodatkowo przy każdym poziomie wyświetlany jest status Completed, informujący o ukończeniu danego etapu.  
Fragment odpowiedzialny za generowanie menu przedstawiono poniżej:

|  |
| --- |
| def draw\_menu(mouse\_pos):      if show\_high\_scores:          draw\_high\_scores(mouse\_pos)          return      if show\_learn\_page:          draw\_learn\_page(mouse\_pos)          return      draw\_gradient\_bg(BG\_DARK, BG\_ACCENT)      title = TITLE\_FONT.render("CodeQuest: Learn Python", True, TEXT\_LIGHT)      title\_width, title\_height = title.get\_size()      title\_card = pygame.Rect(WINDOW\_WIDTH // 2 - title\_width // 2 - 20, 50, title\_width + 40, title\_height + 20)      pygame.draw.rect(screen, CARD\_BG, title\_card)      pygame.draw.rect(screen, BORDER, title\_card, 2)      screen.blit(title, (WINDOW\_WIDTH // 2 - title\_width // 2, 60))      points\_text = UI\_FONT.render(f"Total Points: {total\_points}", True, TEXT\_LIGHT)      screen.blit(points\_text, (WINDOW\_WIDTH // 2 - points\_text.get\_width() // 2, 90 + title\_height))      level\_order = ["Beginner", "Intermediate", "Advanced", "Interactive"]      total\_height = 4 \* BUTTON\_HEIGHT + 3 \* BUTTON\_SPACING      start\_y = (WINDOW\_HEIGHT - total\_height - title\_height - 70) // 2 + 90      left\_x = WINDOW\_WIDTH // 2 - BUTTON\_WIDTH - 25      right\_x = WINDOW\_WIDTH // 2 + 25      level\_rects = []      learn\_rect, \_ = draw\_button(left\_x, start\_y, BUTTON\_WIDTH, BUTTON\_HEIGHT, "Learn Python", mouse\_pos, BUTTON\_NORMAL, BUTTON\_HOVER)      for idx, level\_name in enumerate(["Beginner", "Intermediate"]):          y = start\_y + (idx + 1) \* (BUTTON\_HEIGHT + BUTTON\_SPACING)          color = BUTTON\_COMPLETED if level\_name in completed\_levels else BUTTON\_NORMAL          rect, is\_hover = draw\_button(left\_x, y, BUTTON\_WIDTH, BUTTON\_HEIGHT, level\_name, mouse\_pos, color, BUTTON\_HOVER)          level\_rects.append((rect, level\_name))          if level\_name in completed\_levels:              completed\_text = UI\_FONT.render("Completed", True, TEXT\_LIGHT)              screen.blit(completed\_text, (left\_x + BUTTON\_WIDTH - completed\_text.get\_width() - 10, y + 10))      for idx, level\_name in enumerate(["Advanced", "Interactive"]):          y = start\_y + idx \* (BUTTON\_HEIGHT + BUTTON\_SPACING)          color = BUTTON\_COMPLETED if level\_name in completed\_levels else BUTTON\_NORMAL          rect, is\_hover = draw\_button(right\_x, y, BUTTON\_WIDTH, BUTTON\_HEIGHT, level\_name, mouse\_pos, color, BUTTON\_HOVER)          level\_rects.append((rect, level\_name))          if level\_name in completed\_levels:              completed\_text = UI\_FONT.render("Completed", True, TEXT\_LIGHT)              screen.blit(completed\_text, (right\_x + BUTTON\_WIDTH - completed\_text.get\_width() - 10, y + 10))      high\_scores\_rect, \_ = draw\_button(right\_x, start\_y + 2 \* (BUTTON\_HEIGHT + BUTTON\_SPACING), BUTTON\_WIDTH, BUTTON\_HEIGHT, "High Scores", mouse\_pos, BUTTON\_NORMAL, BUTTON\_HOVER)      reset\_rect, \_ = draw\_button(right\_x, start\_y + 3 \* (BUTTON\_HEIGHT + BUTTON\_SPACING), BUTTON\_WIDTH, BUTTON\_HEIGHT, "Reset Progress", mouse\_pos, BUTTON\_RESET, BUTTON\_HOVER)      help\_text = HELP\_FONT.render("Ctrl + M: Menu", True, (150, 150, 150))      screen.blit(help\_text, (20, WINDOW\_HEIGHT - 25))      return level\_rects, reset\_rect, high\_scores\_rect, learn\_rect |

Fragment kodu 4 Funkcja draw\_menu

### **3.2.2. Ekran wyników (High Scores)**

Ekran wyników prezentuje listę pięciu najlepszych rezultatów osiągniętych przez graczy. Dla każdego wyniku zapisywane są:

* imię użytkownika,
* liczba zdobytych punktów,
* poziom trudności,
* data uzyskania wyniku.

Lista jest sortowana malejąco według liczby punktów, co pozwala na bieżące porównywanie osiągnięć.

|  |
| --- |
| def draw\_high\_scores(mouse\_pos):      draw\_gradient\_bg(BG\_DARK, BG\_ACCENT)      title = TITLE\_FONT.render("High Scores", True, TEXT\_LIGHT)      title\_width, title\_height = title.get\_size()      title\_card = pygame.Rect(WINDOW\_WIDTH // 2 - title\_width // 2 - 20, 50, title\_width + 40, title\_height + 20)      pygame.draw.rect(screen, CARD\_BG, title\_card)      pygame.draw.rect(screen, BORDER, title\_card, 2)      screen.blit(title, (WINDOW\_WIDTH // 2 - title\_width // 2, 60))      start\_y = 120      for i, score in enumerate(high\_scores[:5]):          score\_text = UI\_FONT.render(f"{i + 1}. {score['name']}: {score['score']} points ({score['level']}, {score['date']})", True, TEXT\_LIGHT)          screen.blit(score\_text, (WINDOW\_WIDTH // 2 - score\_text.get\_width() // 2, start\_y + i \* 30))      back\_rect, \_ = draw\_button(WINDOW\_WIDTH // 2 - BUTTON\_WIDTH // 2, start\_y + 180, BUTTON\_WIDTH, BUTTON\_HEIGHT, "Back", mouse\_pos, BUTTON\_NORMAL, BUTTON\_HOVER)      return back\_rect |

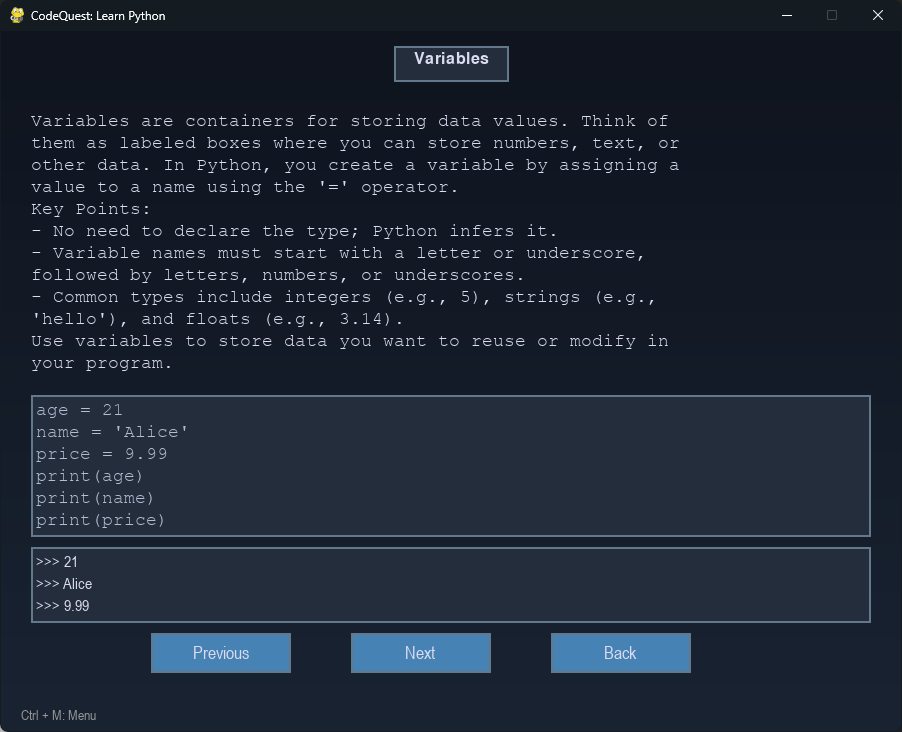
Fragment kodu 5 Funkcja draw\_high\_score

### **3.2.3. Ekran lekcji (Learn Page)**

Sekcja lekcyjna pełni funkcję dydaktyczną — wprowadza nowe pojęcia programistyczne i ilustruje je przykładami kodu. Każda lekcja zawiera:

* krótkie wprowadzenie teoretyczne,
* fragment kodu źródłowego,
* oczekiwany wynik działania programu.

Użytkownik może przechodzić między lekcjami za pomocą przycisków Next i Previous, a przycisk Back umożliwia powrót do menu głównego.



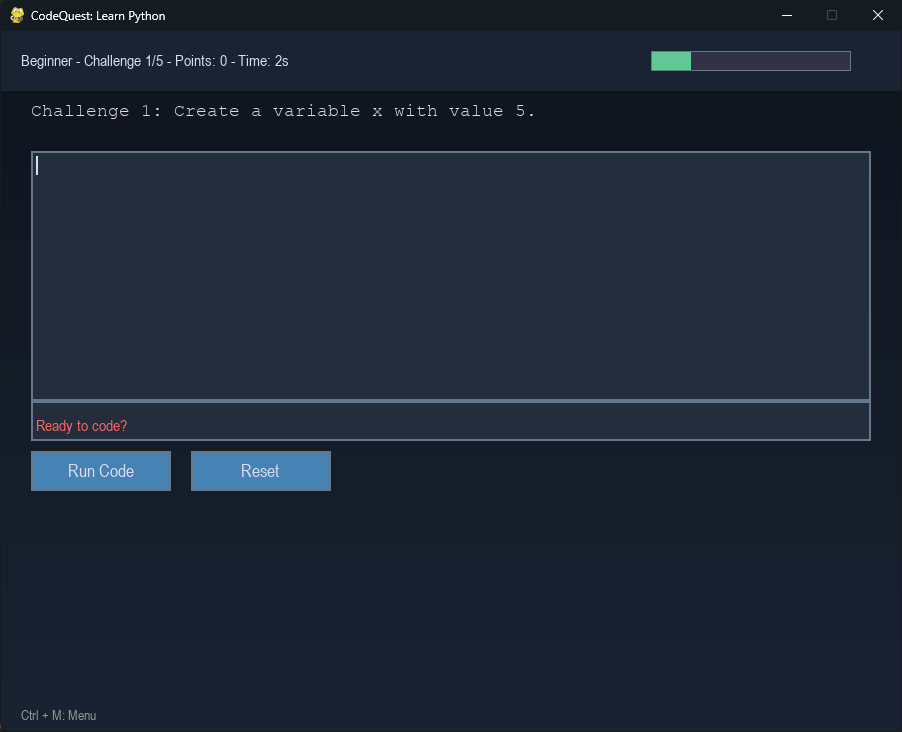
Rysunek 2 Okno Learn page

### **3.2.4. Ekran zadania kodowego (Code Challenge)**

Jednym z kluczowych elementów interfejsu jest ekran, na którym gracz rozwiązuje zadania programistyczne. Składa się on z trzech części:

* Opis zadania – tekstowy problem do rozwiązania,
* Okno edycji kodu – miejsce, w którym użytkownik wpisuje rozwiązanie,
* Pole informacji zwrotnej (Feedback) – wyświetla komunikaty o poprawności kodu.

Po naciśnięciu przycisku Run Code, gra uruchamia kod wprowadzony przez użytkownika i sprawdza jego wynik. Jeśli jest poprawny — gracz otrzymuje punkty i przechodzi do kolejnego zadania.



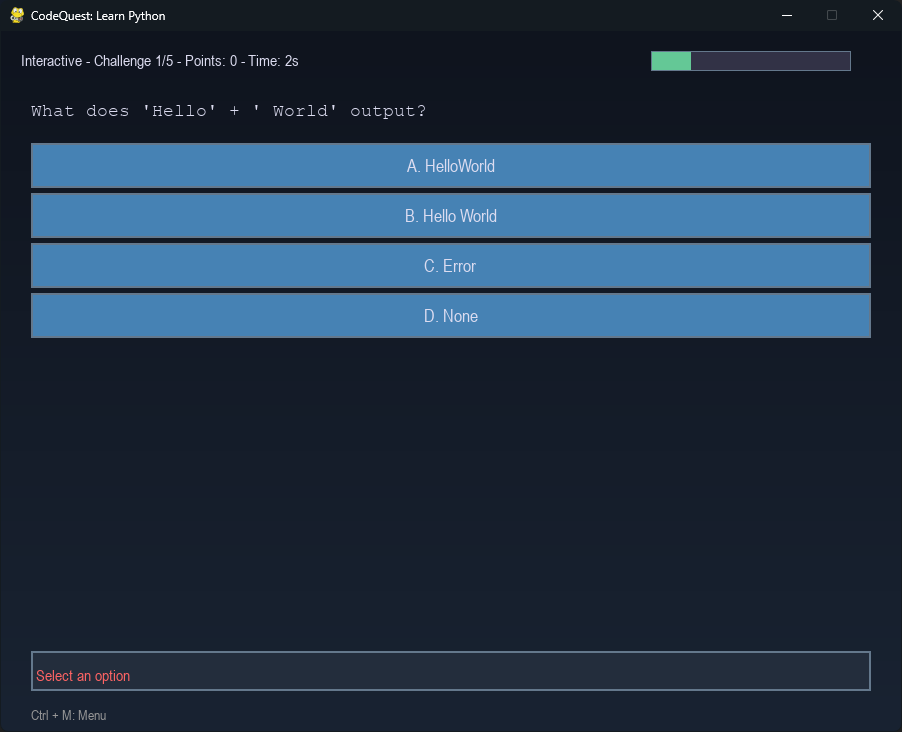
Rysunek 3 Okno z zadaniami kodowymi

### **3.2.5. Ekran zadań interaktywnych**

Oprócz klasycznych wyzwań programistycznych, gra zawiera także interaktywne ćwiczenia, takie jak:

* pytania wielokrotnego wyboru (multiple choice),
* uzupełnianie brakujących fragmentów kodu (fill in the blanks),
* przeciąganie elementów kodu w odpowiednie miejsca (drag & drop).

Każdy typ zadania zapewnia natychmiastową informację zwrotną, wspierając aktywne uczenie się.



Rysunek 4 Okno z zadaniami interaktywnymi

### **3.2.6. Ekran wprowadzania imienia (Name Input)**

Po ukończeniu poziomu użytkownik proszony jest o wpisanie swojego imienia, które następnie zostaje zapisane wraz z wynikiem.  
Interfejs tego ekranu jest prosty: pole tekstowe, przycisk Submit oraz komunikat informacyjny.

To rozwiązanie wprowadza element personalizacji i umożliwia tworzenie lokalnych rankingów.

### **3.2.7. Podsumowanie sekcji graficznej**

Zaprojektowany interfejs łączy prostotę obsługi z estetyczną i nowoczesną formą. Wszystkie elementy graficzne utrzymane są w spójnym stylu kolorystycznym i typograficznym.  
Dzięki zastosowaniu funkcji draw\_ poszczególne sekcje interfejsu są modularne, co ułatwia rozwój gry oraz dodawanie nowych ekranów i funkcjonalności.

## 3.3. Logika gry i ocena odpowiedzi

Centralnym elementem logiki są funkcje sprawdzające poprawność kodu i przyznające punkty.

### **3.3.1. Sprawdzanie kodu użytkownika**

W momencie uruchomienia kodu funkcja run\_user\_code():

* kompiluje wpisany kod (exec(code, env)),
* uruchamia funkcję testującą (puzzle["test"](env)),
* przyznaje punkty i przekazuje informację zwrotną.

W przypadku błędu program przechwytuje wyjątek i informuje użytkownika. Dzięki temu gracz od razu widzi skutki swoich działań, ucząc się przez doświadczenie.

### **3.3.2. Zadania interaktywne**

Obsługa interaktywnych quizów realizowana jest przez funkcję handle\_interactive\_click(), która porównuje wybory użytkownika z poprawnymi odpowiedziami

### **3.3.3. System punktacji**

Każde zadanie przyznaje **100 punktów bazowych**, z premią za szybkość rozwiązania:

|  |
| --- |
| def calculate\_points(time\_taken):      base\_points = 100      if time\_taken < 90:          bonus = int(50 \* (90 - time\_taken) / 60)      else:          bonus = 0      return base\_points + bonus |

Fragment kodu 6 System punktacji

### **3.3.4. Zapis postępu i wyników**

Postęp gry oraz tabele wyników są zapisywane w plikach JSON:

|  |
| --- |
| def save\_progress():      with open(PROGRESS\_FILE, 'w') as f:          json.dump({"completed\_levels": list(completed\_levels), "total\_points": total\_points}, f)  def save\_high\_scores():      high\_scores.sort(key=lambda x: x["score"], reverse=True)      with open(HIGH\_SCORES\_FILE, 'w') as f:          json.dump(high\_scores[:5], f) |

Fragment kodu 7 Funkcje zapisu postępu i najlepszych wyników

### **3.3.5. Główna pętla programu**

Serce gry stanowi pętla główna (Main Loop), odpowiedzialna za obsługę zdarzeń, aktualizację interfejsu i logiki.

Schemat działania pętli obejmuje:

* odczyt zdarzeń z klawiatury i myszy,
* aktualizację interfejsu i logiki,
* renderowanie nowej klatki,
* zapis danych przy zakończeniu.

### **3.3.6. Mechanizmy pomocnicze**

W projekcie zastosowano liczne funkcje pomocnicze, zwiększające czytelność i ergonomię gry:

* capture\_output() - przekierowuje wynik działania kodu do bufora,
* get\_case\_insensitive() - wyszukuje zmienne w kodzie niezależnie od wielkości liter,
* basic\_syntax\_highlight() - nadaje kolor składni kodu (słowa kluczowe, ciągi znaków),
* set\_cursor\_from\_mouse() - ustawia kursor tekstowy w miejscu kliknięcia.

# Zakończenie

Praca miała na celu opracowanie i implementację gry edukacyjnej wspierającej naukę podstaw programowania i logiki. Odpowiadając na pytania badawcze, wykazano, że gry komputerowe mogą skutecznie angażować graczy w proces nauki, szczególnie gdy stosują mechanizmy interaktywne i elementy blokowego programowania.

Testy gry wykazały, że zastosowanie metodologii opartej na interaktywnych zadaniach pozwala na lepsze przyswajanie wiedzy i rozwijanie umiejętności logicznego myślenia. Tym samym potwierdzono postawioną hipotezę, że edukacyjna gra komputerowa może skutecznie wspierać proces nauki programowania.

W przyszłości warto rozważyć rozwój projektu o dodatkowe funkcjonalności oraz przeprowadzenie szerzej zakrojonych badań nad skutecznością gry w różnych grupach wiekowych. Uzyskane wyniki mogą być podstawą do dalszego rozwijania narzędzi edukacyjnych bazujących na technologii gier komputerowych.

# Literatura

### **Pozycje zwarte i artykuły naukowe**

1. Barry, P. (2024). Python. Rusz głową! Wydanie III. Wydawnictwo Helion, Gliwice.
2. Bell, A. (2019). Python. Uczymy się programowania. Wydawnictwo Helion, Gliwice.
3. Briggs, J. (2023). Python dla dzieci : programowanie na wesoło. PWN.
4. Ceder, N. (2019). Python. Szybko i prosto. Wydawnictwo Helion, Gliwice.
5. Gaddis, T. (2019). Python dla zupełnie początkujących. Wydawnictwo Helion, Gliwice.
6. Jaśniewski, T. (2024). Python : zbiór zadań z rozwiązaniami. Wydawnictwo Helion, Gliwice.
7. Lubanovic, B. (2020). Python. Nowoczesne programowanie w prostych krokach. Wydanie II. Wydawnictwo Helion, Gliwice.
8. Matthes, E. (2023). Python. Instrukcje dla programisty. Wydanie III. Wydawnictwo Helion, Gliwice.
9. Matusiewicz, Z. (2023). Zacznij od Pythona. Programowanie dla młodzieży w praktyce. Wydawnictwo Helion, Gliwice.
10. Moskala, M. (n.d.). Python od podstaw. Zacznij swoją przygodę z programowaniem.
11. Sarbicki, G. (2019). Python : kurs dla nauczycieli i studentów. Wydawnictwo Helion, Gliwice.
12. Sweigart, A. (2017). Twórz własne gry komputerowe w Pythonie. Wydawnictwo Helion, Gliwice.
13. Zichermann, G., & Cunningham, C. (2012). Grywalizacja. Mechanika gry na stronach WWW i w aplikacjach mobilnych. Wydawnictwo Helion, Gliwice.
14. Parker, J. R. (2024). Game Development Using Python. Mastering Interactive

Game Creation and Development through Python. Wydawnictwo Helion,

Gliwice.

1. Raheja, N. R. (2023). Python Programming for Student. Wydawnictwo Helion, Gliwice.
2. Dey, S. (2025). Image Processing and Computer Vision Masterclass with Python - 2nd Edition. Wydawnictwo Helion, Gliwice.
3. Dawson, M. (2020). Python dla każdego. Podstawy programowania. Wydanie III. Wydawnictwo Helion, Gliwice.
4. Matthes, E. (2023). Python. Instrukcje dla programisty. Wydanie III. Helion,Gliwice.
5. • Juul, J. (2005). Half-Real: Video Games between Real Rules and Fictional Worlds. Cambridge, MA: The MIT Press.
6. •Salen, K., i Zimmerman, E. (2004). Rules of Play: Game Design

Fundamentals. Cambridge, MA: The MIT Press

1. Filiciak, M. (2006). Wirtualny plac zabaw. Gry sieciowe i przemiany kultury współczesnej. Warszawa: Wydawnictwa Akademickie i Profesjonalne
2. Hainey, T., Connolly, T., Boyle, E., Wilson, A., i Razak, A. (2024). A serious game for programming in higher education. Computers & Education. Elsevier.
3. Videnovik, M., Janeska-Iliev, A., i Trajkovik, V. (2023). Game-based learning in computer science education. Journal of STEM Education Research.

SpringerOpen.

1. Lutz, M. (2016). Python. Wprowadzenie. Wydanie V. Helion.
2. Gui, Y. I in. (2023). Effectiveness of digital educational game and game design in STEM education: A meta-analysis. STEM Education Journal
3. Garcia, M. B., i Revano, T. F. (2021). Assessing the role of Python programming gamified course on students’ knowledge, skills performance, attitude, and self-efficacy. IEEE Xplore.
4. Deterding, S., Dixon, D., Khaled, R., i Nacke, L. (2011). From game design elements to gamefulness: Defining “gamification”. *Proceedings of the 15th International Academic MindTrek Conference*
5. Werbach, K., i Hunter, D. (2012). *For the Win: How Game Thinking Can Revolutionize Your Business*. Wharton Digital Press.
6. Kapp, K. M. (2012). *The Gamification of Learning and Instruction: Game-based Methods and Strategies for Training and Education*. Pfeiffer
7. Domínguez, A., Saenz-de-Navarrete, J., de-Marcos, L., Fernández-Sanz, L., Pagés, C., & Martínez-Herráiz, J. J. (2013). Gamifying learning experiences: Practical implications and outcomes. *Computers & Education*

### **Netografia**

1. Python Academy. Kurs Python Academy.
2. Python 101. Podstawy Pythona.
3. Python Software Foundation. Dokumentacja Pythona.
4. Python.org. (n.d.). Python w Polsce. <https://pl.python.org>
5. Pystart. Pystart.pl - kurs programowania w Pythonie dla początkujących.
6. Real Python. (2024). Top Python game engines. Real Python. https://realpython.com/top-python-game-engines

# Spis fragmentów kodu

[Fragment kodu 1 Przykładowe zadanie z odpowiedzią i podpowiedzią 14](#_Toc211550234)

[Fragment kodu 2 Fragment słownika levels 25](#_Toc211550235)

[Fragment kodu 3 Fragment listy lessons 26](#_Toc211550236)

[Fragment kodu 4 Funkcja draw\_menu 28](#_Toc211550237)

[Fragment kodu 5 Funkcja draw\_high\_score 29](#_Toc211550238)

[Fragment kodu 6 System punktacji 33](#_Toc211550239)

[Fragment kodu 7 Funkcje zapisu postępu i najlepszych wyników 33](#_Toc211550240)

# Spis rysunków

[Rysunek 1 Menu główne programu 27](#_Toc211550448)

[Rysunek 2 Okno Learn page 30](#_Toc211550449)

[Rysunek 3 Okno z zadaniami kodowymi 31](#_Toc211550450)

[Rysunek 4 Okno z zadaniami interaktywnymi 32](#_Toc211550451)

# Załączniki

Kod źródłowy - Projekt gry edukacyjnej.py