**Зміст**

Вступ

1. Аналіз стану мови програмування Python та обґрунтування теми 4
   1. Вибір бібліотеки для графічного інтерфейсу програми 6
2. Розробка програмного забезпечення «Класи та об’єкти» 7
   1. Аналіз поставленої задачі 7
   2. Розробка головного меню 7
   3. Розробка допоміжних вікон 9
   4. Розробка файлу для запуску програми 10

Висновок

Використана література

Додатки

Вступ

В даній курсовій роботі був розроблений програмний засіб «Класи та об’єкти» за допомогою мови програмування Python.

Тема була вибрана через її актуальність на сьогоднішній день. Саме в наш час йде активний розвиток технологій, з кожним роком з’являються все більше нових мов програмування які допомагають програмістам розробляти високотехнологічні програмні засоби, але щоб опанувати якусь мову програмування її потрібно освоїти як теоретично так і практично. Особливо зараз надається велика перевага програмістам які вміють працювати з мовами які підтримують об'єктно-орієнтоване програмування. Можна сказати, що ця тема вибрана для того щоб навчити школярів і студентів працювати з об'єктно-орієнтованим програмуванням.

Обрана тема важлива тому що на даний час є мало ресурсів які надають безкоштовне і якісне навчання, а програма «Класи та об’єкти» є у вільному доступі і повністю безкоштовна. Дуже важливо щоб зараз люди навчалися програмувати, адже програмування вчить логічному мисленню, розвиває творчість і допомагає вирішувати проблеми.

Метою даної курсової роботи є створення за допомогою мови Python, програми яка допоможе освоїти об'єктно-орієнтоване програмування людям, які починають вивчати програмування.

**Аналіз стану мови програмування Python та обґрунтування теми**

На 2020 рік мова програмування Python є одною з самих популярних і потужних мов програмування. За допомогою Python створюють різноманітні програми для широкого класу комп’ютерів. Python підтримує модулі та пакети модулів, що сприяє модульності та повторному використанні коду.

Об'єктно-орієнтоване програмування (ООП) – це результат еволюції більш ранніх метедологій програмування. Воно з’явилось з процедурного програмування. В об’єктно-орієнтованому програмуванні більше не потрібно ділити завдання на під задачі. Ви намагаєтесь побачити проблему, сформовану взаємодією між абстракціями - ідеалізованими об'єктами в реальному світі. Тому мета полягає у використанні в програмуванні повсякденний досвід та знання про поведінку звичайних об’єктів. Об'єкти абстрактні, але мають чітко визначені атрибути або ролі. Об'єктно-орієнтоване програмування - це метод програмування, що імітує виконання людиною певної роботи. У порівнянні з традиційним програмуванням воно є більш структурованим, модульним та абстрактним.

Є три основних «кита» на яких тримається ООП – це інкапсуляція, успадкування та поліморфізм.

Інкапсуляція – приховування деталей роботи класів і об’єктів від користувача, Використовується для захисту від зміни даних із інших частнн коду.

Успадкування – має на увазі те, що дочірній клас містить всі атрибути батьківського класу, при цьому деякі з них можуть бути перевизначені чи додані в дочірньому.

Поліморфізм – різна поведінка одного і того ж метода в різних класах. Наприклад ми можемо додати два числа, і додати два рядки. При цьому отримаємо різний результат, так як числа і рядки є різними класами.

Python є об'єктно-орієнтованою мовою програмування. Він надзвичайно легкий в вивчені і має невелику кількість ключових слів, і в той же час дуже гнучкий і потужний. Ця мова більш високо рівня, ніж Pascal, C++ і звичайно C, що досягається в основному, за рахунок вбудованих високорівневих структур даних (списків, словників, кортежів).

В мові Python є багато переваг. Наприклад те що Python реалізований майже на всіх платформах і операційних системах, Наступна важлива риса – це розширюваність і цьому надається велике значення адже Python, як пише автор, був задуманий як розширювана мова. Це означає що кожний може вдосконалювати мову і писати під неї нові бібліотеки. Або ж написав на C++ свої доповнення до Python і скомпілювавши програму ми отримаємо «покращений» інтерпретатор з новими можливостями і функціями. Ще одне досягнення мови – це велика кількість різноманітних бібліотек і модулів, які забезпечують різні додаткові можливості. Такі модулі можна писати на C, C++ і навіть на самому Python. Приклад деяких додаткових модулів:

Модуль **os.path** – надає безліч функцій для роботи з операційною системою, причому їхня поведінка, як правило не залежить від операційної системи, тому програми залишаються переносними.

Модуль **unittest** – модуль який використовується для автоматизації тестування програмного забезпечення.

Набір модулів **PyQt5** – призначений для роботи з графічним інтерфейсом користувача, включає в себе близько 650 класів і 6000 функцій і методів.

Але в мові є й недоліки, одним з таких є порівняно невисока швидкість виконання Python-програми. Однак, це не перекриває переваги мови, при написанні програм для яких час запуску не є дуже критичним.

Вибір бібліотеки для графічного інтерфейсу програми

При виборі бібліотеки для створення графічного інтерфейсу програми було вибрано набір модулів PyQt5 і бібліотеку PySide2.

PySide2 – це бібліотека для створення прив’язки Qt до мови програмування Python. Він направлений на реалізацію підтримки всіх можливостей Qt, в тому числі QtQuick і QtMobility. На даний момент підтримується наступні версії компонентів і платформ

|  |  |
| --- | --- |
| Qt | 5.12 + |
| QtMobility | 1.2.2 |
| Python | 2.7, 3, 3.5, 3.6, 3.7+ |
| OS | Linux/X11, Maemo 5, MeeGo, Windows, Mac OS X |

Проект PySide2 також має набір інструментів, які дозволяють створювати прив’язки будь-яких бібліотек написаних на C чи C++. PySide2 розповсюджується під ліцензією LGPL, а це означає що його можна використовувати в закритих і відкритих комерційних проектах. Також вихідний код відкритий і доступний що дозволяє кожному доповнювати програму.

**Розробка програмного забезпечення «Класи та об’єкти»**

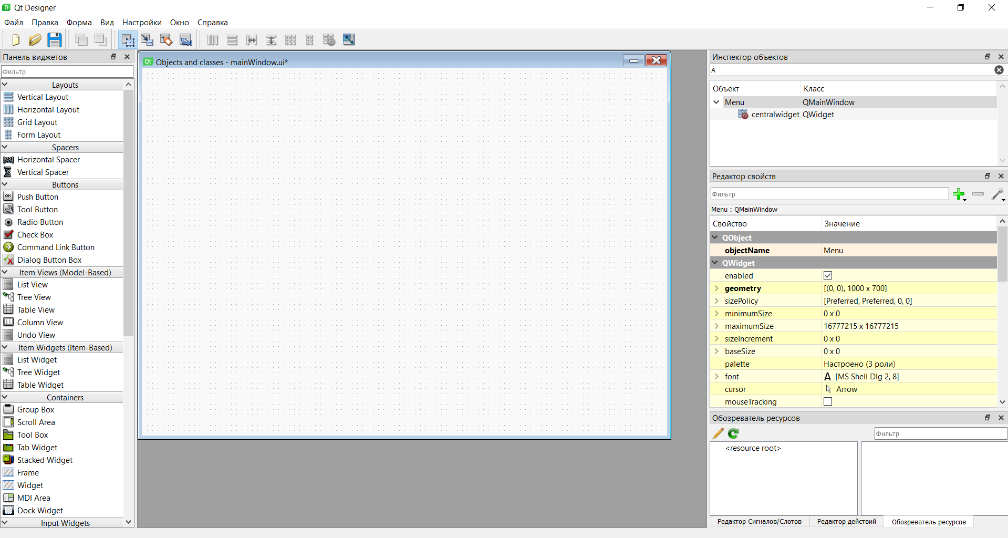
Аналіз поставленої задачі

Програма «Класи та об’єкти» повинна складатися з наступних частин: головного вікна, допоміжних вікон і окремого файлу який буде запускати програму. Програма повинна бути зрозумілою для користувача, в ній повинні бути теоретичні відомості про «Об'єктно-орієнтоване програмування», «Об’єкти та класи в Python» та частина з практикою по пройденому матеріалу.

Розробка головного меню

Для початку встановимо бібліотеку PySide2. Відкриваємо консоль комп’ютера і пишемо наступну команду: “pip install PySide2” і чекаємо кінця завантаження.

Після установки в директорії Programs\Python\Python37\Lib\site-packages\PySide2 з’явиться програма designer, яка створена щоб облегшити розробку графічного інтерфейсу. Запускаємо програму і бачимо перед собою вікно «Новая форма – Qt designer» і в представленому списку вибираємо «Main Window». Ми створили вікно нашої програми, зліва знаходиться блок з елементами які ми можемо перетягнути на головне вікно, праворуч знаходяться «Инспектор объектов» і «Редактор свойств» для визначення властивостей класів і об’єктів.



1.Вікно програми designer

Перетягуємо на наше вікно елемент QPushButton, який відповідає за відображення кнопок на нашому вікні, тепер ми можемо змінити його розмір і задекорувати. Натискаємо ПКМ по створеній кнопці і вибираємо пункт «Изменить styleSheet», в вікні що відрилось ми можемо прописувати дизайн кнопки на мові стилю сторінок CSS.

Пропишемо наступний код:

QPushButton{

background:#F9F9F9;

border-radius: 10px;

}

QPushButton:hover{

background:#c2c2ff

}

Тепер додамо до нашої програми основний текст, зробити це можна через елемент QLabel, перетягуємо його на наше вікно, натискаємо ПКМ по об’єкту і вибираємо пункт «Изменить форматированный текст…» і вписуємо потрібний нам текст. Щоб додати зображення потрібно потрібно додати елемент QLabel і в меню «Редактор свойств» знайти пункт pixmap, і натиснути «Выбрать файл..». Тепер коли ми закінчили редагування нашого графічного інтерфейсу потрібно зберегти файл. В designer натискаємо комбінацію кнопок Ctrl + S і вказуємо місце куди буде збережено файл. Файл зберігся в вказану директорію в фоматі mainWindow.ui.

Тепер потрібно переконвертувати файл із mainWindow.ui в mainWindow.py. Для цього скористаємось наступною командою:

“pyside2-uic mainWindow.ui -x -o mainWindow.py”

Вставляємо команду в консоль і натискаємо Enter. Коли програма закінчила конвертування поряд із mainWindow.ui повинен з’явитися mainWindow.py. Відкриваємо щойно створений файл в редакторі кода, якщо ми запустимо файл в нас відкриється вікно з готовим інтерфейсом. Переходимо до створення інших вікон

Розробка допоміжних вікон

Розробка інших вікон схожа із розробкою головного вікна, а саме ми розставляємо елементи на новому створеному вікні, декоруємо їх через styleSheet і конвертуємо із .ui в .py. Всього вийшло п'ятнадцять вікон: одне головне вікно, одне вікно «Про програму», одне вікно «Бонус», два вікна «Основні поняття ООП», сім вікон «Класи і об’єкти» і три вікна «Практика по пройденому матеріалу».

Розробка файлу для запуску програми

Переходимо в редактор коду і створюємо файл main.py. Далі відкриваємо файл і імпортуємо бібліотеки:

import sys

from PySide2 import QtCore, QtGui, QtWidgets

from mainWindow import Ui\_Menu

from secondWindow1 import Ui\_Menu\_oop1

from secondWindow2 import Ui\_Menu\_oop2

from bonusWindow import Ui\_Menu\_bonus

from about\_pr import Ui\_About\_pr

from secondWindow\_class1 import Ui\_class\_1

from secondWindow\_class2 import Ui\_class\_2

from secondWindow\_class3 import Ui\_class\_3

from secondWindow\_class4 import Ui\_class\_4

from secondWindow\_class5 import Ui\_class\_5

from secondWindow\_class6 import Ui\_class\_6

from secondWindow\_class7 import Ui\_class\_7

from practicWindow1 import Ui\_practicWin1

from practicWindow2 import Ui\_practicWin2

from practicWindow3 import Ui\_practicWin3

import webbrowser

import os

Ми імпортували всі бібліотеки які нам потрібні для роботи з вікнами, далі нам треба вирізати із файлу mainWindow.py останню частину кода, а саме:

app = QtWidgets.QApplication(sys.argv)

mainWindow = QtWidgets.QDialog()

ui = Ui\_Menu()

ui.setupUi(mainWindow)

mainWindow.show()

sys.exit(app.exec\_())

Вставляємо в наш файл і запускаємо код. Тепер можна перейти до настроювання вікон. Створюємо нову функцію:

def open\_Oop():

global Menu\_oop1

Menu\_oop1 = QtWidgets.QMainWindow()

ui = Ui\_Menu\_oop1()

ui.setupUi(Menu\_oop1)

Menu.close()

Menu\_oop1.show()

Ми створили функцію яка буде відкривати нам вікно Menu\_oop1, команда Menu.close() відповідає за закриття головного вікна, а Menu\_oop1.show() – за відкриття вікна «Основні поняття ООП», але щоб переключати вікна між собою нам потрібно підключити кнопку, використаємо кнопку яку ми зробили:

ui.main\_oop.clicked.connect(open\_Oop)

Тепер наш код виглядатиме так:

app = QtWidgets.QApplication(sys.argv)

mainWindow = QtWidgets.QDialog()

ui = Ui\_Menu()

ui.setupUi(mainWindow)

mainWindow.show()

def open\_Oop():

global Menu\_oop1

Menu\_oop1 = QtWidgets.QMainWindow()

ui = Ui\_Menu\_oop1()

ui.setupUi(Menu\_oop1)

Menu.close()

Menu\_oop1.show()

ui.main\_oop.clicked.connect(open\_Oop)

sys.exit(app.exec\_())

В програмі була реалізована кнопка «Далі» і «Назад», щоб ними можна було користуватись запишемо в функцію open\_Oop наступне вікно - open\_Oop2:

def open\_Oop():

global Menu\_oop1

Menu\_oop1 = QtWidgets.QMainWindow()

ui = Ui\_Menu\_oop1()

ui.setupUi(Menu\_oop1)

Menu.close()

Menu\_oop1.show()

def open\_Oop():

global Menu\_oop2

Menu\_oop2 = QtWidgets.QMainWindow()

ui = Ui\_Menu\_oop2()

ui.setupUi(Menu\_oop2)

Menu\_oop1.close()

Menu\_oop2.show()

ui.next.clicked.connect(open\_Oop2)

ui.main\_oop.clicked.connect(open\_Oop)

sys.exit(app.exec\_())

За допомогою такого методу можна реалізовувати багато вікон. В програмі є кпопка «Вихід» для неї теж пропишемо код:

def exit\_app():

app.quit()

ui.exit. .clicked.connect(exit\_app)

Далі потрібно реалізувати невелику логіку для вікна «Практика по пройденому матеріалу», для цього перейдемо в файл practicWindow.py і в функції Ui\_practicWin1 напишемо наступне:

def check\_r():

balls = 0

if self.one.isChecked():

balls = balls + 1

if self.one\_2.isChecked():

balls = balls + 1

if self.two\_3.isChecked():

balls = balls + 1

else:

balls = balls

self.result.setText(str(balls))

self.verify.clicked.connect(check\_r)

Представлений код буде рахувати кількість відповідей, в разі вибору користувачем правильної відповіді додавати до змінної balls одну правильну відповідь

Для Ui\_practicWin2 було додане поле для вводу правильно відповіді:

if self.lineEdit.text() == "@staticmethod":

balls = balls + 1

Виснок

При розробці програми «Класи та об’єкти» я застосував свої знання з об'єктно-орієнтованого програмування. Поглибив розуміння основних принципів об’єктно-орієнтованої ідеології програмування. Відпрацював на практиці основні методи та засоби об’єктно-орієнтованого програмування. Навчився розробляти ієрархію класів. Практично застосував такі поняття, як інкапсуляція, наслідування, перевантаження та перевизначення функцій. Набув практичних навиків роботи.

Використана література

 Прохоренок Н. А. Python 3 и PyQt. Разработка приложений

Программирование на Python 3. Подробное руководство

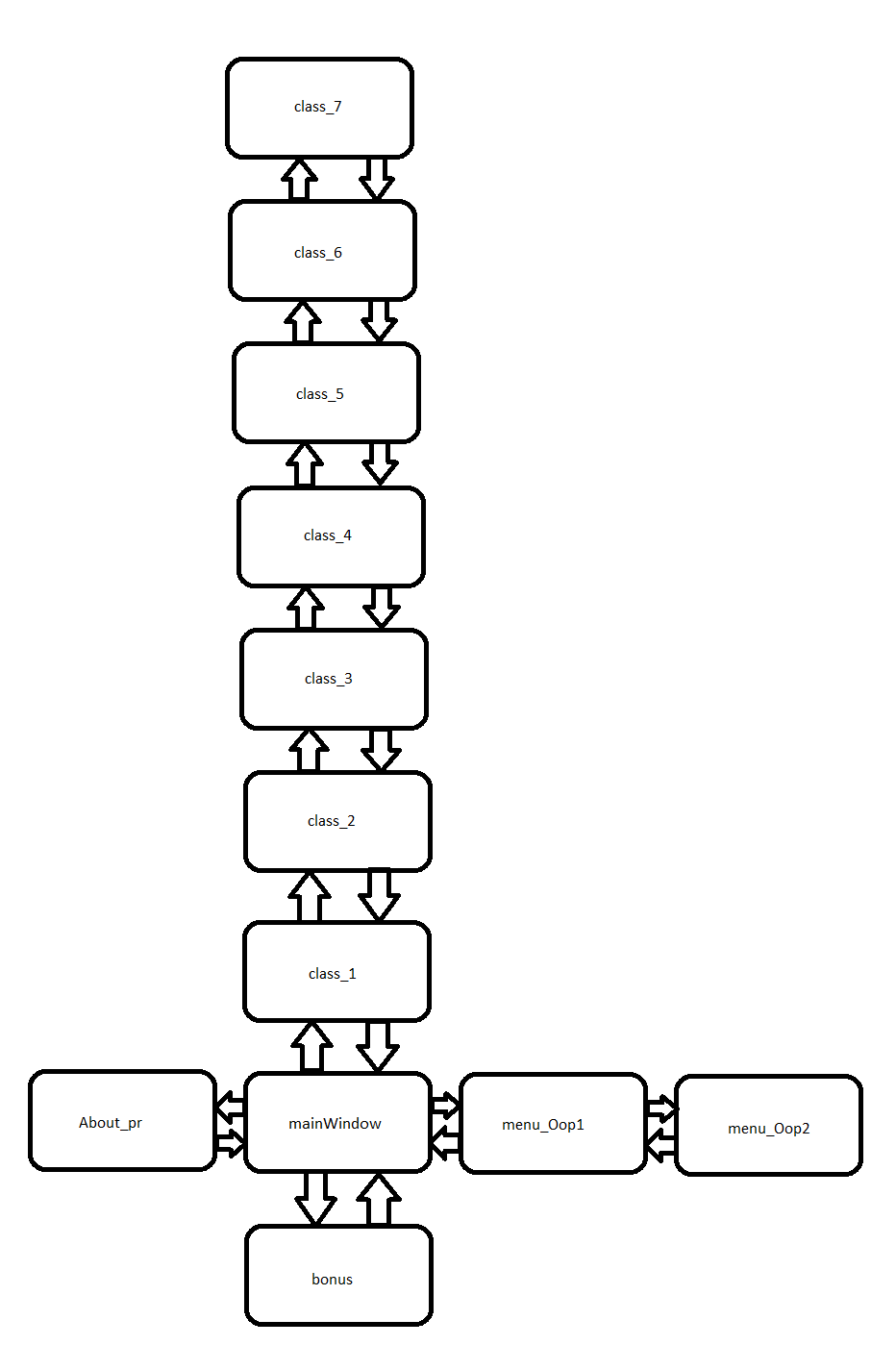
Python Cookbook: Recipes for Mastering Python 3

Qt5 Python GUI Programming Cookbook: Building responsive and powerful cross-platform

Python Tricks: A Buffet of Awesome Python Features

ДОДАТОК А

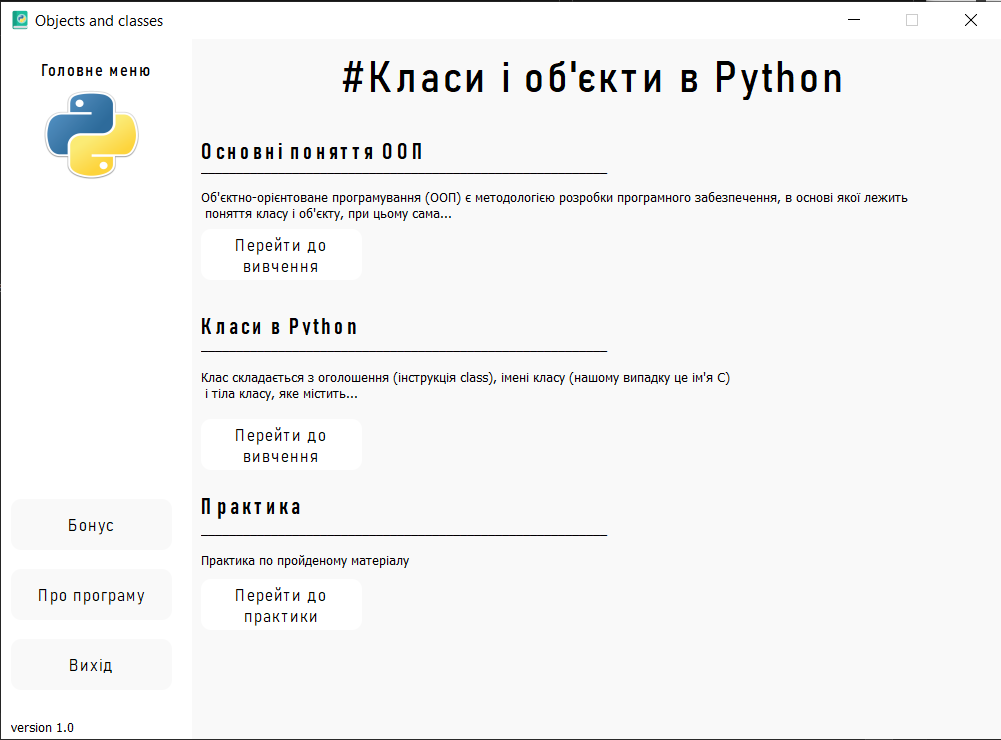
СХЕМА АЛГОРИТМИ ПРОГРАМИ



ДОДАТОК Б

РОЗДРУК РЕЗУЛЬТАТІВ РОБОТИ ПРОГРАМИ

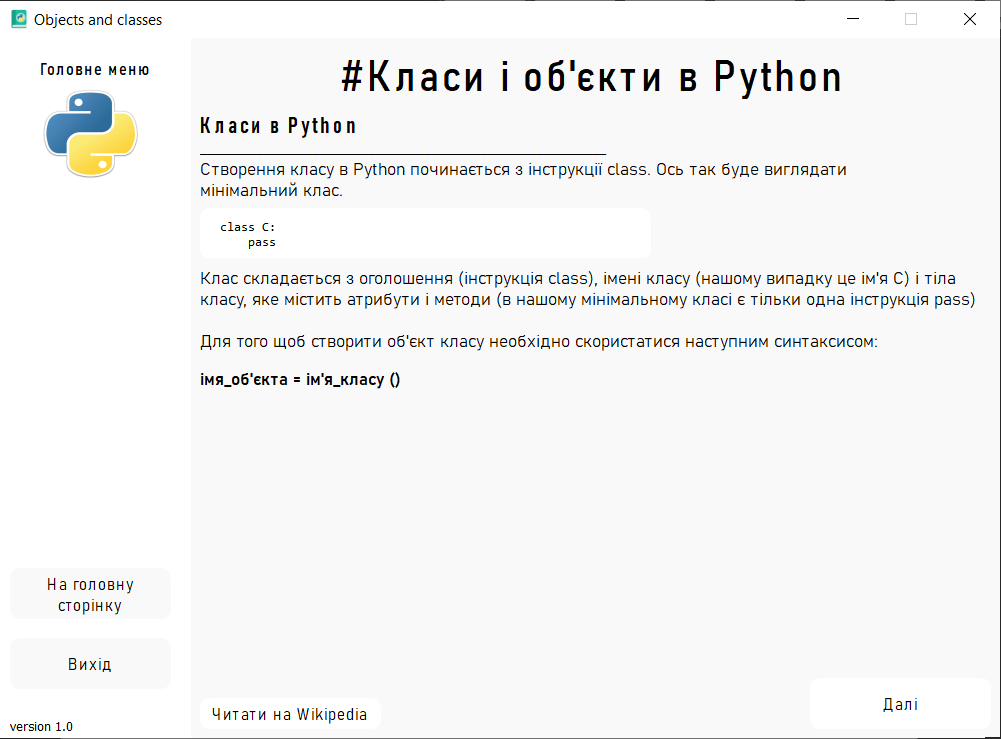
Додаток 1 Вікно головного меню



Додаток 2 Вікно «Про програму»



Додаток 3 Вікно «Об’єкти та класи в Python»



Додаток 4 Вікно «Практика по пройденому матеріалові»

