**МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ**

**ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ ПЕДАГОГІЧНИЙ УНІВЕРСИТЕТ**

**ІМЕНІ Г.С.СКОВОРОДИ**

Фізико-математичний факультет

Кафедра інформатики

Освітня програма «Інформатика в закладах освіти»

Спеціальність 014 Середня освіта (Інформатика)

**КВАЛІФІКАЦІЙНА РОБОТА**

**на здобуття освітнього ступеня магістр**

**НАВЧАННЯ ШКОЛЯРІВ ОСНОВ ПРОГРАМУВАННЯ ЗА ДОПОМОГОЮ СЕРЕДОВИЩА MINECRAFT**

здобувача другого (магістерського) рівня вищої освіти

**Сівочки Ігоря Глібовича**

Науковий керівник Олефіренко Надія Василівна

доктор педагогічних наук, професор

Рецензент Андрієвська Віра Михайлівна,

доктор педагогічних наук, доцент

Харків – 2021

**ЗМІСТ**

[ВСТУП 3](#_Toc88567326)

[РОЗДІЛ 1. СУЧАСНІ ПІДХОДИ ДО НАВЧАННЯ ШКОЛЯРІВ ОСНОВ ПРОГРАМУВАННЯ 7](#_Toc88567327)

[1.1. Об’єктно-орієнтоване програмування як сучасна парадигма програмування 7](#_Toc88567328)

[1.2. Місце вивчення основ об’єктно-орієнтованого програмування у шкільному курсі інформатики 11](#_Toc88567329)

[1.3. Підходи до навчання програмування 16](#_Toc88567330)

[Висновки до розділу 1 26](#_Toc88567331)

[РОЗДІЛ 2. РОЗРОБКА ПРАКТИЧНИХ ЗАВДАНЬ ДЛЯ НАВЧАННЯ ШКОЛЯРІВ ОСНОВ ПРОГРАМУВАННЯ МОВОЮ PYTHON В СЕРЕДОВИЩІ MINECRAFT 27](#_Toc88567332)

[2.1. Специфіка середовища Minecraft для навчання програмування 27](#_Toc88567333)

[2.2. Розробка тренувальних завдань для навчання школярів основ програмування в середовищі Minecraft 29](#_Toc88567334)

[2.3. Розробка ігрових ситуацій для школярів в середовищі Minecraft 38](#_Toc88567335)

[Висновки до розділу 2 44](#_Toc88567336)

[РОЗДІЛ 3. ЕКСПЕРИМЕНТАЛЬНЕ ВПРОВАДЖЕННЯ ДИДАКТИЧНИХ МАТЕРІАЛІВ В ПОЗАШКІЛЬНОМУ НАВЧАННІ 45](#_Toc88567337)

[3.1. Організація та хід упровадження розроблених практичних завдань 45](#_Toc88567338)

[3.2. Аналіз результатів експерименту 46](#_Toc88567339)

[Висновки до розділу 3 49](#_Toc88567340)

[ЗАГАЛЬНІ ВИСНОВКИ 50](#_Toc88567341)

[СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 51](#_Toc88567342)

[ДОДАТКИ 54](#_Toc88567343)

# ВСТУП

Для сучасної молоді все більш актуальним стає вивчення мов програмування для подальшого працевлаштування. Тому дуже важливо розвивати інтерес школярів до цієї галузі. Це можна робити за допомогою впровадження в освітній процес різних ігрових елементів. Зокрема, при вивченні базових принципів програмування можна використовувати онлайн- тренажери або ігри, наприклад Scratch, CodeMonkey, Tynker, Kodu чи Codecombat. Але підліткам важливо поєднувати ігрові технологім з навчанням. З цією метою у нагоді стане середовище Minecraft, яке на даний час має потужну освітню компоненту і надає різні можливості для впровадження у навчальний процес. Minecraft – це гра жанру «пісочниця», де немає сюжетної лінії, але є нескінченний світ, в якому можна творити - створювати, будувати, взаємодіяти з іншими гравцями.. Зараз Minecraft це не тільки гра, але й освітня платформа, яку використовують більше ніж в 1000 шкіл по всьому світу: в США, Фінляндії, Швеції, Австралії і навіть Росії. Школярі у цьому середовищі досліджують кораблі, пишуть твори, здійснюють вимірювання, вивчають тривимірну систему координат, будують моделі атомів та молекул, знайомляться з джерелами енергії тощо. Завдяки своїй гнучкості гра легко підлаштовується під різні дисципліни. Гармонійне поєднання ігрових та навчальних технологій дає змогу учням сприймати складний навчальний матеріал у ігровій формі, що більш цікаво та комфортно для них.

На сучасному етапі розвитку шкільної інформатики статус об'єктно-орієнтованого програмування в базовому курсі досі повністю не визначений. Основна мета вивчення об'єктно-орієнтованого програмування - моделювання об'єктів реального світу й опис взаємодії між ними. Об'єктно-орієнтоване програмування є провідним підходом у програмуванні і реалізується сьогодні вже практично в будь-якій сучасній мові програмування. Вивчення даної парадигми програмування передбачає розгляд основних понять об'єктно-орієнтованого програмування (клас; екземпляр класу; характеристики, методи, властивості, індексатори класу; інтерфейси; абстрактні класи), роботи з великим числом вже створених бібліотек і створенням персональних компонентів.

Широке застосування ідей об'єктно-орієнтованого підходу вимагає глибокого знайомства з його ідеями, а це наштовхує на цілу низку труднощів. По-перше, знайомство з об'єктно-орієнтованим програмуванням зазвичай супроводжується одночасним вивченням нових для учня конструкцій мови програмування. По-друге, часто дуже складно знайти відповідні ретельно продумані матеріали з обговорюваної теми, а також програмні підтримки для закріплення вивченого матеріалу. Існують й інші фактори, що ускладнюють викладання основ об’єктно-орієнтованого програмування. Незважаючи на існування різних ідей з приводу навчання основ об'єктно-орієнтованого програмування, єдина позиція в цьому питанні досі не сформувалася.

Сучасні учителі прикладають багато зусиль для оновлення змісту завдань для школярів, впроваджують інноваційні технології, пристовують завдання до інтересів учнів, використовують ігрові середовища або ігрові завдання. Але наразі бракує методичних розробок з навчання програмуванню, які би враховували інтереси школярів та були доступними для сприйняття. На наш погляд, розробка практичних завдань з навчання мови Python у середовищі MineCraft є цілком доречною і актуальною на даний час.

**Об’єкт дослідження:** освітній процес у закладах середньої освіти.

**Предмет дослідження:** навчання школярів основ об’єктно-орієнтованого програмування засобами середовища Minecraft.

**Ступінь дослідження проблеми в спеціальній науковій літератур**і. Проблеми навчання школярів програмування висвітлені у багатьох наукових працях. Питання навчання молодших школярів програмуванню мовою Scratch висвітлені у роботах В.Вдовенко, М.Гладун, Г.Громко, І.Зарецька, М.Корнієнко, С.Крамаровська, Г.Ломаковська, Н.Морзе, Т.Огієнко, Е.Патаракіна, Г.Проценко, Н.Стрілецької. Роботи Т. Барболіна, К.Бек, О.Лінник, С. Семерікова, О.Теплицького, Ю.Тріуса, присвячені окремим питанням навчання учнів базової та старшої школи основ програмування. Наукові публікації цих авторів свідчать про актуальність пошуку ефективних шляхів навчання учнів програмування.

**Мета дослідження**: визначити способи навчання учнів об’єктно-орієнтованому програмуванню, а також розробити практичні завдання з основ об'єктно-орієнтованого програмування для навчання учнів 7-9 класів мови програмування Python у середовищі Minecraft.

Відповідно до мети, поставлено такі **завдання**:

1. Розкрити роль ООП у системі інформатичної підготовки учнів 7-9 класів і розглянути методичні аспекти викладання інформатики.
2. Схарактеризувати основні підходи до навчання програмування й описати ООП як сучасну технологію в розробці програмного забезпечення.
3. Розробити практичні завдання з основ об'єктно-орієнтованого мовою Python у середовищі MineCraft.

**Джерельну базу дослідження** складають нормативні документи (навчальні програми, концепції, положення), підручники з предмету, методичні посібники, публікації у наукових журналах – всього 17 найменувань.

**Методи дослідження**: для вирішення поставлених завдань та досягнення мети було використано такі методи дослідження: *теоретичні*: вивчення навчальних програм, підручників, методичних посібників, а також аналіз статей, наукових досліджень, публікацій та метод *проектування*.

**Наукова новизна отриманих результатів:** *вперше*виділено підходи до навчання програмування (аналіз готових кодів, використання готових блоків програм, використання простих конструкцій, розв’язання комплексних завдань, комбінації підходів)*; уточнено*роль ООП у системі інформатичної підготовки учнів 7-9 класів*; запропоновано* практичні завдання для навчання основ об’єктно-орієнтованого програмування учнів .

**Практичне значення отриманих результатів** полягає в тому, що розроблені практичні завдання можуть бути використаними для навчання об’єктно-орієнтованого програмування на уроках інформатики у 7-9 класах.

**Апробація дослідження:** основні положення й результати дослідження доповідалися на науково-практичній конференції Інноваційні педагогічні технології в цифровій школі (14-15 травня, 2020 р.), науково-методичній конференціях «Наумовські читання» 24-25 листопада, 2020 р., 23-24 листопада 2021 року.

**Публікації**: результати дослідження висвітлено у друкованих працях: науково-методична конференція «Наумовські читання» 23-24 листопада 2021 року, VІ міжнародна науково-практична конференція «Сучасна наука: проблеми та перспективи» 12-13 січня 2022 року.

# РОЗДІЛ 1. СУЧАСНІ ПІДХОДИ ДО НАВЧАННЯ ШКОЛЯРІВ ОСНОВ ПРОГРАМУВАННЯ

# 1.1. Об’єктно-орієнтоване програмування як сучасна парадигма програмування

Навчання програмування – найбільш розроблена частина методики інформатики й має давню історію. Перші роки після введення в школі інформатики як обов'язкового предмета, школярі вивчали, в основному, саме програмування. Тому існує велика кількість методичних розробок та навчальних програм різних курсів програмування. Ускладненою обставиною для вчителя при виборі такого курсу є те, що існують різні парадигми програмування і необхідно визначитися з вибором мови або системи програмування.

Певний спосіб мислення (парадигма) служить основою для створення мови програмування. Існують різні парадигми програмування - процедурна, об'єктно-орієнтована, функціональна, логічна. Кожна з парадигм використовується для розв’язання певного класу задач. Деякі мови підтримують кілька парадигм, інші ж, навпаки, орієнтовані на реалізацію тільки однієї парадигми. Для кожної мови програмування одна з парадигм є основною, а інші парадигми - додатковими.

Як правило, для початківців першою парадигмою стає процедурна. У межах процедурної парадигми програміст складає послідовність викликів процедур (функцій) для виконання поставленого завдання. При цьому, будь-які процедури (функції) можуть використовувати будь-які дані за умови відповідної кількості і типів параметрів. Така ситуація виникає у зв'язку з тим, що дані і процедури (функції) для їх обробки не пов'язані за змістом; отже, неможливо захистити дані від неправильного використання. Крім того, застосування процедурного програмування пов'язане з певними труднощами для створення великих програмних систем (великі програми важко створювати, налагоджувати, супроводжувати).

Об'єктно-орієнтоване програмування (ООП) виникло в першу чергу у відповідь на зростаючу складність програмного забезпечення. ООП є результатом розвитку процедурного програмування, проте пропонує інший підхід до розробки програм. Використання засобів ООП дає наступні основні переваги: зменшення складності програмного забезпечення, підвищення його надійності програмного забезпечення, забезпечення можливості модифікації окремих компонент програм без зміни решти компонент, забезпечення можливості повторного використання окремих компонент програмного забезпечення.

У основі ООП лежить об'єктний підхід, коли прикладна наочна область представляється у вигляді сукупності об'єктів, які взаємодіють між собою за допомогою передачі повідомлень. Об'єкт - це сукупність даних (змінних) і способів роботи з ними (компонентних процедур і функцій). Стан об'єкта характеризується переліком всіх його можливих властивостей – структурою і значеннями кожної з цих властивостей. Поведінка об'єкта (або його функціональність) характеризує те, як об'єкт взаємодіє з іншими об'єктами або піддається впливу інших об'єктів, проявляючи свою індивідуальність. Поведінка об'єкта реалізується у вигляді функцій, які називають методами. При цьому структура об'єкта доступна тільки через його методи, які в сукупності формують інтерфейс об'єкта. Індивідуальність об'єкта характеризують такі властивості, які відрізняють його від всіх інших об'єктів [1].

Такий підхід дозволяє локалізувати ухвалювані рішення рамками об'єкту, об'єднуючи в нім і структуру, і поведінку, а, отже, знижуючи складність окремої програми (що реалізовує об'єкт). Ця ідея об'єднання структури і поведінки в одному місці і заховування всіх даних усередині об'єкту, що робить їх невидимими для всіх, за винятком методів самого об'єкту називається інкапсуляцією. Це дозволяє об'єктам функціонувати абсолютно незалежно один від одного, приховуючи за інтерфейсом деталі реалізації. Інкапсуляція дозволяє розглядати об'єкти, як ізольовані "чорні ящики", які знають і уміють виконувати певні дії. З цієї точки зору, внутрішній устрій "чорних ящиків" для нас значення не має, нам все одно, що відбувається усередині. Важливо тільки знати, що треба покласти в ящик при зверненні до нього і що ми при цьому з нього отримаємо. Таким чином, об'єкти об'єктно-орієнтованих систем - це мінімальні одиниці інкапсуляції.

Одна з ключових концепцій об'єктно-орієнтованого програмування - ідея угрупування об'єктів в класи, відповідно до того як вони влаштовані і діють. Така ідея вперше була реалізована ще в 60-і роки в мові Simula. Під класом розуміється безліч об'єктів, зв'язаних спільністю структури і поведінки. Клас можна порівняти з шаблоном, по якому створюються об'єкти. Саме клас спочатку описує змінні і методи об'єкту, тобто структуру і поведінку об'єкту, і визначає механізми створення об'єкту, що реально існує в системі, який, коли створюється, є екземпляр класу.

Наведення за допомогою класів порядку в світі об'єктів - велике досягнення, але можна піти далі, визначаючи деякий порядок і серед класів. Досягається це за допомогою введення механізму спадкоємства – найпотужніший засіб в об'єктно-орієнтованій системі, оскільки він дозволяє багато разів використовувати один і той же створений код. Механізм спадкоємства дуже простий: один клас, званий в рамках цих відносин суперкласом, повністю передає іншому класу, який називається підкласом, свою структуру і поведінку, тобто всі свої змінні і всі методи. Що далі робити з цим багатством визначає тільки підклас: він може додати в структуру щось своє, щось з успадкованого інтерфейсу він може використовувати без змін, щось змінити, і, зрозуміло, може додати свої власні методи. Тобто клас за допомогою підкласів розширюється, і як результат, створювані об'єкти стають все більш і більш спеціалізованими. Класи, розташовані за принципом спадкоємства, починаючи з найзагальнішого, базового класу, утворюють ієрархію класів.

Система, що реалізоує такі принципи побудови, має більш складні вимоги, ніж її структурний аналог, до продуктивності обчислювальної системи.

Сучасна актуальність ООП полягає в тому, що систематичне застосування засобів об'єктно-орієнтованого програмування дозволяє розробляти добре структуровані, надійні в експлуатації програмні системи, що досить просто модифікуються. Цим пояснюється вибір багатьох галузей професійної діяльності до об'єктно-орієнтованого підходу і об'єктно-орієнтованим мовам програмування. Тому ознайомлення з ним необхідне для учнів, орієнтованих на професію програміста. Вивчення подібного профільного курсу спрямоване на виконання таких завдань:

* засвоєння методології об'єктно-орієнтованого програмування;
* вивчення техніки програмування на одній з мов;
* розширення загального кругозору учнів.

Шкільний курс об’єктно-орієнтованого програмування, з одного боку, повинен бути сучасним, а з іншого - бути елементарним і доступним для вивчення. Поєднання цих двох, багато в чому суперечливих, вимог є складним завданням.

Наразі принципи ООП в основному пояснюються на прикладах фрагментів консольних додатків у тексті підручника. На практиці відразу пропонується розробка віконних додатків. У зв'язку з цим принципи ООП так і залишаються деякою абстракцією, не випробувані учнями. Зрозуміло, що трудомісткість розробки, складність матеріалу і звичайна нестача часу не дозволяють запропонувати учням з різними здібностями розробляти з нуля програми з використанням власних класів.

За класифікацією Ю.К. Бабанського [2], у групі методів навчання для організації і здійснення навчально-пізнавальної діяльності виділяються наступні методи: перцептивні, словесні, наочні, практичні, логічні, гностичні та самоврядування навчальними діями. найбільш оптимальним при вивченні складних тем з програмування, зокрема принципів ООП, є поєднання наочного, практичного та гностичного методів навчання.

Практичний метод навчання може бути реалізований за допомогою вправ, лабораторних робіт, практичних робіт або пізнавальних ігор. Зазвичай проводять практичні роботи, які спрямовані на формування вміння застосовувати знання на практиці. Практична робота проводиться після вивчення теоретичного матеріалу з основ ООП і включає:

* попередній інструктаж;
* виконання роботи;
* поетапний контроль виконання з озвучуванням необхідного результату;
* перевірку та оцінювання.

Безумовно, що при розгляді технології ООП повинна бути його реалізація в інтегрованому середовищі розробки програмного забезпечення. При використанні візуального середовища з'являється можливість проєктувати деяку частину, наприклад, інтерфейси майбутнього продукту із застосуванням візуальних засобів додавання і налаштування спеціальних бібліотечних компонентів.

# 1.2. Місце вивчення основ об’єктно-орієнтованого програмування у шкільному курсі інформатики

Зосередимо увагу на освітню програму з інформатики для учнів 5-9 класів на порталі МОН України (рис 1.) [3, 4].

Почнемо пошук місця вивчення основ ООП з сьомих класів. Тому, що саме у 7, 8 і 9 класах програмою передбачено години на розв’язування компетентнісних задач, які є однією з важливих ділянок роботи в системі навчання інформатики.

На етапі 7 класу не ставиться завдання глибокого вивчення ІКТ, акцент зроблено на набутті навичок практичного застосування інформаційно-комунікаційних технологій, а також на розвивальній спрямованості навчання. На вивчення інформатики у сьомому класі передбачено 35 годин, які розподіляються на три великі блоки: служби Інтернету, опрацювання табличних даних, алгоритми з повторенням і алгоритми з розгалуженням. Зупинімося більш детально на останньому блоці «Алгоритми з повторенням і розгалуженням».

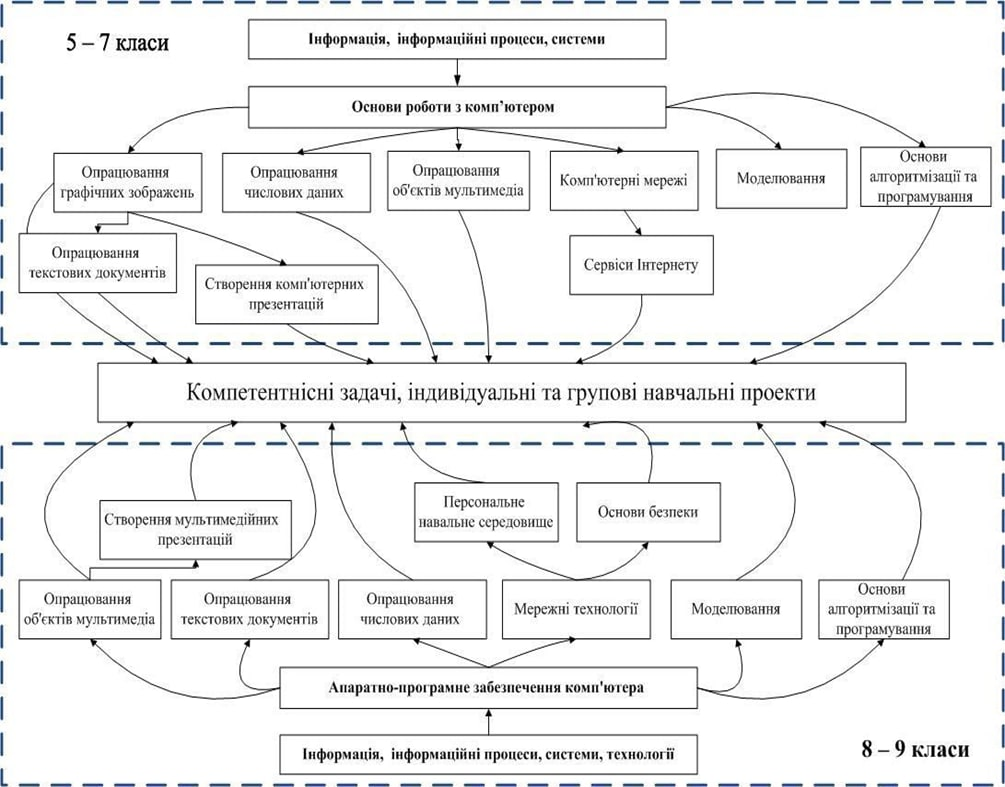


Рис. 1.1. Теми освітньої програми з інформатики та їх змістові залежності

В освітній програмі зазначено, що учень повинен *уміти*:

* визначати правильність або неправильність простих висловлювань і умовних висловлювань «Якщо – то»;
* формально виконувати алгоритми з повторенням та розгалуженням з навчальної діяльності та побуту;
* складати та виконувати алгоритми з повтореннями,у визначеному навчальному середовищі;
* складати та виконувати алгоритми з розгалуженням у визначеному навчальному середовищі;
* складати та виконувати алгоритми з повторенням та розгалуженням у визначеному навчальному середовищі.

Також учень:

*пояснює поняття*: висловлювання.

*розрізняє*: алгоритмічні структури слідування, розгалуження, повторення; правильні (істинні) та неправильні (хибні) висловлювання.

*наводить приклади*: структур повторення та розгалуження в алгоритмах із життя та навчальної діяльності;істинних і хибних висловлювань.

*формулює*: умовні висловлювання «Якщо – то»[4].

Відповідно до освітньої програми восьмого класу[4] на уроки інформатики виділяється вже 70 годин. Але й кількість блоків маємо більшу: кодування даних та апаратне забезпечення, опрацювання текстових даних, створення та публікація веб-ресурсів, опрацювання мультимедійних об’єктів, алгоритми та програми. У восьмому та дев’ятому класі відбувається повноцінне формування предметних ІТ-компетентностей. У восьмому класі учень повинен *уміти*:

* відкривати середовище програмування;
* створювати новий проект, відкривати, зберігати й закривати проект;
* компілювати й виконувати об’єктно-орієнтовану програму;
* настроювати параметри проекту та інтерфейсу середовища розробки;
* відкривати й закривати вікна та панелі інструментів, що є у середовищі розробки програм;
* настроювати параметри форми;
* додавати до форми елементи керування та введення даних, переміщувати їх та змінювати їхній розмір;
* відкривати вікно обробника події, пов’язаної з елементом керування;
* редагувати код обробника події натискання кнопки;
* виводити текстові повідомлення у вікнах повідомлень;
* визначати текст і властивості кнопки на формі;
* програмно визначати вміст написів на формі;
* зчитувати дані різних типів, за допомогою елементів введення даних з метою їх подальшої обробки.

Також учень:

*пояснює*: програмного проекту; об’єкта у програмуванні; властивостей і методів об’єкту; події й обробника події; елемента керування.

*описує*: інтерфейс середовища програмування; поняття програмного проекту; поняття форми й елементу керування; об’єкту, його властивостей та методів; поняття події й обробника події; метод редагування коду обробника події; метод програмування відображення повідомлень у вікнах повідомлень.

*має уявлення*: про величини; про типи величин[4].

У дев’ятому класі учні вже мають більш глибокі знання з блоку «алгоритми і програми», а також знайомляться з новим блоком «бази даних та системи управління базами даних». На вивчення інформатики передбачено 70 годин,і складається з п’яти блоків[4]: програмне забезпечення та інформаційна безпека, 3D-графіка, опрацювання табличних даних, бази даних та СУБД, алгоритми та програми. У дев’ятому класі учень повинен *уміти*:

* складати алгоритми опрацювання табличних величин;
* складати, редагувати, налагоджувати та виконувати програми опрацювання табличних величин у навчальному середовищі програмування;
* аналізувати результати виконання програм опрацювання табличних величин;
* за допомогою графічних примітивів відображати табличні величини.

Також учень:

*пояснює зміст поняття*: таблична величина; елемент табличної величини.

*пояснює*: методи знаходження сум і кількостей значень елементів табличної величини, що задовольняють заданими умовам; методи пошуку елементів табличної величини, що задовольняють заданим умовам.

*описує*: властивості табличної величини та її елементів[4].

Час, виділений на кожну тему, визначається вчителем відповідно до рівня попередньої підготовки учнів, а також обраної методики навчання. Але є рамки доступної кількості годин на кожну тему (табл. 1) [4].

Таблиця 1

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **№** | **Назва розділу** | **Класи і кількість годин** | | | | |  |
| **5**  **кл.** | **6**  **кл.** | **7**  **кл.** | **8**  **кл.** | **9**  **кл.** | **Всього** |
| 1 | Інформація, інформаційні процеси, системи, технології | 4 | – | – | 3 | 2 | 9 |
| 2 | Комп’ютер як універсальний пристрій для опрацювання даних | 10 | 6 | – | 5 | – | 21 |
| 3 | Інформаційні технології |  |  |  |  |  |  |
| 3.1 | Створення та опрацювання текстових документів | – | 8 | – | 6 | 5 | 19 |
| 3.2 | Створення та опрацювання графічних зображень | 9 | – | – | – | 6 | 15 |
| 3.3 | Створення та опрацювання об’єктів мультимедіа | – | 4 | – | 6 | – | 10 |
| 3.4 | Створення та опрацювання комп'ютерних презентацій | 9 | – | – | – | 6 | 15 |
| 3.5 | Створення та опрацювання числових даних | – | – | 8 | 10 | – | 18 |
| 4 | Комп’ютерні мережі | – | 8 | 4 | – | 16 | 28 |
| 5 | Моделювання | – | – | 3 | – | 6 | 9 |
| 6 | Основи алгоритмізації та програмування | – | 7 | 9 | 28 | 10 | 54 |
| 7 | Розв’язування компетентнісних задач, виконання індивідуальних і групових навчальних проектів | – | – | 8 | 9 | 15 | 32 |
| 8 | Резерв | 3 | 2 | 3 | 3 | 4 | 15 |
|  | Всього | 35 | 35 | 35 | 70 | 70 | 245 |

Потрібно зауважити, що такі теми, як «Моделювання» та «Основи алгоритмізації та програмування», вивчаються не менше 40 % навчального часу в шостих–восьмих класах і не менше 30 % у дев’ятому класі. Послідовність вивчення тем може змінюватись учителем з необхідності.

# 1.3. Підходи до навчання програмування

На основі аналізу методичних посібників, навчальних підручників, досвіду учителів-практиків, які висвітлюють рекомендації на Інтернет-ресурсах, можна виділити кілька підходів, за якими здійснюється навчання основ програмування:

- аналіз готових фрагментів коду;

- використання готових блоків програм («будівельних блоків»);

- використання простих конструкцій;

- розв’язання комплекстних завдань;

- комбінації підходів.

*1. Аналіз готових фрагментів коду.*

При підході до навчання програмування, заснованому на аналізі коду, учні читають і опановують логіку програмування, перш ніж писати свою власну. Цей підхід заснований на використанні псевдокоду, тому він не залежить від мови програмування.

Здатність пояснювати логіку програмування та код, мабуть, є необхідною умовою, хоча і не має на увазі здатності писати код. Такий підхід дозволяє розвивати навички розв’язання проблем, які можуть застосовуватися в багатьох галузях. Цей підхід може розчарувати учнів, які хочуть працювати з комп'ютерами, і може не підходити для ситуацій незалежного навчання через відсутність зворотного зв'язку. Підхід до аналізу коду аналогічний навчанню читання перед навчанням письма. Навчання читання розкриває правила граматики мови, що вивчається. Спроба вивчення мови таким чином повинна дозволити учню познайомитися з тим, як компоненти і конструкції мови комбінуються для створення сенсу. А вже потім учень може намагатися будувати свої конструкції.

Реалізація цього підходу може включати надання учням практичних вправ. Ці вправи можуть бути підготовлені із використанням прийнятного псевдокоду. Вправи можуть існувати тільки на папері або відображатися у відповідному середовищі розробки, якщо така існує для реалізації псевдокоду. Однак, як індивідуальний підхід, аналіз коду не вимагає, щоб учні взаємодіяли з кодом у середовищі комп'ютера.

Чотири нещодавних дослідження підтверджують використання підходу до аналізу коду. В Університеті Вікторії в Мельбурні Мілішевська і Тан (2007) [5] змінили дизайн одного з курсів з інформатики першого року, включивши в нього об'єктно-орієнтовану парадигму, але спочатку навчили структурного програмування. Новий підхід до викладання курсу включає вивчення прикладів добре написаного коду. Вони припускали, що студенти будуть учитися, наслідуючи приклади гарної практики.

Цей підхід також підтримують Келлинг і Розенберг (2001) [6], які є прихильниками вивчення коду для засвоєння стилів і використання ідіом. Кемпбелл і Болкер (2002) [7] погоджуються, виходячи з власного переконання, що студенти дізнаються більше, читаючи програми, написані досвідченими програмістами, ніж починаючи з написання власного коду. Проходження коду і читання коду, обидва елементи конструктивістського навчання комп'ютерного програмування, були використані Луї та ін. (2004) в їх дослідженні студентів, які вивчають більш слабке програмування [5]. Вони припускають, що цикл редагування, компіляції та виконання може виснажити терпіння і впевненість слабших учнів. Вони виступають за те, щоб слабші учні працювали з папером і олівцем.

Звичайно, разом ці результати можуть просто означати, що здатність писати код вимагає певних навичок у поясненні, але ця навичка може не породжувати здатності писати код. Хоча з точки зору учня навчання програмування без використання комп'ютера може здатися дивним, введення аналізу коду із самого початку має свої переваги. По-перше, цей підхід означає, що немає інструментів або середовищ, які потрібно освоїти при підготовці до вивчення логіки програмування. По-друге, можна використовувати будь-яку реалізовану мову програмування, але, можливо, найбільш зручним є відсутність певної мови взагалі. По-третє, аналіз, у першу чергу, дозволяє розвинути навички, пов'язані з налагодженням і трасуванням, щоб виправити або гарантувати поведінку програми. По-четверте, це також дає можливість ознайомити учнів з логікою простих базових алгоритмів, таких як лінійний пошук і бульбашкове сортування. Раннє знайомство з аналізом коду може допомогти підготувати учнів до іспитів, які включають питання, пов'язані з відстеженням або пробним запуском коду. Але в залежності від віку та здібностей учнів викладачеві може бути складно розробити набір значущих і зрозумілих інструкцій псевдокоду. Наприклад, чи слід використовувати слова або символи?

Спочатку аналіз може бути проблематичним для незалежного навчання через відсутність негайного зворотного зв'язку. Крім того, існує ймовірність того, що розуміння псевдокоду може не привести до навичок, які можна перенести на розуміння мови. Хоча підхід до аналізу коду доцільний не у всіх ситуаціях, він корисний для початківців програмістів. Його перевага полягає в тому, що він не вимагає певної мови програмування або комп'ютерного середовища. Він також сприяє розвитку навичок розв’язання проблем і логічного мислення, необхідних у багатьох галузях. Його використання може поліпшити здатність розуміти код, який становить частину базової основи, необхідної для полегшення написання програми.

*2. Використання готових блоків програм («будівельних блоків»)*

У підході до навчання програмування, заснованому на будівельних блоках, учні розвивають розуміння окремих частин, перш ніж об'єднувати їх для створення сенсу. Це вимагає набору інструментів розробки і певної мови програмування. Мовні конструкції вводяться і розуміються окремо, перед їх об'єднанням. Такий підхід не обмежується одномовними парадигмами і використовується в середовищах процедурних, функціональних і об'єктно-орієнтованих мов.

Підхід, заснований на будівельних блоках, має перевагу введення точно певного синтаксису мови і негайного зворотного зв'язку від інструменту перевірки синтаксису. На жаль, навчитися користуватися редактором коду, засобом перевірки синтаксису і, можливо, компілятором може бути складно для новачків. Крім того, той факт, що синтаксично правильний код часом не призводить до правильної логіки, викликає в учнів розчарування. Підхід, заснований на будівельних блоках, аналогічний навчанню говорити на окремих частинах мови перед їх об'єднанням спочатку в усній формі, а потім у письмовій. Учні зазвичай набувають усних мовних навичок раніше, ніж письмових. При спробі оволодіти мовою, окремі компоненти (іменники, дієслова, прикметники) деякою мірою розуміються, перш ніж з'єднати їх разом, щоб створити речення. Складний зміст будується на розумінні більш дрібних частин. Для цього підходу потрібен спеціально підібраний набір інструментів, що складається з мови програмування і середовища розробки. Передбачається, що вибрана мова і середовище відповідають можливостям учнів. Розуміння поведінки конструкцій буде покращено за рахунок написання конструкцій у відповідному середовищі розробки, в якій інструмент може виділяти синтаксичні помилки. Крім того, виконання цих окремих конструкцій можливе в середовищі розробки, де учні в змозі зрозуміти, можливо, за допомогою візуалізації, поведінку кожної конструкції.

Той же підхід можна використовувати з об'єктно-орієнтованою парадигмою, як показав Саяніемі і Ху (2006). [8]. Вони вважають за краще представляти поведінку змінних і керуючих структур до об'єктів. Цей підхід включає в себе переваги раннього введення синтаксису мови і раннього впровадження інструментів, які допомагають при програмуванні. Різноманітні конструкції й поняття мови можна навчати ізольовано. Подальше вивчення різних конструкцій може бути виконано шляхом демонстрації того, як змінювати їх поведінку, змінюючи їх синтаксис. Цей підхід також забезпечує виконання розроблених алгоритмів, що складаються з декількох блоків, за умови їх синтаксичної правильності. Це, звичайно, не означає, що будь-яка конкретна комбінація блоків буде логічно правильною. Крім того, освоєння інструментів, необхідних для відображення і перевірки поведінки блоків, надає учням інструменти і стратегії налагодження, які будуть потрібні для розробки реальних рішень.

Помилки в реалізації синтаксису мови програмування призвели до розчарування Аль-Імамі, Алізаде і Нура (2006) [9]. Щоб подолати це, вони розробили середовище, в якому використовуються шаблони конструкцій, в яких учні заповнюють прогалини. Наприклад, шаблон умовного блоку може бути представлений з правильною структурою синтаксису. Учневі потрібно тільки заповнити необхідне поле.

Підхід, заснований на будівельних блоках, може створити проблеми для учнів, які тільки починають своє навчання. Інтерактивне навчання вимагає оволодіння деякими складними інструментами одночасно з вивченням поведінки блоків. Але окремі блокові конструкції можуть не виконуватися або не мати сенсу для виконання, навіть, якщо синтаксис правильний. Наприклад, порожнє повторення або порожній умовний вираз синтаксично дозволені в деяких мовах, але не мають логічного сенсу. Освоєння поведінки окремих блоків може не перейти до конструктивної діяльності, необхідної для створення алгоритму, що виконує важливу задачу. Учень може пояснити і зрозуміти концепцію, але не зможе застосувати це розуміння при побудові алгоритму. Крім того, учням, можливо, буде важко зрозуміти відносини і взаємодії між блоками без контексту проблеми, яку потрібно вирішити. Проте, введення простого контексту проблеми може стати наступною дією.

Хоча використання підходу будівельних блоків часом є складним завданням як для учнів, так і для викладачів, воно передбачає раннє введення точно певного синтаксису мови і раннє впровадження набору інструментів для допомоги в написанні коду. Подання кожної мовної конструкції по черзі дозволяє природним чином досліджувати її, поки не стане зрозуміла її поведінка. Як тільки базова поведінка стає зрозумілою, можна проводити експерименти зі зміни синтаксису конструкцій для поглиблення навчання.

*3. Використання простих конструкцій.*

У підході до навчання програмування з використанням простих конструкцій, учні засвоюють стандартні фрази, використовуючи обмежений набір операцій та методів. Ці багаторазові блоки можуть зберігатися в наборі інструментів програміста для використання в більш складних рішеннях. Невеликі, вузькоспрямовані фрагменти можуть використовуватися для об'єднання конструкцій з метою створення одиниць, які корисні в більш великих рішеннях. Як і у випадку з будівельними блоками, підхід простих одиниць вимагає одночасного освоєння інструментів і конструкцій. Крім того, можна освоїти нові навички вирішення проблем.

У цьому підході конструкції об'єднуються в багаторазово використовувані блоки коду. Наприклад, набір конструкцій можна якимось чином об'єднати для сортування списку або знаходження максимального набору чисел. Невеликі, чітко визначені проблеми, такі як змусити візуальний об'єкт відскакувати, коли він торкається стіни лабіринту, можуть бути вирішені з використанням невеликої кількості конструкцій. Самі по собі ці агрегати можуть здатися банальними. Однак вони стають більш потужними, коли об'єднуються разом для вирішення більш великих проблем. Як і в разі підходу з використанням будівельних блоків, учням обов'язково потрібно середовище розробки, і, на жаль, вони зіткнуться з усіма проблемами, пов'язаними з ним.

Перевага цього підходу - можливість вводити невеликі, керовані і вузькоспрямовані контексти. Наприклад, при введенні конструкції повторення може бути використана концепція середнього значення. Побудова простого модуля, який знаходить середнє значення набору чисел, може бути легко записана з використанням конструкції повторення, але це також приведе до отримання корисного сегмента коду, який можна включити в більш складні програми. Об'єднання окремих конструкцій приводить до простих одиниць, комбінування простих одиниць приводить до більш складних одиниць. Як і в разі підходу з будівельними блоками, інтерактивне навчання вимагає оволодіння деякими складними інструментами одночасно з навчанням конструювання простих одиниць. Це часто є перешкодою для новачків. Більш того, оволодіння практичними навичками вирішення проблем, необхідних для розв’язання поставлених завдань, повинно відбуватися одночасно з вивченням інструментів і оволодінням поведінкою блоків. Також з використанням цього підходу розвивається корисна навичка декомпозиції - розбиття проблеми на невеликі розв'язні частини.

4. *Розв’язання комплексних завдань*

При повному системному підході до навчання програмування учні відразу ж занурюються у повне використання мовних конструкцій та інструментів. Цей підхід передбачає нетривіальне рішення проблеми. Концепції програмування і мовні конструкції вводяться в міру необхідності для вирішення проблеми. Хоча це може здатися нелогічним, деяких учнів мотивують реальні проблеми і їх повне рішення. Цей підхід, як і деякі з перелічених вище, вимагає одночасного володіння безліччю навичок та інструментів. Повноцінний системний підхід аналогічний вивченню мови методом занурення. При повному системному підході учні розробляють або допомагають розробити рішення нетривіальної проблеми.

У концепції програмування мовні конструкції вводяться тільки тоді, коли рішення проблеми вимагає їх застосування. Наприклад, гра в хрестики-нулики для двох може служити вступним завданням. Для цього учням потрібно продемонструвати навички декомпозиції (обробки введення з клавіатури, відображення результатів у будь-якому форматі, відстеження ходів (змінних / присвоєнь), визначення доступності позиції та ідентифікації виграшного ходу). Конструкції мови будуть введені тільки тоді, коли необхідно вирішити певну частину проблеми.

Дослідження довели успішність використання повного системного підходу, що включає вирішення проблем до навчання програмування. Duke та ін. (2000) застосовують цей підхід у своїх класах для початківців з Java. Щоб приховати деякі об'єктно-орієнтовані складності, вони надають набір індивідуальних класів для своїх учнів. В одному завданні учнів попросили реалізувати гру «4 в ряд». Конструкції, які повинні були освоїти студенти, полягали у використанні повторення і логічних виразів для підтримки внутрішньої логіки системи. Студенти повідомили, що такий підхід дозволяє їм ефективно вивчати Java.

Метою Нуутіла та ін. (2008) [10] є придбання навичок проєктування систем. Своїм учням вони ставлять такі завдання, як програмування руху робота у лабіринті. Очікується, що учні розберуться з проблемою, спроєктують заняття і розроблять алгоритми для створення ефективного вирішення. Цікаво, що фактичне створення рішення не завжди є частиною завдання, хоча в курс також входить практика використання реальних інструментів програмування.

Розповідь історій з Алісою, світ тривимірного візуального програмування, - це контекст, обраний Саттар і Лоренценом (2009) [9] для ознайомлення своїх учнів з програмуванням з використанням повносистемного підходу. Студентам надається сценарій розкадровки, і очікується, що вони розроблять код для анімації історії. Поняття (об'єкт, світ і сцена) і конструкції вводяться, коли це необхідно для просування анімації.

Інший приклад повносистемного підходу використовується Кемпбеллом і Болкером (2002) [11]. Вони використовують імерсивні техніки і просять своїх новачків прочитати і змінити симуляцію банкомату. Їх підхід відразу фокусується на інтерфейсах, архітектурі і дизайні. Синтаксис вибраної мови програмування розглядається тільки за необхідності. Вони визнають, що це непростий підхід, але він приділяє більше уваги дизайну і навичкам, ніж оволодінню синтаксисом.

Перевага полягає в тому, що учні можуть зіткнутися з реальними проблемами, для вирішення яких у них вже мають бути концептуальні моделі, наприклад торговий автомат або система телефонного білінгу. Крім того, вони можуть відчувати себе мотивованими і натхненними тим, що вчаться програмувати систему, яка представляє реальне життя. Наприклад, може знадобитися змінити код торгового автомата, щоб отримати правильну решту від монети 2 фунти стерлінгів. Якщо учень зможе внести цю зміну, то вся система стане більш функціональною й ефективною.

За визначенням, занурення або повний системний підхід вимагає одночасного оволодіння безліччю навичок. Інструменти середовища програмування, поведінка окремих блоків, взаємодія між блоками і налагодження повинні розглядатися разом. Це може деморалізувати деяких новачків. З точки зору вчителя, вибір завдань повинен бути добре продуманий, щоб виконати основні вимоги із засвоєння необхідних конструкцій і придбання навичок для роботи в середовищі програмування. Хоча це може здатися нелогічним, повний системний підхід може мотивувати деяких учнів. Розуміння того, як зміни в коді підвищують ефективність рішення, може швидко підштовхнути деяких учнів до подальшого дослідження. Реальні проблеми в уже відомому контексті спонукають розвинути більш досконалі навички вирішення проблем, такі як декомпозиція. Хоча, і за підходів з простими модулями і будівельними блоками необхідно освоїти певний синтаксис мови та інструменти розробки, цей підхід має ту перевагу, що з самого початку працює над нетривіальними рішеннями.

5. *Комбінації підходів*

Комбінування різних підходів забезпечує прогрес у становленні ефективного програміста. Різні набори інструментів дозволяють використовувати різні шляхи в залежності від можливостей і потреб учня. Здатність писати код вимагає певних навичок у відстеженні і поясненні коду[12].

Ефективні програмісти демонструють навички аналізу коду (відстеження і пояснення), розуміння поведінки блоків (відстеження і пояснення), побудови простих одиниць (написання) і комбінування простих одиниць для створення повних систем (написання). Передбачається, що, комбінуючи підходи до навчання програмування, учні зможуть продемонструвати прогрес у становленні ефективних програмістів, незалежно від їх віку та здібностей[13].

Лінійний підхід від будівельних блоків до простих одиниць і до повних систем – це, мабуть, найбільш очевидна лінія розвитку, оскільки вона являє собою поетапний рух у складності. Однак зворотний шлях також є типом прогресу і може сприяти придбанню або поглибленню знань учнів. Порядок підходів залежать тільки від відправної точки. Остаточне рішення про використовувані підходи і порядок їх викладання повинні бути обрані залежно від вимог курсу, віку учнів, середовища або мови програмування.

# Висновки до розділу 1

Програмування є найважливішою частиною інформатики і має тривалу історію. Існує багато навчальних методичних розробок і програм різних курсів програмування. Складною частиною для вчителя при навчанні програмуванню є те, що існують різні різні парадигми програмування і необхідно визначитися з вибором мови або системи програмування.

Актуальність парадигми об'єктно-орієнтованого програмування полягає в тому, що систематичне застосування засобів ООП дозволяє розробляти добре структуровані, надійні в експлуатації програмні системи, що досить просто модифікуються. Цим пояснюється вибір багатьох галузей професійної діяльності до об'єктно-орієнтованого підходу і об'єктно-орієнтованим мовам програмування. Тому ознайомлення з ним необхідне для учнів, орієнтованих на професію програміста.

В закладах освіті учні 5-9 класів вивчають тему «Основи алгоритмізації та програмування», щонайменше 30% від всього курсу інформатики.

На основі аналізу методичних посібників, навчальних підручників, досвіду учителів можна виділити п’ять підходів, за якими здійснюється навчання основ програмування: аналіз готових фрагментів коду; використання готових блоків програм; використання простих конструкцій; розв’язання комплексних завдань; комбінації підходів. Порядок підходів залежить тільки від відправної точки. Остаточне рішення про використовувані підходи і порядок їх викладання повинні бути обрані залежно від вимог курсу, віку учнів, середовища або мови програмування.

# РОЗДІЛ 2. РОЗРОБКА ПРАКТИЧНИХ ЗАВДАНЬ ДЛЯ НАВЧАННЯ ШКОЛЯРІВ ОСНОВ ПРОГРАМУВАННЯ МОВОЮ PYTHON В СЕРЕДОВИЩІ MINECRAFT

# 2.1. Специфіка середовища Minecraft для навчання програмування

Основи програмування - один з найскладніших розділів шкільного курсу інформатики. Одним із завдань навчання інформатики в основній школі є розвиток алгоритмічного мислення як засобу планування та організації діяльності. Тому основи алгоритмізації та програмування становлять одну із змістових ліній курсу інформатики, яка є наскрізною для всього курсу. Однією з вагомих проблем вивчення програмування інформатики є вибір середовища програмування. Доцільно розглядати процедурну об’єктно-орієнтовну парадигму програмування, прикладом якої мова Python.

Метою більшості навчальних ігор є допомогти учням засвоїти та зберегти нові знання. Підліткам важливо поєднувати ігрові технології з навчанням[14]. З цією метою у нагоді стане середовище Minecraft, яке на даний час має потужну освітню компоненту і надає різні можливості для впровадження у навчальний процес. Minecraft — гра з відкритим світом, де гравець може створювати, будувати та досліджувати. Вперше була випущена у 2009 році шведським програмістом Маркусом Перссоном під назвою «Miner». Гра була продана понад 144 мільйонами копій по всьому світу на всіх платформах, що робить її однією з найбільш популярних відеоігор усіх часів. Понад 1000 шкіл по всьому світу використовують Minecraft для проектів, які варіюються від навчання комп’ютерного програмування до відпрацювання математичних навичок, будування конструкції з різних матеріалів[15]. Завдяки своїй гнучкості гра легко підлаштовується під різні дисципліни. Гармонійне поєднання ігрових та навчальних технологій дає змогу учням сприймати складний навчальний матеріал у ігровій формі, що більш цікаво та комфортно для них. Але наразі бракує методичних розробок з навчання програмуванню, які би враховували інтереси школярів та були доступними для сприйняття. Отже, на наш погляд, розробка практичних завдань з навчання мови Python у середовищі Minecraft є цілком доречною і актуальною на даний час.

У 2013 році вийшла версія Minecraft для Raspberry Pi, яка була створена разом з бібліотекою Python. Вбудований в гру API дозволяє взаємодіяти з ігровим світом, тим самим навчаючи програмування. API складається з трьох стандартних окремих бібліотек, всі вони призначені для різної взаємодії з ігровим клієнтом. Початкова підтримка була надана лише для Python 2, але під час конференціі на PyconUK 2014 бібліотека була перенесена на Python 3 і створено py3minepi [16]. Нажаль, в процесі було порушено багато зворотної сумісності.

Можливість використовувати Minecraft з Python була дуже популярною, і незабаром був створений плагін RaspberryJuice для Minecraft Java Edition. Цей плагін є ключем з’єднання гри з бібліотеками як py3minepi. RaspberryJuice також розширив оригінальний API, додавши додаткові функції. Мартіном Он’Хонелом було створено біблотеку mcpi. Ця бібліотека підтримує Python 2 та Python 3 та Minecraft: Pi edition та плагін RaspberryJuice. Бібліотека вийшла в реліз на сайті PyPl в травні 2018 року. На сьогодні це найсучасніша бібліотека для роботи з Minecraft. Ця бібліотека пропонує шість різних класів для взаємодії з клієнтом гри.

Клас Minecraft - це основний клас для взаємодії з грою, який містить чотири підкласи : Camera, Entity, Events, Player. Block.py та event.py – додаткові бібліотеки для опису типа блока зі зберіганням констант кожного блоку та перевірки подій що сталися з блоком.

Оскільки мова програмування Python дуже гнучка, і гра Minecraft має відкрите з'єднання, було створено ще багато додаткових бібліотек [17] для спрощення деяких операція, або навпаки, для більш функціонального програмування.

Бібліотека minecraftstuff - стороння бібліотека для взаємодії з грою. Бібліотека забезпечує функції для малювання ліній, створення, переміщення та обертання фігур та модуль черепахи. У додатку В висвітлено характерні оператори бібліотеки.

# 2.2. Розробка тренувальних завдань для навчання школярів основ програмування в середовищі Minecraft

**Завдання №1. Перевірка уведених даних.**

Користувачеві пропонується ввести числові значення за допомогою input() в программі Python, які присвоються відповідній координаті гравця Minecraft. Проаналізувати різницю між командами postToChat() та print(). Перевірити введені данних.

Приклад вікна із завданням зображений на рис.2.1.а та рис.2.1.б

*Забезпечити*:

1. Перевірка уведеного значення Y.

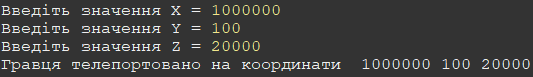


Рис. 2.1.а. Приклад виконання завдання

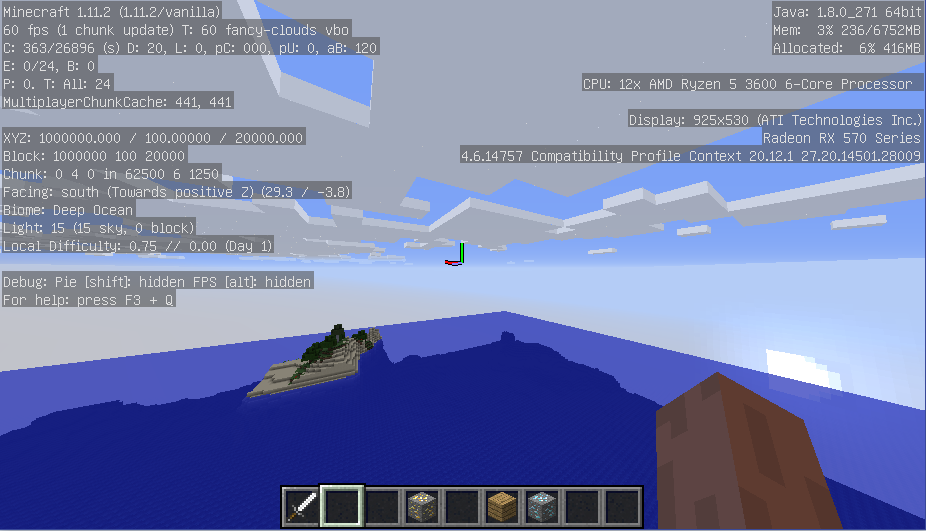


Рис. 2.1.б. Приклад виконання завдання в Minecraft

Наведемо приклад завершенної програми, яку повинен зробити користувач після виконання завдання №1:

import mcpi.minecraft as minecraft

mc = minecraft.Minecraft.create()

x = int(input("Введіть значення X = "))

y = int(input("Введіть значення Y = "))

while y<0:

print("Невірне значення Y")

y = int(input("Введіть значення Y = "))

z = int(input("Введіть значення Z = "))

mc.player.setPos(x,y,z)

mc.postToChat("Teleported to "+str(x)+" "+str(y)+" "+str(z))

print("Гравця телепортовано на координати ",x,y,z)

**Завдання №2. Знайти помилку у коді**

Дано фрагмент кода (рис 2.2), в якому потрібно знайти помилку, яка заважає роботі програми. Наприкінці потрібно отримати результат – повідомлення в консолі «Все працює». Код зображений на рис.2.2.

*Забезпечити*:

Оцінювання правильності виконаного завдання таким чином, щоб при старті правильного коду виводилося повідомлення в консоль Python з текстом «Все працює!».

*Рекомендації*:

Використати перевірку типу змінної за допомогою type().

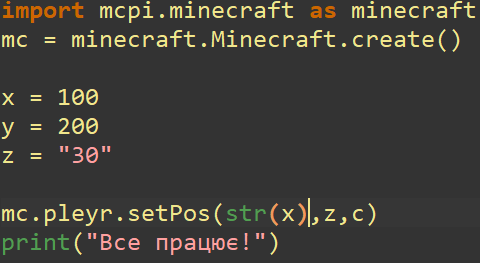


Рис. 2.2. Початкова форма завдання

**Завдання №3. Змінити позицію гравця на випадкове значення**

Створити програму, яка змінює позицію гравця на випадкове значення 3 рази з інтервалом 10 секунд. Приклад виконання завдання зображений на рис.2.3..

*Рекомендації*:

Для виконання завдання доцільно скористатися:

Модуль random, в якій знаходиться функція randint(a,b);

Модуль time, в якій знаходиться функція sleep(m).

**Завдання №4. Вивести координати гравця**

У наданому коді (рис. 2.4) потрібно додати команди для виведення координат гравця, отриманих за допомогою getTilePos().

*Забезпечити:*

Оцінювання правильності виконаного завдання таким чином, щоб при запуску правильного коду виводилося 3 повідомлення в грі з координатами гравця.

*Рекомендації*:

Скористатися командами команд getTilePos() та postToChat();

Використовувати зміну pos для створення списку координат гравця.

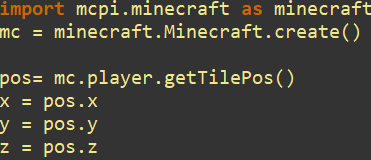
. 

Рис. 2.4. Приклад коду до завдання 4

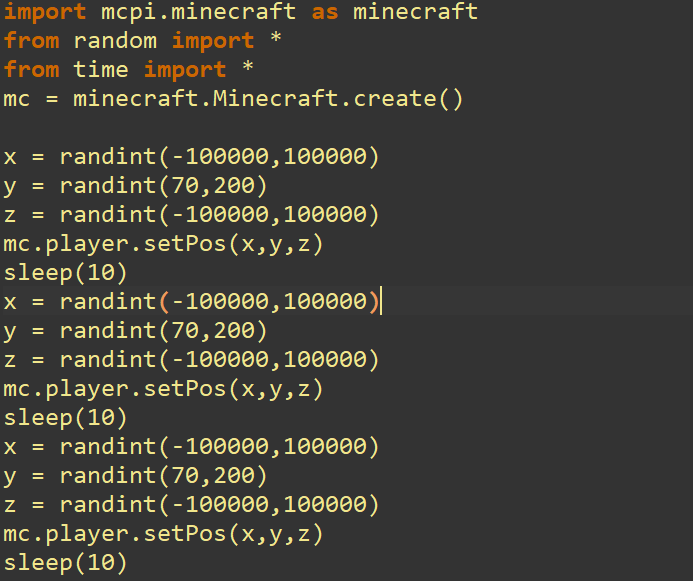


Рис. 2.3. Приклад програми до завдання 3

Наведемо приклад програми, яку отримає користувач після реалізації завдання 4:

import mcpi.minecraft as minecraft

mc = minecraft.Minecraft.create()

pos= mc.player.getTilePos()

x = pos.x

y = pos.y

z = pos.z

mc.postToChat(x)

mc.postToChat(y)

mc.postToChat(z)

**Завдання №5. Відстежити відстань, яку проходить гравець за 20 секунд**

Створити програму, за допомогою якої можна відстежити відстань, яку проходить гравець за 20 секунд вільної гри в Minecraft. Приклад виконаного завдання зображений на рис. 2.5..

*Виконати завдання таким чином:*

1. Створити зміну pos1 з координатами гравця при запуску програми.
2. Записати окремо кожну координату в зміну(X1,Y1,Z1).
3. Використати модуль time з функцією sleep().
4. Створити зміну pos2 з координатами гравця.
5. Записати окремо кожну координату в зміну(X2,Y2,Z2).
6. Створити нові зміні – різниці координат, від координати з pos1 відняти координату з pos2. Приклад : *distX=x2-x1;*
7. Вивести в гру кожну різницю координат.

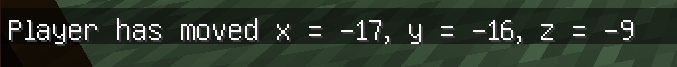


Рис. 2.5. Приклад виконаного завдання 5 в Minecraft

**Завдання №6 Змінити координати гравця**

Користувачеві пропонується ввести, яку координату потрібно змінити та значення, на яке змінити. Це значення присвоється відповідній координаті гравця Minecraft.

*Рекомендації*:

Висвітлення зміни координати гравця відносно свого місцезнаходження та різниці здійснюється за допомогою команд getTilePos() та getPos().

Приклад коду зі завданням:

import mcpi.minecraft as minecraft

mc = minecraft.Minecraft.create()

pos= mc.player.getPos()

x = pos.x

y = pos.y

z = pos.z

a = input()

if a=='x':

mc.player.setPos(x+int(input()),y,z)

elif a=='y':

mc.player.setPos(x,y+int(input()),z)

else:

mc.player.setPos(x,y,z+int(input()))

**Завдання №7. Створення нового блоку**

Користувачеві пропонується ввести ідентифікатор блоку в программі Python. Після цього, в світі Minecraft, поруч з гравцем, створиться блок згідно цього ідентифікатору. Переглянути параметри команди Приклад виконаного завдання зображений на рис. 2.6..

*Рекомендації*:

Використати список ID блоків.



Рис. 2.6. Приклад виконаного завдання 7 в Minecraft

**Завдання №8.Створення барєру**

Створити програму, яка створює навколо гравця бар’єр висотою 1 блок з дерева. Приклад виконаного завдання зображений на рис. 2.7.

*Виконати завдання таким чином:*

1. Створити зміну pos з координатами гравця.
2. Записати окремо кожну координату в зміну(X,Y,Z).
3. Додати 8 команд setBlock(x,y,z,5,0), де x,y,z – відповідно будуть змінюватись згідно позиції поруч з гравцем.

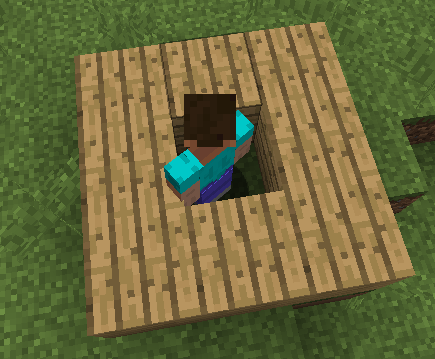


Рис. 2.7. Приклад виконаного завдання 8 в Minecraft

**Завдання №9. Побудова вежі з каменя**

Створити програму, яка створює поруч з гравцем вежу з камня висотою 10 блоків. Приклад виконаного завдання зображений на рис. 2.8.

*Виконати завдання таким чином:*

1. Створити зміну pos з координатами гравця.
2. Визначити цикл for, що виконує 10 ітерацій
3. Додати в цикл команду створення нового блока, в якій до координати Y слід додати змінну циклу



Рис. 2.8.б. Приклад виконаного завдання 9 в Minecraft

Приклад коду зі завданням:

import mcpi.minecraft as minecraft

mc = minecraft.Minecraft.create()

pos= mc.player.getPos()

for i in range(10):

mc.setBlock(pos.x,pos.y+i,pos.z,1,0)

**Завдання №10. Побудова вежі з випадкових блоків**

Створити програму, яка створює поруч з гравцем вежу висотою 30 випадкових блоків. Приклад виконаного завдання зображений на рис. 2.9.

*Рекомендації*:

Для виконання завдання доцільно скористатися циклом for.

*Додаткове завдання:*

Додати у цикл for команду sleep(1). Відстежити зміну значення змінних виконання програми в грі;

Змінити умову коду, додати можливість побудові вежі одного кольору. Запрограмувати вибір способу побудови користувачем.



Рис. 2.9. Приклад виконаного завдання 10 в Minecraft

**Завдання №11 Побудова куба**

Користувачеві пропонується ввести розмір куба та ідентифікатор блока в программі Python. Ці значення використаються в команді setBlocks(). Висвітлення використання команди setBlocks() та команди map(). Приклад вікна із завданням зображений на рис. 2.10.



Рис 2.10. Приклад виконаного завдання 11 в Minecraft

**Завдання №12 Додавання умови в код**

Дано код, який працює - з'єднання з Minecraft присутнє, в чаті гри друкується повідомлення «Code completed». Але більше нічого не з’являється. Створіть програму, в якій буде використовуватись перевірка на те, що гравець в повітрі чи на землі.

*Рекомендації*:

1. Використати команду minecraft.player.getPos() для отримання поточної позиції гравця.

2. Використати команду minecraft.getBlock() для отримання ідентифікатора блоку в якому знаходиться гравець.

3. Використати команду minecraft.postToChat() для друку повідомлення в чат гри.

**Завдання №13 Створення квадрату**

В бібліотеці mcpi є чудова команда setBlocks(), яка будує будь-який паралелепіпед від двух точок. Але за допомогою цієї команди неможливо створити пустий квадрат, в якого буде видно лише його сторони. Створіть програму, за допомогою якої можно позбавитись цієї проблеми.

*Виконати завдання таким чином:*

1. Відкрийте шаблон файлу «з’єднання з грою Minecraft».

2. Створіть зміну pos з координатами гравця при старті програми (використайте команду minecraft.player.getPos() для отримання значень).

3. Побудуйте квадрат за допомогою команди minecraft.setBlocks(). Використайте зміну pos як першу точку, в другій змініть лишу пару XY або YZ – для досягнення бажаного результату. Можна використати будь який матеріал.

4. Побудуйте ще один квадрат, але тепер з іншими точками. Зробіть зрушення по осям квадрату (значення в команді minecraft.setBlock() які змінюються користувачем) на один блок, в першій точці в плюс, в другій в мінус. В якості матеріалу обов'язково виберіть повітря.

5. Перевірте результат, у вас повинно вийти квадрат, в якого видно лише сторони. Якщо зовсім зник квадрат – поміняте місцями віднімання та додавання в другому квадраті з повітря.

# 2.3. Розробка ігрових ситуацій для школярів в середовищі Minecraft

**Завдання №1 «Неймовірні пригоди»**

Стів трохи занудьгував. Він не може знайти жодного цікавого містечка по всьому світу Minecraft. Допоможіть йому. Напишіть код, в якому потрібно телепортувати Стіва в цікаві місця кожну хвилину. Координати цих точок спочатку потрібно вам самим відшукати (рис. 2.11.), тому зупинимось на п’яти цікавих місцевостях. Програму можна зробити більш хаотичною, якщо використовувати бібліотеку random замість заданих координат.



Рис. 2.11. Приклад цікавої локації до завдання «Неймовірні пригоди»

**Завдання №2 «Супер сила»**

Всі мріють стати супергероєм, Алекс також не виняток. Але все що вона може робити, це створити випадково пожежу в своєму лісі. Давайте зробимо їй сюрприз. Створіть програму, яка підкине її на 100 блоків вгору, по 10 блоків кожну секунду. Після цього програма поставить під нею блок скла, щоб вона змогла побачити всі простори під собою (рис. 2.12.). Нехай відчує себе справжнім супергероєм!



Рис. 2.12. Приклад фінальної стадії завдання «Супер сила»

**Завдання №3 «Блоки за секунду»**

Всі знають швидкість вагонетки, коня, свині але ніхто не знає швидкість Стіва в Minecraft! Виправте ситуацію, напишіть код, який буде виводити результат підрахунків - скільки блоків пройшов Стів по прямій за одну хвилину. Результат зображенний на рис. 2.13.

Приклад виконаного завдання №3:

from mcpi.minecraft import Minecraft

import time

mc = Minecraft.create()

pos = mc.player.getPos()

time.sleep(60)

newPos = mc.player.gePos()

difference = abs(pos - newPos)

mc.postToChat("Speed - " + difference)



Рис. 2.13. Приклад виконанання завдання «Блоки за секунду»

**Завдання №4 «Безпека понад усе»**

Інколи трапляються ситуації, коли дещо пішло не по плану. Наприклад, у процесі шахтарства, Стів провалився в лаву, або під час подорожі, Алекс забула, що не вміє плавати та почала тонути в воді. Треба цього уникнути! Напишіть код, який буде завжди перевіряти, де знаходиться гравець. Якщо він впав на один блок в лаву, то піднімить його на два блоки верх, та поставте під ним блок, бажано непалаючий. Якщо ж гравець занурився в воду на три блоки, то підніміть його на чотири блоки, і також розмістіть, як в ситуації з лавою, під ним блок, можна будь-який. Результат зображенний на рис. 2.14.

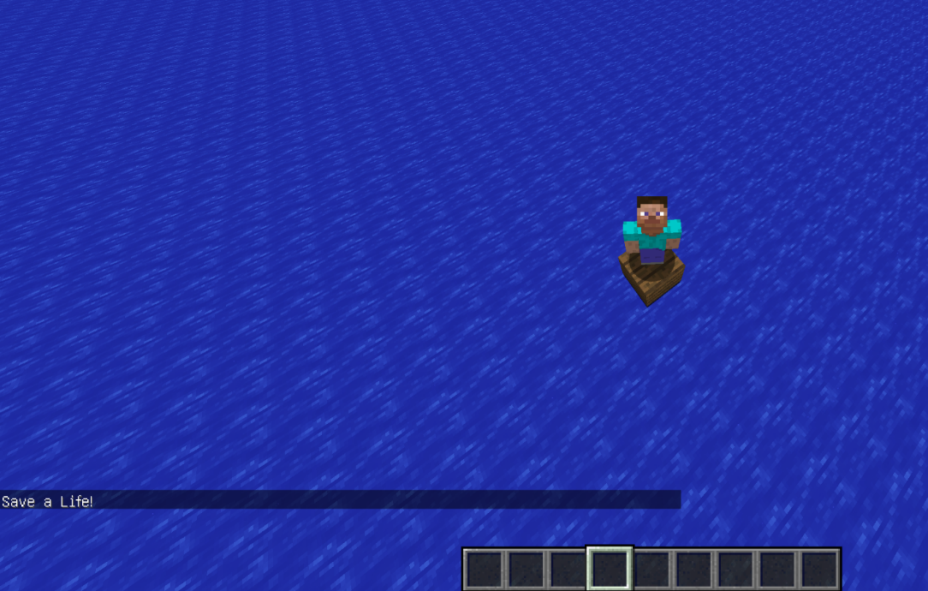


Рис. 2.14. Приклад виконаного завдання «Безпека понад усе»

**Завдання №6 «Копач»**

Стів втомився шукати золоту руду для своєї нової кирки. Щоб зберегти час від шахтарства, напишіть код, який дізнається, чи є в землі під гравцем будь-яка руда(залізо, золото, алмаз). Програма отримує поточні координати гравця і по одному перебирає блоки під ногами, перевіряючи, чи не є вони рудою. Якщо знайдеться задовільний блок або блоки, повідомте Стіва де він знаходиться. Результат зображенний на рис. 2.15.



Рис. 2.15. Приклад виконаного завдання «Копач»

**Завдання №6 «Шахтар-експерс»**

Всі гравці полюбляють мандрувати по світу Minecraft, незалежно де, в болотах, в шахтах, в глубинах океану. Але є одна прикра річ, відстань, інколи доводиться проходити значний шлях. Спробуймо це подолати! Створіть код, який буде будувати автоматичну колію для вагонеток. Так як, всі світи дуже привабливі і Стів хоче всюди побувати, вам потрібно поміркувати як зробити колію придатну для підземних та підводних пригод. Приклад виконання зображенно на рис. 2.16.

Фрагмент виконаного завдання №6:

def railLong(n):

p = mc.player.getPos()

for i in range(n):

mc.setBlock(p.x,p.y ,p.z+i,idRoad)#дорога

mc.setBlock(p.x,p.y+1 ,p.z+i,idRail) #колія

mc.setBlock(p.x,p.y+2 ,p.z+i,idAir)#повітря

mc.setBlock(p.x,p.y+3 ,p.z+i,idAir)#повітря

mc.setBlock(p.x,p.y+4 ,p.z+i,idAir)#повітря

mc.setBlock(p.x,p.y+5 ,p.z+i,idAir)#повітря

mc.setBlock(p.x,p.y+6 ,p.z+i,idBlock)#верхній люк

mc.setBlock(p.x-1,p.y ,p.z+i,idLamp)#стіна 1

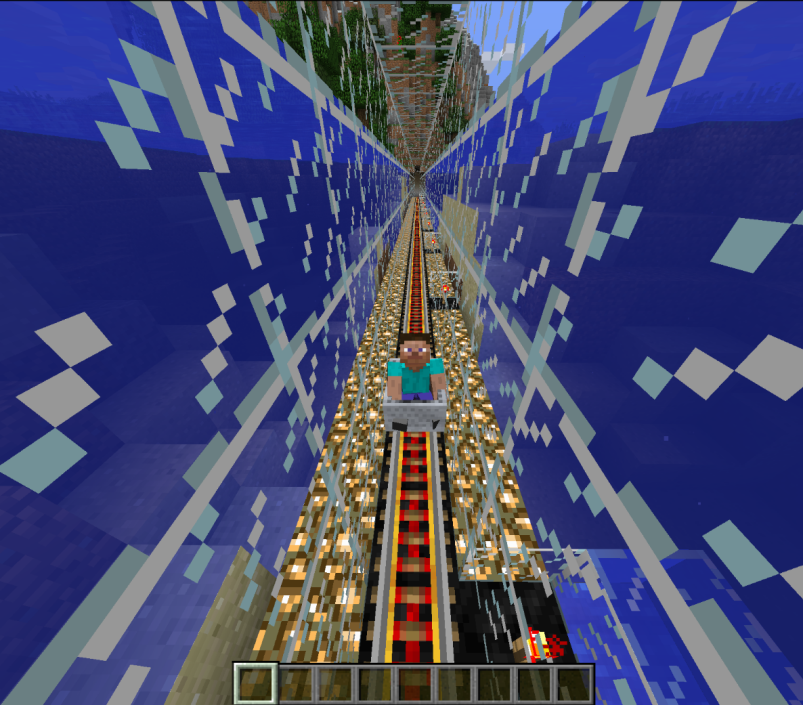


Рис. 2.16. Приклад виконаного завдання «Шахтар-експерс»

**Завдання №7 «Неприступна фортеця»**

Незабаром настане темрява! Стів та Алекс загубили свою домівку, а блоків для побудови нового будинку в них зовсім немає. Починають з’являтися зомбі, і без нашої допомоги вони не впораються. Створіть код, який побудує захисну оселю для Стіва та Алекс. Параметри оселі: будинок має бути розміром 5 x 5 x 5 блоків, матеріал стіни – обсидіан, мати вхід з дверима та принаймні 1 віконце для того, щоб дізнатись чи є небезпека зовні. Приклад виконання зображенно на рис. 2.17.

Фрагмент виконаного завдання №7:

from mcpi.minecraft import Minecraft

mc = Minecraft.create()

p=mc.player.getPos()

x=p.x

y=p.y

z=p.z

mc.setBlocks(x, y, z, x+5,y+5,z+5,5)

mc.setBlocks(x+1,y+1,z+1,x+4, y+4, z+4, 0)

mc.setBlocks(x, y+5,z, x+5,y+5,z+5,20)

mc.setBlocks(x, y, z, x+5,y, z+5,20)

mc.setBlock(x+2,y,z,1)

mc.setBlock(x+2,y+1,z,64,1)

mc.setBlock(x+2,y+2,z,64,8)



Рис.2.17. Приклад виконаного завдання «Неприступна фортеця»

**Завдання №8 «Фобія темряви»**

«Фобія темряви» Після останього візиту до лікара Стів точно знає, що в нього ніктофобія (страх перед тьмою). Та тепер усіма силами намагається уникати темряву будь-де, навіть в своїй домівці! Допоможіть, будь-ласка, Стіву з цією кепською ситуацією! Створіть код, який завжди буде будувати на землі попереду Стіва факел. Якщо так трапиться, що Стів виявиться в повітрі – побудуйте замість факелу блок лампи. Ні темряві!

# Висновки до розділу 2

У ході роботи в розділи було детально висвітлено шляхи вивчення мови Python з залученням середовища Minecraft. Середовище Minecraft має потужну освітню компоненту і надає різні можливості для впровадження у навчальний процес. Minecraft — гра з відкритим світом, де гравець може створювати, будувати та досліджувати. Середовище підлаштовується під різні дисципліни, а гармонійне поєднання ігрових та навчальних технологій дає змогу учням сприймати складний навчальний матеріал у ігровій формі, що більш цікаво та комфортно для них.

Можливість використовувати Minecraft з мовою програмування Python вбачається корисним при навчання школярів програмування. Minecraft підтримує , і на сьогодні це найсучасніша бібліотека для роботи з Minecraft.

Було розроблено комплект тренувальних завдань та ігрових ситуацій для навчання школярів 7-9 класів основ об'єктно-орієнтованого програмування мовою Python у середовищі Minecraft. Запропоновано й детально висвітлено 13 тренувальних завдань та 8 ігрових ситуацій. Кожне із завдань сформульовано у такий спосіб, щоб учневі було цікаво вирішувати завдання. Крім того, кожне завдання супроводжується прикладом його виконання, дано зображення з копією екрана. До окремих тренувальних завдань надані рекомендації до виконання, які включають перелік основних операторів, що знадобляться для розв’язання. При цьому слід зазначити, що наведений код не є єдиним правильним варіантом, і може бути видозміненим.

# РОЗДІЛ 3. ЕКСПЕРИМЕНТАЛЬНЕ ВПРОВАДЖЕННЯ ДИДАКТИЧНИХ МАТЕРІАЛІВ В ПОЗАШКІЛЬНОМУ НАВЧАННІ

# 3.1. Організація та хід упровадження розроблених практичних завдань

Невід’ємною частиною впровадження дидактичних матеріалів є організація завдань, їх перевірка згідно поставлених вимог, які були визначені на початку їх написання.

Всі завдання були протестовані у практичній роботі в позашкільній підготовки школярів. З метою підсумовування тестування було створено тест «Навчання основ програмування за допомогою середовища Minecraft» на платформі всесвітньої компаніі Google, яка має назву «Google Форми». Форма подана на рис. 3.1..

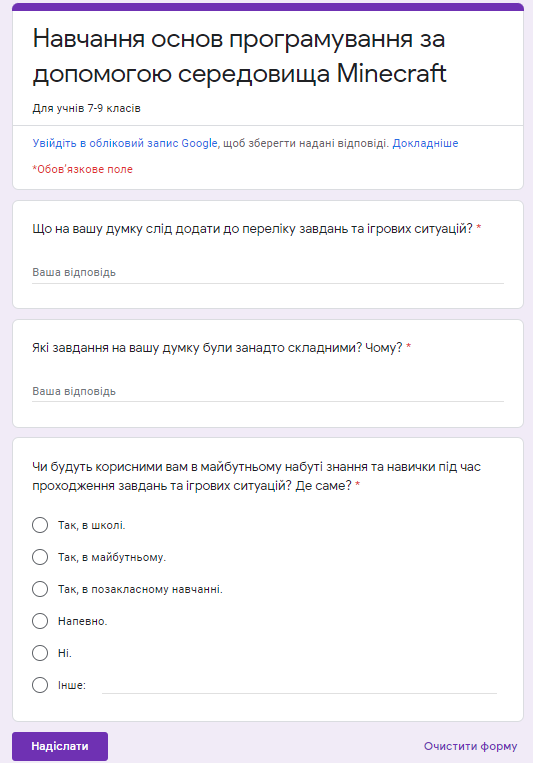


Рис. 3.1. Форма для перевірки доцільності запровадження розроблених завдань

Вибірка складала 20 учнів. Кожному школяру давався документ з практичними вправами та ігровими ситуаціями в порядку зростання складності. Після, через 48 годин, кожен учень отримував посилання на форму.

Статистика відповідей на питання 3 «Чи будуть корисними вам в майбутньому набуті знання та навички під час проходження завдань та ігрових ситуацій? Де саме?» зображена на рис. 3.2..

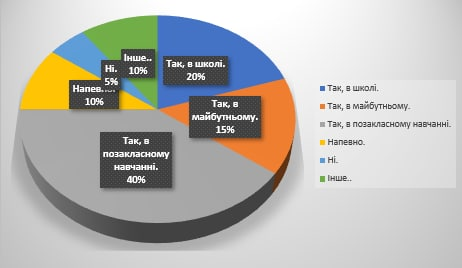


Рис. 3.2. Статистика відповідей на питання 3

Як видно з діаграми:

* Так, в позакласному навчанні – 40%,
* Так, в майбутньому – 15%,
* Так, в школі – 20%,
* Інше – 10%,
* Ні – 5%,
* Напевно – 10%.

# 3.2. Аналіз результатів експерименту

З метою дослідження стану навчання школярів 7-9 класів основ об'єктно-орієнтованого програмування мовою Python було проведено опитування на тему «Програмування мовою Python» серед учнів загальноосвітніх закладів 7-9 класів та окремо серед учнів, які займалися позашкільним навчанням з теми «Навчання мови Python у середовищі Minecraft».

Опитування було зроблено засобами «Google Форми». Вибірка школярів загальноосвітніх закладів складала 40 осіб, а вибірка учнів індивідуальной підготовки 20 осіб. Кожному школяру відправлялось посилання на форму. Фрагмент форми зображено на рис. 3.3.

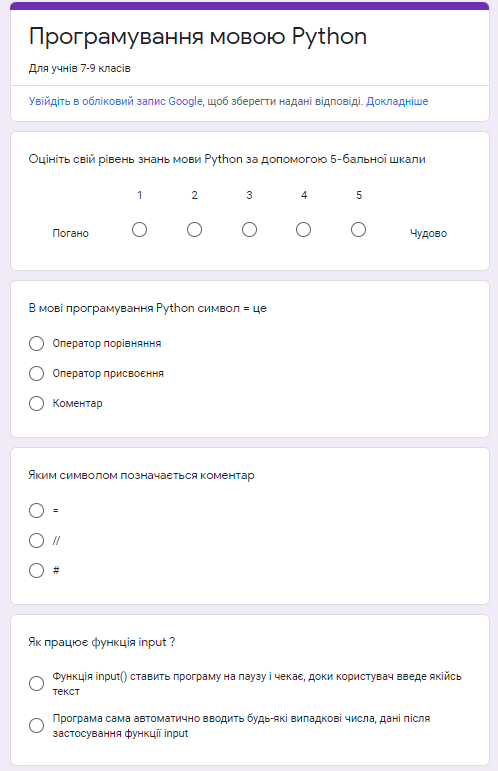


Рис. 3.3. Фрагмент форми на тему «Програмування мовою Python»

Наведемо декілька прикладів відповідей учнів на питання з данної форми. Переважна більшіть школярів добре розуміють як працює input та print, але є винятки (рис. 3.4.). Немає різниці в відповідях учнів різних підготовок в програмуванні.

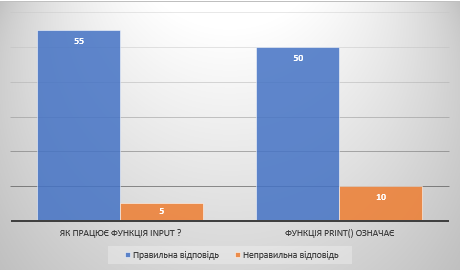


Рис. 3.4. Статистика правильних та неправильних відповідей на питання

Але якщо взяти до уваги складні питання, то тут з'являється перевага учнів з індивідуальної підготовки. Наприклад завдання «for i in range (2):, Які значення має і?» мало низький рівень правильних відповідей у учнів загальноосвітніх закладів (рис. 3.5.).

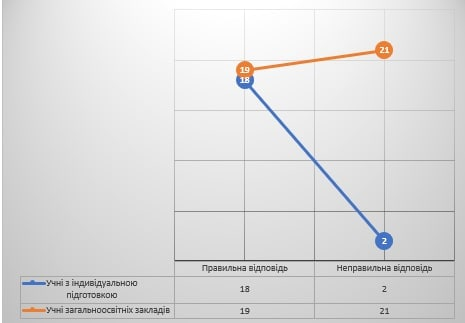


Рис. 3.5. Статистика правильних та неправильних відповідей на питання

Як можна побачити з таблиці результатів, відсоток правильних відповідей набагато більший в учнів з позашкільним навчанням, що свідчить про те, що специфіка програмування мовою Python розкривається на недостатньому рівні в школах.

# Висновки до розділу 3

Розроблені у ІІ розділі практичні завдання для учнів базової школи були впроваджені в освітню практику – в процесі позашкільної підготовки школярів. В експериментальному впровадженні завдань взяли участь 20 школярів базової школи, які пройшли опитування, а також взяли участь у експериментальній роботі. Склад контрольної групи утворювали 40 осіб закладів загальноосвітньої школи.

Після експериментальної роботи учні проходили опитування за допомогою Google Форм. Учнів запитували щодо програмного матеріалу, наприклад, щодо оформлення циклів, використання операторів уведення, окремих операцій у мові програмування Python.

Результати свідчать, що в основи програмування на мові Python добре знають переважна кількість учнів. Проте, питання підвищеної складності викликають труднощі у учнів з базою рівня загальноосвітніх закладів 7-9 класів.

# ЗАГАЛЬНІ ВИСНОВКИ

Навчання будь-якого предмета в школі повинно бути організоване таким чином, щоб учням було цікаво на уроках, щоб вони самі прагнули отримувати нові знання і вчителю не доводилося б змушувати їх засвоювати навчальний матеріал. Предмет «Інформатика», з одного боку, знаходиться в більш вигідному становищі, ніж інші шкільні предмети, тому що використання на уроках комп’ютера саме по собі вже привабливе для учнів. Але, з іншого боку, багато учнів пов'язують комп'ютер виключно з можливістю пограти в ігри, як, наприклад, Minecraft. Тому дітей потрібно навчити не тільки грати в ігри, але й вміти їх створювати за допомогою мов програмування.

В результаті аналізу літератури з теми дослідження виявлено великий спектр можливостей ігрових елементів, які можна застосувати в освітньому процесі. Продемонстровані перспективність і актуальність використання різноманітних платформ, ігор та рушіїв гри на уроках інформатики, виявлено особливості шкільних занять з використанням ігрових елементів, виділені переваги їх використання. У ході роботи було розроблено комплект тренувальних завдань та ігрових ситуацій для навчання школярів 7-9 класів основ об'єктно-орієнтованого програмування мовою Python у середовищі Minecraft. Запропоновано й детально висвітлено 13 тренувальних завдань та 8 ігрових ситуацій. Всі завдання є авторськими і апробованими у практичній роботі з індивідуальної підготовки школярів.

Використання сучасних технологій у навчальному процесі дозволяє підтримувати високий рівень мотивації учнів, запропонувати учням велику кількість готових, старанно відібраних, відповідним чином організованих знань, розвивати інтелектуальні, творчі здібності учнів і сприяти розвитку комунікативних аспектів, навичок самостійного створення програм.

**АНОТАЦІЯ**

**Сівочка Ігор Глібович**. Навчання школярів основ програмування за допомогою середовища Minecraft. Кер.: д. пед. н., професор, завідувач кафедри інформатики Н.В. Олефіренко. – Х.: ХНПУ імені Г. С. Сковороди, 2021. – 50 с.

У кваліфікаційній роботі розглянуто об’єктно-орієнтоване програмування, його різновиди та підходи до навчання програмування. Особливості та місця вивчення основ об’єктно-орієнтованого програмування у шкільному курсі інформатики. Розроблено та впроваджено у освітню практику комплект практичних завдань для навчання основ програмування мовою Python в середовищі Minecraft.

Кваліфікаційна робота: 50 c., 1 табл., 23 рис., 2 дод, 17 джерел.

ОСНОВИ ПРОГРАМУВАННЯ, ОБ’ЄКТНО-ОРІЄНТОВАНЕ ПРОГРАМУВАННЯ, PYTHON, MINECRAFT, РОЗРОБКА ПРАКТИЧНИХ ЗАВДАНЬ.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

* + - 1. В.В. Горборуков, О.Є. Стрижак, О.В. Франчук, В.Б. Шаповалов Онтологічне представлення задачі ранжування альтернатив URL: <http://ekmair.ukma.edu.ua/bitstream/handle/123456789/17887/Ontolohichne_predstavlennia_zadachi_ranzhuvannia_alternatyv.pdf?sequence=1&isAllowed=y>
      2. Ю. К. Бабанського Методы обучения в современной общеобразовательной школе. Укитувчи. : Б-ка учителя по общ. пробл. теории обучения и воспитания, 1990. 229 c.
      3. Офіційний сайт міністерства освіти і науки України, Навчальні програми для учнів 5-9 класів. URL: <https://mon.gov.ua/ua/osvita/zagalna-serednya-osvita/navchalni-programi/navchalni-programi-5-9-klas>
      4. Офіційний сайт міністерства освіти і науки України; Програма курсу інформатика 5-9 класи загальноосвітніх навчальних закладів URL: <https://mon.gov.ua/storage/app/media/zagalna%20serednya/programy-5-9-klas/onovlennya-12-2017/programa-informatika-5-9-traven-2015.pdf>
      5. Linxiao Ma. Investigating and Improving Novice Programmers’ Mental Models of Programming Concepts/ с.2. URL: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.721.8479&rep=rep1&type=pdf>
      6. Kolling M., Rosenberg J. Guidelines for Teaching Object Orientation with Java. URL: <https://kar.kent.ac.uk/13607/1/guidelines_for_teaching_object_kolling.pdf>
      7. Lui A., Kwan R., Poon M., Cheung Y. Saving weak programming students: applying constructivism in a first programming course. URL: <https://dl.acm.org/doi/10.1145/1024338.1024376>
      8. Sajaniemi J. and other Roles of variables in three programming paradigms. URL: https://www.researchgate.net/publication/228623601\_ Roles\_of\_variables\_in\_three\_programming\_paradigms
      9. Sattar A., Lorenzen T. Mathematics and computer science faculty publications. URL: <https://vc.bridgew.edu/math_compsci_fac/16/>
      10. Nuutila E., Törmä S., Kinnunen P., Malmi L. Learning Programming with the PBL Method — Experiences on PBL Cases and Tutoring. URL: <https://www.researchgate.net/publication/226134435_Learning_Programming_with_the_PBL_Method_-_Experiences_on_PBL_Cases_and_Tutoring>
      11. UAE schools need to offer the 'language of the future' in regular lessons. URL: https://www.thenationalnews.com/uae/uae-schools-need-to-offer-the-language-of-the-future-in-regular-lessons-1.790391
      12. Campbell W., Bolker E.. Teaching programming by immersion, reading and writing. URL: http://fie2012.fie-conference.org/sites/fie2012.fie-conference.org/history/fie2002/papers/1637.pdf
      13. Miliszewska I. Befriending Computer Programming: A Proposed Approach to Teaching Introductory Programming URL: <https://www.researchgate.net/publication/240948260_Befriending_Computer_Programming_A_Proposed_Approach_to_Teaching_Introductory_Programming>
      14. Scott T. Leutenegger, Edgington J. A games first approach to teaching introductory programming. URL: https://www.researchgate.net/publication/221538278\_A\_games\_first\_approach\_to\_teaching\_introductory\_programming
      15. Minecraft. Education Edition. URL: https://education.minecraft.net/how-it-works/why-minecraft
      16. Glass K., Hickman G. Official GitHub py3minepi project URL: <https://github.com/py3minepi/py3minepi>
      17. O’Hanlon M. Minecraft - Stuff Library URL: <https://minecraft-stuff.readthedocs.io/en/latest/>

# ДОДАТКИ

# ДОДАТОК А

**ТЕЗИ ПУБЛІКАЦІЇ НАУКОВО-МЕТОДИЧНОЇ КОНФЕРЕНЦІЇ «НАУМОВСЬКІ ЧИТАННЯ» ТА СЕРТИФІКАТ УЧАСТНИКА**

ПІДГОТОВКА ЗАВДАНЬ ДЛЯ НАВЧАННЯ ШКОЛЯРІВ ОСНОВ ПРОГРАМУВАННЯ ЗАСОБАМИ СЕРЕДОВИЩА MINECRAFT

Для сучасної молоді інтерес до програмування часто зумовлює вибір майбутньої професії, тому важливо розвивати у школярів інтерес до цієї сфери. Це можна робити за допомогою впровадження в освітній процес різних ігрових елементів. Складно уявити, що ігри можна використовувати в системі освіти, але існує багато освітніх продуктів, пов’язаних з іграми. Одним з них є Scratch, який навчає дітей програмувати за допомогою простої графічної мови та блоків коду. CodeMonkey навчає користувачів основам програмування на Ruby за допомогою інтерактивного підручника. Tynker має гейміфіковані уроки з робототехніки з веселими завданнями для дітей. Kodu також використовує ігровий процес для навчання таким концепціям програмування, як цикли та послідовності. Codecombat містить навчальні посібники з синтаксису JavaScript або Python через битви гравців зі штучним інтелектом.

Метою більшості навчальних ігор є допомогти учням засвоїти та зберегти нові знання. Підліткам важливо поєднувати ігрові технології з навчанням [1]. З цією метою у нагоді стане середовище Minecraft, яке на даний час має потужну освітню компоненту і надає різні можливості для впровадження у навчальний процес. Minecraft — гра з відкритим світом, де гравець може створювати, будувати та досліджувати. Вперше була випущена у 2009 році шведським програмістом Маркусом Перссоном під назвою «Miner». Гра була продана понад 144 мільйонами копій по всьому світу на всіх платформах, що робить її однією з найбільш популярних відеоігор усіх часів. Понад 1000 шкіл по всьому світу використовують Minecraft для проектів, які варіюються від навчання комп’ютерного програмування до відпрацювання математичних навичок, будування конструкції з різних матеріалів[2]. Завдяки своїй гнучкості гра легко підлаштовується під різні дисципліни. Гармонійне поєднання ігрових та навчальних технологій дає змогу учням сприймати складний навчальний матеріал у ігровій формі, що більш цікаво та комфортно для них. Але наразі бракує методичних розробок з навчання програмуванню, які би враховували інтереси школярів та були доступними для сприйняття. Отже, на наш погляд, розробка практичних завдань з навчання мови Python у середовищі Minecraft є цілком доречною і актуальною на даний час.

У 2013 році для Raspberry Pi була випущена версія Minecraft. Ця версія містила Python як мову програмування та внутрішній API, який дозволяв користувачам взаємодіяти зі світом гри, навчаючи їх елементарним навичкам програмування. API складається з трьох різних бібліотек: перша призначена для взаємодії з гравцями (наприклад, показ повідомлень), друга використовується для маніпулювання блоками та предметами в ігровому світі (наприклад, створення матеріалів або створення нових істот), остання для редагування світу в Minecraft.

Розроблені завдання будуть поділятись на два типи – тренувальні завдання для відпрацювання основ програмування мовою Python в середовищі Minecraft та ігрові ситуації креативного характеру, вирішення яких буде розроблене мовою Python в середовищі Minecraft.

Наведемо приклади розробленних завдань:

Тренувальне завдання №1. «Створення квадрату». В бібліотеці mcpi є чудова команда setBlocks(), яка будує будь-який паралелепіпед від двух точок. Але за допомогою цієї команди неможливо створити пустий квадрат, в якого буде видно лише його сторони. Створіть програму, за допомогою якої можно позбавитись цієї проблеми.

Виконати завдання таким чином: 1. Відкрийте шаблон файлу «з’єднання з грою Minecraft». 2. Створіть зміну pos з координатами гравця при старті програми (використайте команду minecraft.player.getPos() для отримання значень). 3. Побудуйте квадрат за допомогою команди minecraft.setBlocks(). Використайте зміну pos як першу точку, в другій змініть лишу пару XY або YZ – для досягнення бажаного результату. Можна використати будь який матеріал. 4. Побудуйте ще один квадрат, але тепер з іншими точками. Зробіть зрушення по осям квадрату (значення в команді minecraft.setBlock() які змінюються користувачем) на один блок, в першій точці в плюс, в другій в мінус. В якості матеріалу обов'язково виберіть повітря. 5. Перевірте результат, у вас повинно вийти квадрат, в якого видно лише сторони. Якщо зовсім зник квадрат – поміняте місцями віднімання та додавання в другому квадраті з повітря.

Тренувальне завдання №2. «Додавання умови в код». Дано код, який працює - з'єднання з Minecraft присутнє, в чаті гри друкується повідомлення «Code completed». Але більше нічого не з’являється. Створіть програму, в якій буде використовуватись перевірка на те, що гравець в повітрі чи на землі.

Рекомендації: 1. Використати команду minecraft.player.getPos() для отримання поточної позиції гравця. 2. Використати команду minecraft.getBlock() для отримання ідентифікатора блоку в якому знаходиться гравець. 3. Використати команду minecraft.postToChat() для друку повідомлення в чат гри.

Ігрова ситуація №1. «Фобія темряви» Після останього візиту до лікара Стів точно знає, що в нього ніктофобія (страх перед тьмою). Та тепер усіма силами намагається уникати темряву будь-де, навіть в своїй домівці! Допоможіть, будь-ласка, Стіву з цією кепською ситуацією! Створіть код, який завжди буде будувати на землі попереду Стіва факел. Якщо так трапиться, що Стів виявиться в повітрі – побудуйте замість факелу блок лампи. Ні темряві!

В результаті аналізу літератури з теми дослідження виявлено великий спектр можливостей ігрових елементів, які можна застосувати в освітньому процесі. Було представлено перспектива і актуальність використання різних платформ, ігор на уроках інформатики та факультативних заняттях. У ході роботи було підготовлено комплект тренувальних завдань та ігрових ситуацій для навчання школярів основ об'єктно-орієнтованого програмування мовою Python у середовищі Minecraft. Запроповновано й висвітлено 2 тренувальних завдання та 1 ігрову ситуацію. Наведені вище завдання оригінальні та апробовані в практичній роботі з індивідуальної підготовки школярів.

****

**ДОДАТОК Б**

**ТЕЗИ ПУБЛІКАЦІЇ МІЖНАРОДНОЇ НАУКОВО-ПРАКТИЧНОЇ КОНФЕРЕНЦІЇ «СУЧАСНА НАУКА: ПРОБЛЕМИ ТА ПЕРСПЕКТИВИ»**

РОЗРОБКА ЗАВДАНЬ ДЛЯ НАВЧАННЯ ШКОЛЯРІВ ОСНОВ ПРОГРАМУВАННЯ МОВОЮ PYTHON В СЕРЕДОВИЩІ MINECRAFT

Одним із важливих розділів шкільного курсу інформатики є «Моделювання. Основи алгоритмізації та програмування», а розвиток алгоритмічного мислення є надзвичайно важливим для школярів в ракурсі опанування умінь й навичок планування та організації діяльності. Однією з вагомих проблем вивчення програмування в курсі інформатики є вибір середовища програмування.

На даний час багато вчителів віддають перевагу вивченню мови Python, починаючи із 7 класу. Слід зазначити, що учні базової школи (7-9 класи) потребують особливих підходів до навчання програмування. На нашу думку, найбільш зручним і мотивуючим підходом у навчанні програмування семикласників є використання ігрових технологій.

Підліткам важливо поєднувати ігрові технології з навчанням [1]. У цьому зв’язку для навчання школярів програмування пропонуємо звернути увагу на середовище Minecraft, яке має потужну освітню компоненту і надає різні можливості для впровадження у навчальний процес. Minecraft — це гра з відкритим світом, де гравець може створювати, будувати та досліджувати.

Середовище підлаштовується під різні дисципліни, а гармонійне поєднання ігрових та навчальних технологій дає змогу учням сприймати складний навчальний матеріал у ігровій формі, що більш цікаво та комфортно для них. Незважаючи на високий розвиток середовища, слід відзначити недостатність методичних розробок для навчання школярів, які б враховували інтереси школярів і були б доступними для сприйняття.

У 2013 році для Raspberry Pi була випущена версія Minecraft[2]. Ця версія містила Python як мову програмування та внутрішній API, який дозволяв користувачам взаємодіяти зі світом гри, навчаючи їх елементарним навичкам програмування. API складається з трьох різних бібліотек: перша призначена для взаємодії з гравцями (наприклад, показ повідомлень), друга використовується для маніпулювання блоками та предметами в ігровому світі (наприклад, створення матеріалів або створення нових істот), остання для редагування світу в Minecraft [3].

Представимо розроблені практичні завдання для навчання підлітків мови Python в середовищі Minecraft. Розроблений комплект містить завдання двох типів — тренувальні завдання для відпрацювання основних умінь програмування мовою Python в середовищі Minecraft та ігрові ситуації креативного характеру.

Наведемо приклади розроблених завдань:

Тренувальне завдання 1. Побудова вежі з каменя. Створити програму, яка створює поруч з гравцем вежу з каменя висотою 10 блоків.

Завдання передбачає виконання у такий спосіб: створити змінну pos з координатами гравця; визначити цикл for, що виконує 10 ітерацій; додати в цикл команду створення нового блока, в якій до координати Y слід додати змінну циклу.

Тренувальне завдання 2. Розробити програму, яка створює поруч з гравцем вежу висотою 30 випадкових блоків.

Додаткові завдання: додати у цикл for команду sleep(1). Відстежити зміну значень змінних; змінити умову коду, додати можливість побудові вежі одного кольору. Запрограмувати вибір способу побудови користувачем.

Ігрові ситуації сформульовані у вигляді завдань для учасників (Стіва, Алекса тощо), які є героями в середовищі Minecraft і виконавцями коду. З одного боку, ситуації пов’язані між собою сюжетною лінією, а з іншого боку можуть бути розв’язані незалежно одна від одної.

Ігрова ситуація 1 «Копач». Стів втомився шукати золоту руду для своєї нової кирки. Щоб зберегти час від шахтарства, напишіть код, який дізнається, чи є в землі під гравцем будь-яка руда (залізо, золото, алмаз). Програма отримує поточні координати гравця і по одному перебирає блоки під ногами, перевіряючи, чи не є вони рудою. Якщо знайдеться потрібний блок або блоки, повідомте Стіва, де він знаходиться.

Ігрова ситуація 2. «Неприступна фортеця». Незабаром настане темрява! Стів та Алекс загубили свою домівку, а блоків для побудови нового будинку в них зовсім немає. Починають з’являтися зомбі, і без нашої допомоги вони не впораються. Створіть код, який побудує захисну оселю для Стіва та Алекс. Параметри оселі: будинок має бути розміром 5 x 5 x 5 блоків, матеріал стіни – обсидіан, мати вхід з дверима та принаймні 1 віконце для того, щоб дізнатись чи є небезпека зовні.

Таким чином, середовище Minecraft надає широкі можливості для навчання школярів програмування мовою Python. Розроблено комплект практичних завдань, що включає 15 тренувальних завдань та 7 ігрових ситуацій, окремі з яких наведено. Всі розроблені завдання є авторськими та апробованими в практичній роботі з індивідуальної підготовки школярів.

**ДОДАТОК В**

**ОПЕРАТОРИ БІБЛІОТЕК MINECRAFT ДЛЯ НАВЧАННЯ ШКОЛЯРІВ ПРОГРАМУВАННЮ МОВОЮ PYTHON**

1. MinecraftDrawing API - можна використовувати для створення ліній, кіл, сфер та граней.

*Class minecraftstuff.MinecraftDrawing(mc)*

*MinecraftDrawing* - клас корисних функцій малювання

|  |  |
| --- | --- |
| Параметри: | mc (mcpi.minecraft.Minecraft) – Об'єкт Minecraft, який пов'язаний зі світом. |

*drawCircle(x0, y0, z, radius, blockType, blockData=0)* - малює коло в площині Y (тобто вертикально)

|  |  |
| --- | --- |
| Параметри: | x0 (int) – Х положення центру кола.  y0 (int) – У положення центру кола.  z0 (int) – Z положення центру кола.  radius (int) – Радіус кола.  blockType (int) – Ідентифікатор блоку.  blockData (int) – Значення даних блоку за замовчуванням дорівнює 0. |

*drawFace(vertices, filled, blockType, blockData=0)* - малює грань, коли передається колекція вершин, що складають багатогранник

|  |  |
| --- | --- |
| Параметри: | vertices (list) – Список точок, переданих як будь-який minecraftstuff.Points об'єкт або як список mcpi.minecraft.Vec3 об'єктів.  filled (boolean) – Якщо True наповнює грань блоками.  blockType (int) – Ідентифікатор блоку, далі “ID”.  blockData (int) – Значення даних блоку, за замовчуванням дорівнює 0, далі “Значення даних”. |

*drawHollowSphere(x1, y1, z1, radius, blockType, blockData=0)* - малює порожню сферу навколо точки до радіуса

|  |  |
| --- | --- |
| Параметри: | x1 (int) – Х положення центру сфери.  y1 (int) – У положення центру сфери.  z1 (int) – Z положення центру сфери.  radius (int) – Радіус сфери.  blockType (int) – ID.  blockData (int) – Значення даних. |

*drawHorizontalCircle(x0, y, z0, radius, blockType, blockData=0)* - малює коло в площині X (тобто горизонтально)

|  |  |
| --- | --- |
| Параметри: | x0 (int) – Х положення центру кола.  y0 (int) – Y положення центру кола.  z0 (int) – Z положення центру кола.  radius (int) – Радіус кола.  blockType (int) – ID.  blockData (int) – Значення даних. |

**2)Points** - колекція Minecraft позицій або Vec3. Використовується для малювання граней MinecraftDrawing.drawFace().

*Class minecraftstuff.Points*

*add(x,y,z)* - додати одну позицію до списку точок.

|  |  |
| --- | --- |
| Параметри: | x (int) – Позиція Х.  y (int) – Позиція Y.  z (int) – Позиція Z. |

*getVec3s()* - повертає список позицій Vec3

**3)MinecraftShape API** - дозволяє створювати та обробляти фігури (колекції блоків) у Minecraft.Фігури можуть бути малими або великими, і API допомагає, ефективно рухаючи їх і змінюючи лише ті блоки, які насправді змінилися.

Сlassminecraftstuff.MinecraftShape(mc, position, shapeBlocks=None, visible=True)

*MinecraftShape* - реалізація "фігури" в Minecraft. Кожна фігура складається з одного або багатьох блоків з положенням відносно один одного. Фігури можуть бути перетворені за допомогою руху та обертання. Коли фігура змінюється і перемальовується в Minecraft, оновлюються лише ті блоки, які змінились.

*clear()* - очищає фігуру в Minecraft.

*draw()* - малює фігуру в Minecraft, беручи до уваги, де вона була намальована востаннє, лише оновлюючи змінені блоки.

*getShapeBlock(x, y, z*) - повертає ShapeBlock для "фактичного положення".

|  |  |
| --- | --- |
| Параметри: | x (int) – Позиція Х.  y (int) – Позиція Y.  z (int) – Позиція Z. |

*move(x, y, z)* - переміщує положення фігури до x, y, z.

|  |  |
| --- | --- |
| Параметри: | x (int) – Позиція Х.  y (int) – Позиція Y.  z (int) – Позиція Z. |

*moveBy*(x, y, z) - переміщує положення фігури на x, y, z.

|  |  |
| --- | --- |
| Параметри: | x (int) – Кількість блоків для переміщення в X.  y (int) – Кількість блоків для переміщення Y.  z (int) – Кількість блоків для переміщення Z. |

*redraw()* - перемальовує фігуру в Minecraft, очищаючи всі блоки та перемальовуючи їх.

*reset()* - скидає фігуру у початкове положення.

*rotate(yaw, pitch, roll)* - встановлює обертання фігури за допомогою повороту навколо вертикальної осі, повороту та обертання

|  |  |
| --- | --- |
| Параметри: | yaw (float) – Обертання щодо вертикальної осі в градусах.  pitch (float) – Кут обертання в градусах.  roll (float) – Обертання осі в градусах. |

*rotateBy(yaw, pitch, roll)* – збільшує the rotation of a shape by yaw, pitch and roll

*setBlock(x, y, z, blockType, blockData=0, tag='')* - встановлює один блок у фігурі і перемальовує, щоб він намалював одну точку в Minecraft, тобто 1 блок

**5) MinecraftTurtle**

**Minecraft Turtle** - це відтворення класичної графічної черепахи для Minecraft. Ключова відмінність полягає в тому, що ви можете малювати в 3 вимірах, а не лише в 2.

Class*minecraftstuff.MinecraftTurtle(*mc*,*position=<MagicMock name='mock()' id='139984989718456'>*)*

*MinecraftTurle* - графічна черепаха, яку можна використовувати для створення "малюнків" у Minecraft, контролюючи її положення, кути та напрямок

**ДОДАТОК Г**

**КОД ДО ПРАКТИЧНИХ ЗАВДАНЬ**

1. Завдання 1.

import mcpi.minecraft as minecraft

mc = minecraft.Minecraft.create()

x = int(input("Введіть значення X = "))

y = int(input("Введіть значення Y = "))

while y<0:

print("Невірне значення Y")

y = int(input("Введіть значення Y = "))

z = int(input("Введіть значення Z = "))

mc.player.setPos(x,y,z)

mc.postToChat("Teleported to "+str(x)+" "+str(y)+" "+str(z))

print("Гравця телепортовано на координати ",x,y,z)

1. Завдання 2

import mcpi.minecraft as minecraft

mc = minecraft.Minecraft.create()

x = 100

y = 200

z = "30"

mc.pleyr.setPos(str(x),z,c)

print("Все працює!")

1. Завдання 3.

import mcpi.minecraft as minecraft

from random import \*

from time import \*

mc = minecraft.Minecraft.create()

x = randint(-100000,100000)

y = randint(70,200)

z = randint(-100000,100000)

mc.player.setPos(x,y,z)

sleep(10)

x = randint(-100000,100000)

y = randint(70,200)

z = randint(-100000,100000)

mc.player.setPos(x,y,z)

sleep(10)

x = randint(-100000,100000)

y = randint(70,200)

z = randint(-100000,100000)

mc.player.setPos(x,y,z)

sleep(10)

1. Завдання 4.

import mcpi.minecraft as minecraft

mc = minecraft.Minecraft.create()

pos= mc.player.getTilePos()

x = pos.x

y = pos.y

z = pos.z

mc.postToChat(x)

mc.postToChat(y)

mc.postToChat(z)

1. Завдання 5.

import mcpi.minecraft as minecraft

import time

mc = minecraft.Minecraft.create()

pos= mc.player.getTilePos()

x = pos.x

y = pos.y

z = pos.z

time.sleep(10)

pos= mc.player.getTilePos()

x1 = pos.x

y1 = pos.y

z1 = pos.z

mc.postToChat("Player has moved x = " + str(x1-x)+", y = "

+ str(y1-y) + ", z = "+ str(z1-z))

1. Завдання 6.

import mcpi.minecraft as minecraft

mc = minecraft.Minecraft.create()

pos= mc.player.getPos()

x = pos.x

y = pos.y

z = pos.z

a = input()

if a=='x':

mc.player.setPos(x+int(input()),y,z)

elif a=='y':

mc.player.setPos(x,y+int(input()),z)

else:

mc.player.setPos(x,y,z+int(input()))

1. Завдання 7.

import mcpi.minecraft as minecraft

mc = minecraft.Minecraft.create()

pos= mc.player.getPos()

x = pos.x

y = pos.y

z = pos.z

ID = int(input())

mc.setBlock(x+1,y,z,ID,0)

1. Завдання 8.

import mcpi.minecraft as minecraft

mc = minecraft.Minecraft.create()

pos= mc.player.getPos()

x = pos.x

y = pos.y

z = pos.z

mc.setBlock(x+1,y,z,5,0)

mc.setBlock(x-1,y,z,5,0)

mc.setBlock(x,y,z-1,5,0)

mc.setBlock(x,y,z+1,5,0)

mc.setBlock(x+1,y,z+1,5,0)

mc.setBlock(x+1,y,z-1,5,0)

mc.setBlock(x-1,y,z+1,5,0)

mc.setBlock(x-1,y,z-1,5,0)

1. Завдання 9.

import mcpi.minecraft as minecraft

mc = minecraft.Minecraft.create()

pos= mc.player.getPos()

for i in range(10):

mc.setBlock(pos.x,pos.y+i,pos.z,1,0)

1. Завдання 10.

import mcpi.minecraft as minecraft

from random import \*

mc = minecraft.Minecraft.create()

pos= mc.player.getPos()

pos.x+=1

for i in range(30):

mc.setBlock(pos.x,pos.y+i,pos.z,randint(0,100),0)

1. Завдання 11.

import mcpi.minecraft as minecraft

mc = minecraft.Minecraft.create()

pos= mc.player.getPos()

x = pos.x + 5

y = pos.y

z = pos.z

size,ID = map(int,input().split())

mc.setBlocks(x,y,z,x+size,y+size,z+size,ID,0)

1. Завдання 12

import mcpi.minecraft as minecraft

mc = minecraft.Minecraft.create()

pos= mc.player.getPos()

x = pos.x + 4

y = pos.y

z = pos.z

mc.setBlocks(x,y,z,x+10,y+10,z,1)

mc.setBlocks(x+1,y+1,z,x+9,y+9,z,0)