**TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP. HCM**

**KHOA CÔNG NGHỆ THÔNG TIN**

**A logo of hands holding a book

Description automatically generated**

**ĐỒ ÁN HỌC PHẦN**

TÊN HỌC PHẦN: Lập trình Python (PYPRO)

MÃ SỐ LỚP HP: IPPA233277\_01CLC

Tên chủ đề EDA: **PHÂN TÍCH THỐNG KÊ VỀ CẦU THỦ NBA CHO MÙA GIẢI THƯỜNG NÊN 2023-2024**

Chủ đề CV: NHẬN DIỆN HÌNH HỌC CƠ BẢN

Chủ đề Game: LẬP TRÌNH PYTHON GAME CAR RACING

Họ tên sinh viên: Cao Thị Ngọc Phụng [27]

Mã số sinh viên: 21110276

Lớp: 21110CLIS

Ngày nộp: 8/12/2024

Ký tên:

**TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP.HCM**

**KHOA [CÔNG NGHỆ THÔNG TIN | CHẤT LƯỢNG CAO]**

**ĐỀ TÀI ĐỒ ÁN HỌC PHẦN**

TÊN HỌC PHẦN: Lập trình Python (PYPRO)

MÃ SỐ LỚP HP: IPPA233277\_01CLC

Tên đề tài: **Lập trình Python phân tích thống kê về cầu thủ NBA cho mùa giải thường niên 2023-2024**

Chủ đề: Lập trình Python hỗ trợ CV nhận diện hình học cơ bản

Chủ đề: Lập trình Python Game Car Racing

**Giảng viên giảng dạy: VÕ XUÂN THỂ**

**Họ tên sinh viên: Cao Thị Ngọc Phụng [27]**

**Mã số sinh viên: 21110276**

**Lớp: 21110CLIS**

**TÊN CÁC FILE SẢN PHẨM ĐỀ TÀI:** NỘI DUNG THƯ MỤC BÀI LÀM ĐỒ ÁN HỌC PHẦN

**G127CTNPNBAPlayer, bao gồm:**

[1] File báo cáo: **2\_ G127CTNPNBAPlayer.docx**

[2] Các files mã nguồn lập trình Python:

G127CTNPNBAPlayer10EDA.py: ……….

G127CTNPNBAPlayer20CV.py: ….

G127CTNPNBAPlayer30Game.py: ….

G127CTNPNBAPlayer31GameWin.py: ....

G127CTNPNBAPlayer32GameLayer.py: ....

G127CTNPNBAPlayer33GameDraw.py: ....

G127CTNPNBAPlayer34GameCar.py: ....

G127CTNPNBAPlayer40HNTower.py: ….

G127CTNPNBAPlayer41GPTB2.py:…

G127CTNPNBAPlayer50VA.py: ….

G127CTNPNBAPlayer60Plot.py: …

G127CTNPNBAPlayer61Plot1WH.py: …

G127CTNPNBAPlayer70RollCall.py: ….

[3] Các thư mục lưu các dữ liệu phục vụ mã nguồn Python:

G127CTNPNBAPlayer10EDA: …

G127CTNPNBAPlayer20CV:…

G127CTNPNBAPlayer30Game: …

[4] File giới thiệu nội dung thực mục bài làm này:

1\_ G127CTNPNBAPlayer \_Intro.docx

**NHẬN XÉT CỦA GIẢNG VIÊN**

**Ngày……./……../2024**

**Giảng viên: Ký tên**

**TP.HCM, ngày …. tháng 12 năm 2024**

# LỜI CẢM ƠN

Trước tiên, em xin chân thành cảm ơn thầy Võ Xuân Thể, giảng viên hướng dẫn, người đã luôn tận tình chỉ bảo và hỗ trợ em trong suốt quá trình học tập và hoàn thành báo cáo này. Thầy không chỉ truyền đạt cho em những kiến thức quý giá về lập trình Python mà còn giúp em rèn luyện tư duy logic, phương pháp giải quyết vấn đề hiệu quả. Sự kiên nhẫn và nhiệt huyết của thầy đã giúp em vượt qua nhiều thử thách trong học tập và nghiên cứu.

Bên cạnh đó, em xin cảm ơn các bạn bè và đồng nghiệp đã cùng em chia sẻ kiến thức, kinh nghiệm trong quá trình học. Sự hỗ trợ và động viên của các bạn đã tạo ra một môi trường học tập đầy năng lượng và sáng tạo, giúp em cải thiện bản thân từng ngày.

Em cũng không thể không nhắc đến gia đình, những người luôn là chỗ dựa vững chắc và động viên em trong những lúc khó khăn. Những lời khuyên và sự yêu thương của gia đình là nguồn động lực lớn lao giúp em tiếp tục cố gắng và hoàn thành bài báo cáo này.

Cuối cùng, em xin cảm ơn các tài liệu, nguồn tài nguyên trực tuyến và cộng đồng lập trình viên đã chia sẻ những kiến thức hữu ích, giúp em áp dụng và phát triển kỹ năng lập trình Python. Em hy vọng rằng báo cáo này sẽ góp phần nhỏ vào sự phát triển của bản thân và đóng góp vào cộng đồng học thuật trong lĩnh vực lập trình.

# MỤC LỤC

[CHƯƠNG 1: GIỚI THIỆU VỀ ĐỒ ÁN HỌC PHẦN 1](#_Toc184339549)

[1.1 Tổng quan về Đồ án học phần 1](#_Toc184339550)

[1.2 Nội dung chuyên môn chính của Đồ án học phần 1](#_Toc184339551)

[1.3 Công cụ và nền tảng kỹ thuật thực hiện Đồ án học phần 2](#_Toc184339552)

[1.4 Bố cục của báo cáo 2](#_Toc184339553)

[CHƯƠNG 2: CƠ SỞ LÝ THUYẾT CỦA HỌC PHẦN 3](#_Toc184339554)

[2.1 GIỚI THIỆU VỀ LẬP TRÌNH PYTHON 3](#_Toc184339555)

[2.1.1 Giới thiệu chung 3](#_Toc184339556)

[2.1.2 Cài đặt “bộ thảo chương” (soạn thảo chương trình IDE) và “bộ dịch” 3](#_Toc184339557)

[2.1.3. Giới thiệu các bộ tương tác lập trình 3](#_Toc184339558)

[2.1.4. Cấu hình môi trường tương tác lập trình 4](#_Toc184339559)

[2.2 TỔNG QUAN VỀ NGÔN NGỮ LẬP TRÌNH PYTHON 4](#_Toc184339560)

[2.2.1 Danh hiệu (Identifier) 4](#_Toc184339561)

[2.2.2 Về viết mã lệnh lập trình (Code) 4](#_Toc184339562)

[2.2.3 Về chú giải (Comment) 4](#_Toc184339563)

[2.2.4 Các phép toán 4](#_Toc184339564)

[2.2.5 Chuỗi 4](#_Toc184339565)

[2.2.6 Lệnh 4](#_Toc184339566)

[2.2.7 Định nghĩa hàm 4](#_Toc184339567)

[2.2.8 Danh sách 5](#_Toc184339568)

[2.2.9 Bộ dữ liệu: Tuple 5](#_Toc184339569)

[2.2.10 Tập hợp: Set 5](#_Toc184339570)

[2.2.11 Từ điển: Dictionary 5](#_Toc184339571)

[2.2.12 Module 5](#_Toc184339572)

[2.2.13. Package 5](#_Toc184339573)

[2.2.14 Lớp 5](#_Toc184339574)

[2.2.15 Xử lý ngoại lệ 5](#_Toc184339575)

[2.3 GIỚI THIỆU CÁC THƯ VIỆN VÀ CÔNG CỤ CĂN BẢN 6](#_Toc184339576)

[2.3.1 Nạp thư viện 6](#_Toc184339577)

[2.3.2 speech\_recognition 6](#_Toc184339578)

[2.3.3 gtts 6](#_Toc184339579)

[2.3.4 playsound 6](#_Toc184339580)

[2.3.5 tkinter 6](#_Toc184339581)

[2.3.6 os 6](#_Toc184339582)

[2.3.7 Pandas 6](#_Toc184339583)

[2.3.8 Tensorflow 6](#_Toc184339584)

[2.3.9 NumPy 7](#_Toc184339585)

[2.3.10 SCIPY 7](#_Toc184339586)

[2.3.11 Theano 7](#_Toc184339587)

[2.3.12 Matplotlib 7](#_Toc184339588)

[2.3.13 scikit-learn = sklearn 7](#_Toc184339589)

[2.3.14 Keras 7](#_Toc184339590)

[2.3.15 NLTK 7](#_Toc184339591)

[2.3.16 PyTorch 8](#_Toc184339592)

[2.3.17 LightGBM 8](#_Toc184339593)

[2.3.18 Eli5 8](#_Toc184339594)

[2.4 KỸ THUẬT LẬP TRÌNH ĐỆ QUY THÁP HÀ NỘI (HANOITOWER) 8](#_Toc184339595)

[2.5 LẬP TRÌNH PYTHON XỬ LÝ GIỌNG NÓI (Trợ lý ảo: Voice Assistant) 8](#_Toc184339596)

[2.6 LẬP TRÌNH GIAO DIỆN ĐỒ HỌA (GUI) 9](#_Toc184339597)

[2.7 LẬP TRÌNH PHÂN TÍCH KHÁM PHÁ (EDA) [Thăm dò] 9](#_Toc184339598)

[2.8 LẬP TRÌNH ĐỒ THỊ & BIỂU ĐỒ TRỰC QUAN HÓA DỮ LIỆU (VISUALIZAION: PLOT) 9](#_Toc184339599)

[2.9 LẬP TRÌNH HỖ TRỢ KỸ THUẬT THỊ GIÁC MÁY TÍNH (CV) 10](#_Toc184339600)

[2.10 LẬP TRÌNH GAMEs CĂN BẢN 10](#_Toc184339601)

[CHƯƠNG 3: PHÂN TÍCH VÀ XÁC ĐINH CÁC CƠ SỞ KỸ THUẬT 11](#_Toc184339602)

[3.1 GIỚI THIỆU CHUNG VỀ CÁC CHỦ ĐỀ CỦA ĐỒ ÁN 11](#_Toc184339603)

[3.2 KỸ THUẬT LẬP TRÌNH ĐỆ QUY: THÁP HÀ NỘI (HANOI TOWER) 11](#_Toc184339604)

[3.2.1 Mô tả bài toán 11](#_Toc184339605)

[3.2.2 Các yêu cầu bài làm 11](#_Toc184339606)

[3.2.3 Full Codes (mã lệnh lập trình) 12](#_Toc184339607)

[3.2.4 Kết quả thực nghiệm 12](#_Toc184339608)

[3.2.5 Các bài tập tương tự 13](#_Toc184339609)

[3.2.5.1 Bài 1.1 13](#_Toc184339610)

[3.2.5.2 Bài 1.2 13](#_Toc184339611)

[3.2.5.3 Bài 1.3 14](#_Toc184339612)

[3.2.5.4 Bài 1.4 14](#_Toc184339613)

[3.3 LẬP TRÌNH PYTHON XỬ LÝ GIỌNG NÓI 15](#_Toc184339614)

[3.3.1 Giới thiệu bài toán 15](#_Toc184339615)

[3.3.2 Các yêu cầu bài làm 15](#_Toc184339616)

[3.3.3 Các thư viện Python liên quan 16](#_Toc184339617)

[3.3.4 Full Codes (mã lệnh lập trình) 16](#_Toc184339618)

[3.3.6 Kết quả thực nghiệm 17](#_Toc184339619)

[3.3.7 Các bài toán tương tự 17](#_Toc184339620)

[3.3.7.1 B2Ex2\_VoiceAssistant\_1 17](#_Toc184339621)

[3.3.7.2 B2Ex2\_VoiceAssistant\_2 [Ứng dụng vào phần mềm discord] 19](#_Toc184339622)

[3.4 LẬP TRÌNH GIAO DIỆN ĐỒ HỌA (GUI) 22](#_Toc184339623)

[3.4.1 Giới thiệu bài toán 22](#_Toc184339624)

[3.4.2 Các yêu cầu bài làm 22](#_Toc184339625)

[3.4.3 Các thư viện Python liên quan 22](#_Toc184339626)

[3.4.4 Full code (mã lệnh lập trình 22](#_Toc184339627)

[3.4.6 Kết quả thực nghiệm 23](#_Toc184339628)

[3.4.7 Các bài toán tương tự 23](#_Toc184339629)

[3.4.7.1 Bài GUI 1 23](#_Toc184339630)

[3.4.7.2 Bài GUI 2 25](#_Toc184339631)

[CHƯƠNG 4 : PHÂN TÍCH THỐNG KÊ VỀ CẦU THỦ NBA CHO MÙA GIẢI THƯỜNG NÊN 2023-2024 28](#_Toc184339632)

[4.1 Giới thiệu chủ đề EDA G127CTNPNBAPlayer 28](#_Toc184339633)

[4.2 Giới thiệu bài toán (Yêu cầu đặt ra) 28](#_Toc184339634)

[4.3 Thông tin về file dữ liệu thực nghiệm EDA 28](#_Toc184339635)

[4.4 Phân tích yều cầu của chủ đề 29](#_Toc184339636)

[4.5 Tiến hành thực hiện 31](#_Toc184339637)

[4.6 Các nền tảng kỹ thuật liên quan 31](#_Toc184339638)

[4.6.1 Hệ thống thư viện sử dụng 31](#_Toc184339639)

[4.6.2 Codes nạp thư viện 31](#_Toc184339640)

[4.6.3 Tập dữ liệu thực nghiệm 31](#_Toc184339641)

[4.6.3.1 Bước 2: Tải tập dữ liệu 31](#_Toc184339642)

[4.6.4 Tiền xử lý (Pre-Processing) 32](#_Toc184339643)

[4.6.4.1 Bước 3 32](#_Toc184339644)

[4.6.1.2 Bước 4 34](#_Toc184339645)

[4.6.1.3 Bước 5 34](#_Toc184339646)

[4.6.1.4 Bước 6 36](#_Toc184339647)

[4.6.1.5 Bước 7 37](#_Toc184339648)

[4.6.5 Phân tích dữ liệu thăm dò: EDA 38](#_Toc184339649)

[4.6.5.1 Bước 8 38](#_Toc184339650)

[4.6.5.2 Bước 9 39](#_Toc184339651)

[4.7 Full codes 39](#_Toc184339652)

[CHƯƠNG 5: LẬP TRÌNH PYTHON HỖ TRỢ CV NHẬN DIỆN HÌNH HỌC 42](#_Toc184339653)

[5.1 Giới thiệu chủ đề lập trình 42](#_Toc184339654)

[5.2 Yêu cầu bài toán 42](#_Toc184339655)

[5.3 Các thư viện liên quan 43](#_Toc184339656)

[5.3 Cắt frames ảnh từ Video Clips 43](#_Toc184339657)

[5.3.1 Full code tạo video chứa những hình học 43](#_Toc184339658)

[5.4 Full codes bài làm 45](#_Toc184339659)

[5.5 Kết quả 46](#_Toc184339660)

[CHƯƠNG 6: LẬP TRÌNH PYTHON GAME CAR RACING 48](#_Toc184339661)

[6.1 Giới thiệu chủ đề Game Car Racing 48](#_Toc184339662)

[6.2 Lý thuyết đơn giản về Lập trình Game Car Racing 48](#_Toc184339663)

[6.3 Các thư viện liên quan 48](#_Toc184339664)

[6.4 Mô tả thuật toán 49](#_Toc184339665)

[6.5 Giao diện trò chơi 50](#_Toc184339666)

[6.4 Full codes bài làm 51](#_Toc184339667)

[6.5 Kết quả 55](#_Toc184339668)

[CHƯƠNG 7: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN CỦA ĐỒ ÁN HỌC PHẦN 56](#_Toc184339669)

[7.1. Những kết quả đạt được 56](#_Toc184339670)

[7.2. Hạn chế & Hướng khắc phục các hạn chế 56](#_Toc184339671)

[7.3. Hướng mở rộng ĐỒ ÁN HỌC PHẦN 56](#_Toc184339672)

# 

# CHƯƠNG 1: GIỚI THIỆU VỀ ĐỒ ÁN HỌC PHẦN

## 1.1 Tổng quan về Đồ án học phần

+ Đồ án Học phần vận dụng kiến thức và kỹ năng về Lập trình Python để xây dựng giao diện đồ hoạ (GUI) cho phép: tương tác với người dùng bằng giọng nói (Trợ lý ảo: Voice Assistant) với các chủ đề:

+ Phân tích thăm dò (EDA: Epolore Data Analysing) lập trình Python phân tích thống kê về cầu thủ bóng rổ trong mùa giải NBA thường niên 2023-2024 tại trang web kaggle từ tháng 10 năm 2024 đến tháng 12 năm 2024 đối với tập dữ liệu thực nghiệm theo link: <https://www.kaggle.com/datasets/bryanchungweather/nba-player-stats-dataset-for-the-2023-2024> có trực quang hóa dữ liệu dùng các kỹ thuật biểu đồ và đồ thị (Plot)

+ Lập trình Python hỗ trợ CV nhận diện hình học cơ bản

+ Lập trình Python Game Car Racing

## 1.2 Nội dung chuyên môn chính của Đồ án học phần

Bao gồm 6 nội dung lập trình chính dùng ngôn ngữ lập trình Python:

+ Trợ lý ảo: thông tin nhập có 2 phương án (lời nói dạng “chọn” OR nhập ký tự). Tin xuất “kết quả thăm dò” ở cả 2 dạng: text và lời nói

+ Phân tích dữ liệu khám phá || thăm dò phân tích thống kê về cầu thủ bóng rổ trong mùa giải NBA thường niên 2023-2024

+ Trực quang hóa dữ liệu thăm phân tích thống kê về cầu thủ bóng rổ trong mùa giải NBA thường niên 2023-2024.

+ Giao diện đồ họa dạng [WinForm or Webform or Mobileform]

+ Lập trình Python hỗ trợ CV nhận diện hình học cơ

+ Lập trình Python Game Car Racing

Nội dung chuyên môn của Đồ án này là nền tảng cho các lĩnh vực:

1. Lập trình AI (TTNT), ML (HM), Deep Learning (HS)
2. Data Science: Lập trình Khoa học dữ liệu, đặc biệt EDA (NBAPlayer): phân tích thống kê về cầu thủ bóng rổ trong mùa giải NBA thường niên 2023-2024
3. CV: Lập trình Python hỗ trợ CV nhận diện hình học cơ
4. Games: Game Car Racing
5. Data Visualization = Plot: Lập trình trực quang hóa dữ liệu, dùng biểu đồ & đồ thi
6. GUI (….): Lập trình giao diện đồ họa

## 1.3 Công cụ và nền tảng kỹ thuật thực hiện Đồ án học phần

Ngôn ngữ lập trình **Python**

+ Công cụ lập trình và biên dịch: **VSCode (Visual Studio Code) + Python Extension**

+ Các thư viện chính:

|  |
| --- |
| # Speech  import speech\_recognition as sr  # Nhận diện giọng nói và chuyển đổi thành văn bản.  from gtts import gTTS  # Chuyển văn bản thành giọng nói (text-to-speech).  import playsound  # Phát file âm thanh (mp3).  # tkinter  import tkinter as tk  # Tạo giao diện đồ họa (GUI).  from tkinter import messagebox  # Hiển thị hộp thoại thông báo trong GUI.  # Thư viện OS (lập thư mục, files)  import os  # Làm việc với hệ thống file: tạo thư mục, lưu file, v.v.  # Nạp các thư viện cần thiết EDA  import numpy as np  # Thư viện cho các phép toán số học và mảng.  import pandas as pd  # Xử lý và phân tích dữ liệu dạng bảng (DataFrame).  from scipy import stats  # Các công cụ thống kê và phân tích toán học.  from sklearn import preprocessing  # Tiền xử lý dữ liệu (chuẩn hóa, mã hóa, v.v.).  from sklearn.feature\_selection import SelectKBest, chi2  # Chọn các đặc trưng quan trọng từ dữ liệu. |

## 1.4 Bố cục của báo cáo

Báo cáo gồm những nội dung như sau:

Chương 1: Giới thiệu đồ án của Học phần

Chương 2: Các cơ sở lý thuyết của Học phần phục việc thực hiện đề tài

Chương 3: Phân tích và xác đinh các cơ sở kỹ thuật thực hiện đề tài

Chương 4: Lập trình Phân tích thống kê về cầu thủ bóng rổ trong mùa giải NBA thường niên 2023-2024: NBAPlayer

Chương 5: Lập trình Python hỗ trợ CV nhận diện hình học cơ

Chương 6: Lập trình Game Racing Car

Chương 7: Tổng kết các kết quản đạt được và còn hạn chế của đồ án, đồng thời đề xuất hướng khắc phục hạn chế và phát triển Đồ án.

# CHƯƠNG 2: CƠ SỞ LÝ THUYẾT CỦA HỌC PHẦN

## 2.1 GIỚI THIỆU VỀ LẬP TRÌNH PYTHON

### 2.1.1 Giới thiệu chung

Python là một ngôn ngữ lập trình cao cấp, được thiết kế để dễ dàng học và sử dụng. Được phát triển bởi Guido van Rossum vào năm 1991, Python được biết đến với cú pháp rõ ràng, dễ đọc và dễ bảo trì. Python hỗ trợ lập trình hướng đối tượng, lập trình hàm và lập trình thủ tục, cho phép người dùng chọn lựa cách thức phát triển chương trình theo ý muốn. Python được sử dụng rộng rãi trong các lĩnh vực như phát triển web, phân tích dữ liệu, học máy, tự động hóa, và phát triển ứng dụng di động. Với cộng đồng phát triển mạnh mẽ và thư viện phong phú, Python trở thành một trong những ngôn ngữ lập trình phổ biến nhất trên thế giới.

### 2.1.2 Cài đặt “bộ thảo chương” (soạn thảo chương trình IDE) và “bộ dịch”

Để lập trình Python, người lập trình cần cài đặt một "bộ thảo chương" (IDE - Integrated Development Environment) và một "bộ dịch" (Interpreter). Bộ thảo chương giúp quá trình viết mã trở nên dễ dàng và hiệu quả hơn với các tính năng hỗ trợ như tô màu cú pháp, gợi ý mã, kiểm tra lỗi, và tự động hoàn thành. Một trong những IDE phổ biến cho Python là Visual Studio Code (VSCode). VSCode không chỉ hỗ trợ Python mà còn cung cấp các công cụ hữu ích khác như tích hợp Git, trình gỡ lỗi, và các tiện ích mở rộng giúp tối ưu hóa việc phát triển ứng dụng.

Để sử dụng Python trên VSCode, người dùng cần cài đặt Python extension trong VSCode, giúp nhận diện cú pháp Python, hỗ trợ chạy mã và gỡ lỗi chương trình trực tiếp trong IDE. Bên cạnh đó, bộ dịch Python (Python Interpreter) là phần mềm chịu trách nhiệm biên dịch mã nguồn Python thành mã máy để chương trình có thể thực thi. Khi cài đặt Python trên máy tính, bộ dịch này sẽ được cài sẵn, và VSCode có thể tự động nhận diện phiên bản Python đã cài để chạy chương trình. Khi viết mã trên VSCode, người lập trình có thể dễ dàng chạy mã Python và nhận kết quả ngay lập tức, tạo điều kiện thuận lợi cho việc phát triển phần mềm.

### 2.1.3. Giới thiệu các bộ tương tác lập trình

Bộ tương tác lập trình (Interactive Development Environments - IDEs) là các công cụ cho phép lập trình viên viết mã và chạy trực tiếp các lệnh Python ngay lập tức để xem kết quả. Các công cụ này rất hữu ích khi muốn thử nghiệm các đoạn mã nhỏ hoặc kiểm tra kết quả của từng lệnh mà không cần phải tạo một chương trình hoàn chỉnh. Một ví dụ điển hình là Python Shell, nơi người dùng có thể nhập lệnh Python và nhận kết quả ngay lập tức. Ngoài ra, Jupyter Notebook cũng là một công cụ phổ biến trong cộng đồng lập trình Python, đặc biệt là trong các lĩnh vực phân tích dữ liệu và học máy, vì nó cho phép tích hợp văn bản, mã nguồn và đồ họa trong một môi trường làm việc duy nhất.

Trong VSCode, mặc dù IDE này chủ yếu được sử dụng để phát triển các dự án lớn, nhưng bạn cũng có thể sử dụng terminal tích hợp để chạy các đoạn mã Python nhanh chóng hoặc sử dụng tính năng Interactive Window để thử nghiệm mã trong một môi trường tương tác ngay bên trong IDE.

### 2.1.4. Cấu hình môi trường tương tác lập trình

Để sử dụng môi trường tương tác lập trình hiệu quả trong VSCode, người lập trình cần cấu hình các công cụ cần thiết. Đầu tiên, bạn cần cài đặt Python trên máy tính nếu chưa có, sau đó cài đặt extension Python cho VSCode từ marketplace. Extension này cung cấp các tính năng hỗ trợ như tô màu cú pháp, gợi ý mã, tự động hoàn thành và đặc biệt là khả năng chạy mã Python trực tiếp trong IDE.

VSCode cũng cho phép sử dụng Python Interactive Window để chạy các đoạn mã Python theo dạng từng khối (cell), giống như cách hoạt động của Jupyter Notebook. Để sử dụng tính năng này, bạn chỉ cần mở một file Python và nhấn Shift + Enter để chạy các đoạn mã từng phần. Bên cạnh đó, bạn có thể sử dụng terminal tích hợp của VSCode để chạy Python Shell, nơi bạn có thể nhập các lệnh Python và nhận kết quả ngay lập tức. Việc cấu hình này giúp bạn linh hoạt hơn trong quá trình phát triển và thử nghiệm mã, mang lại trải nghiệm lập trình mượt mà và hiệu quả.

## 2.2 TỔNG QUAN VỀ NGÔN NGỮ LẬP TRÌNH PYTHON

### 2.2.1 Danh hiệu (Identifier)

Danh hiệu trong Python là tên dùng để xác định các đối tượng như biến, hàm, lớp, v.v. Các danh hiệu phải bắt đầu bằng chữ cái hoặc dấu gạch dưới và có thể chứa chữ cái, chữ số hoặc dấu gạch dưới.

### 2.2.2 Về viết mã lệnh lập trình (Code)

Mã lệnh trong Python được viết bằng cú pháp rõ ràng và dễ hiểu, giúp lập trình viên thực hiện các tác vụ từ cơ bản đến phức tạp. Cú pháp của Python hỗ trợ cả lập trình hướng đối tượng lẫn thủ tục.

### 2.2.3 Về chú giải (Comment)

Chú giải trong Python được sử dụng để ghi chú giải thích mã lệnh. Chúng không được thực thi và giúp lập trình viên hiểu rõ hơn về các đoạn mã, đồng thời giúp cải thiện việc bảo trì mã nguồn.

### 2.2.4 Các phép toán

Python hỗ trợ nhiều phép toán cơ bản như cộng, trừ, nhân, chia, phép toán so sánh, phép toán logic và phép toán bit, cho phép lập trình viên thực hiện các tính toán và thao tác trên dữ liệu.

### 2.2.5 Chuỗi

Chuỗi trong Python là tập hợp các ký tự được bao quanh bởi dấu nháy đơn hoặc nháy kép. Các chuỗi có thể được thao tác bằng các phương thức như cắt, nối, tìm kiếm.

### 2.2.6 Lệnh

Lệnh trong Python là các câu lệnh được lập trình viên viết để thực hiện một hành động cụ thể. Các lệnh có thể là khai báo biến, thực thi phép toán hoặc gọi hàm.

### 2.2.7 Định nghĩa hàm

Hàm trong Python là một đoạn mã có thể tái sử dụng, thực hiện một tác vụ nhất định. Hàm được định nghĩa bằng từ khóa def, và có thể nhận đối số đầu vào và trả về kết quả.

### 2.2.8 Danh **sách**

Danh sách là một cấu trúc dữ liệu trong Python, cho phép lưu trữ các giá trị theo thứ tự. Danh sách có thể chứa các phần tử có kiểu dữ liệu khác nhau và hỗ trợ các thao tác như thêm, xóa, truy cập phần tử.

### 2.2.9 Bộ dữ liệu: Tuple

Tuple là một kiểu dữ liệu bất biến trong Python, dùng để lưu trữ một tập hợp các phần tử. Khác với danh sách, các phần tử trong tuple không thể thay đổi sau khi đã được tạo ra.

### 2.2.10 Tập hợp: Set

Tập hợp trong Python là một kiểu dữ liệu không chứa các phần tử trùng lặp và không có thứ tự. Set hỗ trợ các phép toán tập hợp như giao, hợp và hiệu.

### 2.2.11 Từ điển: Dictionary

Từ điển là một cấu trúc dữ liệu trong Python, lưu trữ các cặp khóa-giá trị. Các giá trị có thể được truy xuất nhanh chóng thông qua khóa, và từ điển hỗ trợ các thao tác thêm, sửa, xóa các cặp khóa-giá trị.

### 2.2.12 Module

Module trong Python là một tập hợp các hàm, lớp và biến liên quan được tổ chức trong một file duy nhất. Chúng giúp tái sử dụng mã nguồn và tổ chức chương trình dễ dàng hơn.

### 2.2.13. Package

Package là một cách tổ chức các module trong Python, cho phép nhóm các module lại với nhau trong một thư mục để quản lý mã nguồn dễ dàng hơn, đặc biệt khi ứng dụng phát triển lớn hơn.

### 2.2.14 Lớp

Lớp trong Python là một khuôn mẫu để tạo đối tượng. Lớp định nghĩa các thuộc tính và phương thức mà đối tượng của lớp đó có thể sở hữu, hỗ trợ lập trình hướng đối tượng.

### 2.2.15 Xử lý ngoại lệ

Xử lý ngoại lệ trong Python là cách để quản lý các lỗi trong quá trình thực thi chương trình. Python cung cấp các cấu trúc try, except để bắt và xử lý các lỗi xảy ra, giúp chương trình tiếp tục chạy mà không bị dừng đột ngột.

## 2.3 GIỚI THIỆU CÁC THƯ VIỆN VÀ CÔNG CỤ CĂN BẢN

### 2.3.1 Nạp thư viện

Trong Python, thư viện được nạp vào chương trình bằng câu lệnh import. Việc nạp thư viện giúp lập trình viên sử dụng các chức năng và công cụ có sẵn trong thư viện đó mà không cần phải viết lại từ đầu. Thư viện Python có thể được nạp toàn bộ hoặc chỉ một phần cụ thể tùy theo nhu cầu của người dùng.

### 2.3.2 speech\_recognition

Thư viện speech\_recognition cung cấp các công cụ để nhận dạng giọng nói và chuyển đổi thành văn bản. Nó hỗ trợ nhiều công cụ nhận dạng âm thanh như Google Web Speech API, Microsoft Bing Voice Recognition, và CMU Sphinx, giúp dễ dàng tích hợp khả năng nhận diện giọng nói vào các ứng dụng.

### 2.3.3 gtts

Thư viện gtts (Google Text-to-Speech) là một thư viện Python giúp chuyển đổi văn bản thành giọng nói. Thư viện này sử dụng dịch vụ Google Text-to-Speech API để tạo âm thanh từ văn bản, hỗ trợ nhiều ngôn ngữ và giọng nói khác nhau.

### 2.3.4 playsound

Thư viện playsound là một thư viện đơn giản giúp phát các tệp âm thanh trong Python. Nó hỗ trợ các định dạng tệp âm thanh phổ biến như MP3 và WAV, rất hữu ích trong các ứng dụng yêu cầu phát âm thanh đơn giản mà không cần nhiều cấu hình.

### 2.3.5 tkinter

Thư viện tkinter là thư viện chuẩn của Python dùng để xây dựng giao diện đồ họa (GUI). Với tkinter, người lập trình có thể dễ dàng tạo ra các cửa sổ, nút bấm, biểu mẫu, và các yếu tố giao diện người dùng khác, giúp xây dựng ứng dụng dễ sử dụng và tương tác.

### 2.3.6 os

Thư viện os cung cấp các công cụ để làm việc với hệ điều hành, bao gồm các thao tác với file, thư mục, và các lệnh hệ thống. Thư viện này rất hữu ích trong việc tự động hóa các tác vụ liên quan đến hệ thống như tạo, xóa, và di chuyển các tệp tin.

### 2.3.7 Pandas

Pandas là thư viện mạnh mẽ và phổ biến trong Python, được sử dụng để xử lý và phân tích dữ liệu. Pandas cung cấp các cấu trúc dữ liệu như DataFrame và Series, giúp lưu trữ và thao tác với dữ liệu dễ dàng và hiệu quả. Đây là công cụ chính trong nhiều lĩnh vực như khoa học dữ liệu và phân tích thống kê.

### 2.3.8 Tensorflow

TensorFlow là một thư viện mã nguồn mở mạnh mẽ được Google phát triển, chuyên dùng để xây dựng và triển khai các mô hình học sâu (deep learning). TensorFlow hỗ trợ các mô hình học máy phức tạp như mạng nơ-ron, mạng CNN, và mạng RNN, và được ứng dụng rộng rãi trong các bài toán như nhận dạng hình ảnh, phân tích ngữ nghĩa văn bản, và dự đoán.

### 2.3.9 NumPy

NumPy là thư viện nổi bật trong Python cho phép thực hiện các phép toán ma trận và các phép toán số học với dữ liệu. NumPy cung cấp cấu trúc dữ liệu mảng đa chiều (ndarray) cùng nhiều hàm toán học, giúp xử lý dữ liệu số nhanh chóng và hiệu quả.

### 2.3.10 SCIPY

SciPy là một thư viện mở rộng của NumPy, cung cấp các công cụ tính toán khoa học phức tạp hơn như tích phân, tối ưu hóa, và phân tích thống kê. SciPy thường được sử dụng trong các bài toán tính toán kỹ thuật, mô phỏng và xử lý tín hiệu.

### 2.3.11 Theano

Theano là một thư viện Python được sử dụng chủ yếu để tính toán các biểu thức toán học phức tạp, đặc biệt là trong học sâu. Theano hỗ trợ tính toán hiệu quả trên GPU, giúp giảm thời gian xử lý trong các mô hình học máy và học sâu.

### 2.3.12 Matplotlib

Matplotlib là một thư viện Python chuyên vẽ đồ thị và trực quan hóa dữ liệu. Với Matplotlib, người lập trình có thể tạo các biểu đồ, đồ thị tuyến tính, đồ thị phân tán, biểu đồ thanh, và nhiều dạng đồ thị khác, giúp hiển thị kết quả phân tích dữ liệu một cách trực quan.

### 2.3.13 scikit-learn = sklearn

Scikit-learn (hay sklearn) là thư viện Python cung cấp các công cụ mạnh mẽ để xây dựng và triển khai các mô hình học máy. Nó hỗ trợ các thuật toán phân loại, hồi quy, phân nhóm và giảm chiều dữ liệu, giúp đơn giản hóa quá trình tạo ra các mô hình dự đoán và phân tích.

### 2.3.14 Keras

Keras là một thư viện mã nguồn mở dùng để xây dựng và huấn luyện các mô hình học sâu. Keras hoạt động trên nền tảng của TensorFlow và giúp phát triển các mạng nơ-ron sâu (deep neural networks) dễ dàng và nhanh chóng hơn với giao diện đơn giản và dễ sử dụng.

### 2.3.15 NLTK

NLTK (Natural Language Toolkit) là một thư viện Python chuyên dùng cho xử lý ngôn ngữ tự nhiên (NLP). NLTK cung cấp các công cụ và tài nguyên để xử lý văn bản, phân tích ngữ nghĩa, tách từ, và phân tích cú pháp, phục vụ cho các ứng dụng như chatbot và phân tích ngữ nghĩa văn bản.

### 2.3.16 PyTorch

PyTorch là một thư viện học sâu mã nguồn mở phát triển bởi Facebook, nổi bật với tính linh hoạt và dễ sử dụng. PyTorch hỗ trợ cả đào tạo và triển khai các mô hình học sâu, đặc biệt trong các bài toán nhận dạng hình ảnh, xử lý ngữ âm, và phân tích ngữ nghĩa.

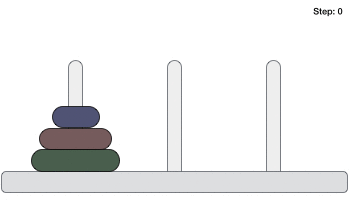
### 2.3.17 LightGBM

LightGBM (Light Gradient Boosting Machine) là thư viện học máy được thiết kế để thực hiện các mô hình boosting cây quyết định với hiệu suất cao. LightGBM thường được sử dụng trong các bài toán phân loại và hồi quy với tập dữ liệu lớn nhờ khả năng tính toán nhanh và tiết kiệm bộ nhớ.

### 2.3.18 Eli5

Eli5 là thư viện Python hỗ trợ giải thích các mô hình học máy. Với Eli5, lập trình viên có thể dễ dàng kiểm tra độ quan trọng của các tính năng trong mô hình, hiểu cách thức hoạt động của các thuật toán học máy, và giải thích các dự đoán của mô hình cho người dùng cuối.

## 2.4 KỸ THUẬT LẬP TRÌNH ĐỆ QUY THÁP HÀ NỘI (HANOITOWER)



Kỹ thuật lập trình đệ quy tháp Hà Nội (Hanoi Tower) là một bài toán cổ điển trong lập trình, yêu cầu người lập trình sử dụng đệ quy để giải quyết vấn đề di chuyển các đĩa từ cột này sang cột khác, tuân thủ các quy tắc nhất định. Bài toán này không chỉ giúp rèn luyện tư duy thuật toán mà còn có ứng dụng thực tế trong việc tối ưu hóa các vấn đề có tính chất đệ quy.

## 2.5 LẬP TRÌNH PYTHON XỬ LÝ GIỌNG NÓI (Trợ lý ảo: Voice Assistant)

Lập trình xử lý giọng nói trong Python thường sử dụng các thư viện như speech\_recognition, pyttsx3 và pyaudio. speech\_recognition cho phép nhận diện và chuyển đổi giọng nói thành văn bản, trong khi pyttsx3 hỗ trợ việc chuyển văn bản thành giọng nói. Quá trình lập trình thường bắt đầu với việc cấu hình micro và nhận diện lệnh giọng nói từ người dùng, sau đó phân tích các lệnh và thực hiện các hành động tương ứng. Điều này đụng đến việc tích hợp với các API và dịch vụ bên ngoài, ví dụ như Google Speech-to-Text API để tăng cường khả năng nhận diện chính xác.

## 2.6 LẬP TRÌNH GIAO DIỆN ĐỒ HỌA (GUI)

Lập trình giao diện đồ họa trong Python chủ yếu sử dụng các thư viện như Tkinter, PyQt, hoặc Kivy. Các thư viện này cho phép tạo ra các ứng dụng với giao diện người dùng trực quan, giúp người dùng tương tác dễ dàng với phần mềm. Cơ bản, lập trình viên sẽ tạo các cửa sổ (windows), thẻ điều khiển (widgets) như nút bấm (buttons), thanh cuộn (scrollbars), và các ô nhập liệu (entry widgets). Quá trình lập trình GUI bao gồm việc thiết kế giao diện và lập trình các sự kiện (event-driven programming), chẳng hạn như khi người dùng nhấn nút, chương trình sẽ phản hồi bằng cách thực hiện một tác vụ cụ thể.

## 2.7 LẬP TRÌNH PHÂN TÍCH KHÁM PHÁ (EDA) [Thăm dò]

Phân tích khám phá dữ liệu (EDA) là bước quan trọng trong việc hiểu rõ đặc điểm của bộ dữ liệu trước khi thực hiện các phân tích sâu hơn. Trong trường hợp phân tích thống kê về các cầu thủ NBA mùa giải 2023-2024, EDA giúp khám phá các thông tin cơ bản như điểm số trung bình, số lần ghi bàn, số lần kiến tạo, số phút thi đấu và các chỉ số quan trọng khác của các cầu thủ trong suốt mùa giải.

Sử dụng Python, chúng ta có thể áp dụng các thư viện như pandas để xử lý và làm sạch dữ liệu, matplotlib và seaborn để trực quan hóa các mối quan hệ giữa các chỉ số khác nhau, như mối tương quan giữa điểm số và số phút thi đấu. Các bước EDA này giúp lập trình viên hiểu được phân phối dữ liệu (ví dụ, điểm số của các cầu thủ phân bố như thế nào), phát hiện các giá trị bất thường (outliers), và nhận diện các xu hướng thú vị trong mùa giải. Cùng với đó, việc tạo ra các biểu đồ như histogram, scatter plots, box plots và heatmap sẽ giúp trực quan hóa mối quan hệ giữa các biến, từ đó hỗ trợ đưa ra các kết luận sâu sắc về hiệu suất của các cầu thủ NBA trong mùa giải 2023-2024.

Quá trình EDA trong phân tích dữ liệu NBA sẽ giúp xây dựng nền tảng vững chắc cho các bước phân tích tiếp theo, chẳng hạn như dự đoán hiệu suất cầu thủ hoặc phân nhóm cầu thủ theo các chỉ số thống kê quan trọng.

## 2.8 LẬP TRÌNH ĐỒ THỊ & BIỂU ĐỒ TRỰC QUAN HÓA DỮ LIỆU (VISUALIZAION: PLOT)

Việc trực quan hóa dữ liệu là một phần quan trọng trong phân tích dữ liệu. Trong Python, thư viện matplotlib là công cụ chính để tạo các biểu đồ như đường (line chart), cột (bar chart), biểu đồ tròn (pie chart), và phân tán (scatter plot). Các biểu đồ này giúp lập trình viên và người dùng dễ dàng nhận diện các xu hướng, mẫu hình, và mối quan hệ trong dữ liệu. Kỹ thuật này không chỉ giúp việc phân tích dễ dàng hơn mà còn giúp truyền đạt thông tin phức tạp một cách dễ hiểu.

## 2.9 LẬP TRÌNH HỖ TRỢ KỸ THUẬT THỊ GIÁC MÁY TÍNH (CV)

Kỹ thuật thị giác máy tính (Computer Vision - CV) trong Python chủ yếu sử dụng thư viện OpenCV, giúp máy tính nhận diện và xử lý hình ảnh và video. Các ứng dụng điển hình của CV bao gồm nhận diện khuôn mặt, phân loại đối tượng, phát hiện chuyển động và nhận diện văn bản từ hình ảnh (OCR). Quá trình lập trình thị giác máy tính yêu cầu xử lý và phân tích ảnh thông qua các kỹ thuật như biến đổi hình ảnh, lọc và phát hiện các đặc trưng. Python cung cấp nhiều công cụ mạnh mẽ như OpenCV, TensorFlow, và PyTorch để phát triển các mô hình học sâu trong thị giác máy tính

## 2.10 LẬP TRÌNH GAMEs CĂN BẢN

Lập trình game đua xe trong Python có thể được thực hiện thông qua thư viện Pygame, giúp tạo ra một trò chơi đua xe 2D cơ bản. Trong quá trình phát triển game, lập trình viên sẽ phải xây dựng các yếu tố cơ bản như các đối tượng xe, đường đua, và các vật cản. Các đối tượng (sprites) như xe đua cần được lập trình để di chuyển trên đường, điều khiển bằng các phím bấm, và xử lý va chạm với các vật cản hoặc biên giới của đường đua.

Lập trình game đua xe yêu cầu thiết kế hệ thống sự kiện (event-driven programming) để xử lý các lệnh điều khiển từ người chơi, như việc di chuyển xe, tăng tốc, và phanh. Thêm vào đó, các yếu tố như thời gian đua, điểm số và mức độ khó của game có thể được lập trình để nâng cao trải nghiệm của người chơi. Hệ thống âm thanh như tiếng động cơ, tiếng va chạm cũng có thể được tích hợp để tăng tính sống động cho game.

# CHƯƠNG 3: PHÂN TÍCH VÀ XÁC ĐINH CÁC CƠ SỞ KỸ THUẬT

## 3.1 GIỚI THIỆU CHUNG VỀ CÁC CHỦ ĐỀ CỦA ĐỒ ÁN

Đồ án này gồm 3 chủ đề chính:

+ Lập trình EDA thăm dò phân tích thống kê về cầu thủ NBA tại giải bóng rổ NBA từ 2003 đến 2024 [0]; có trực quan hóa dữ liệu dùng Plot vẽ đồ thị và biểu đồ [3.5].

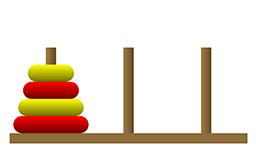
+ Lập trình Python hỗ trợ CV nhận diện hình học cơ bản [Chương 5:]

+ : Lập trình Python Game Car Racing [0]

Được thực hiện trên hệ thống giao diện GUI [3.4] với cơ chế nhập thông tin vào (input) và xuất kết quả (output) theo hình thức Text và Voice nhờ xử lý “Trợ lý ảo” [3.2].

## 3.2 KỸ THUẬT LẬP TRÌNH ĐỆ QUY: THÁP HÀ NỘI (HANOI TOWER)

### 3.2.1 Mô tả bài toán



Chỉ 1 dòng đĩa đến n đĩa từ cặp thứ 1 đến cặp thứ 3 n là số nguyên > o

Điều kiện 1: chỉ sử dụng 1 cột trung gian là cặp 2

Điều kiện 2: mỗi lần chỉ chuyển 1 đĩa duy nhất

Điều kiện 3: đĩa lớn luôn ở dưới, đĩa nhỏ luôn ở trên

### 3.2.2 Các yêu cầu bài làm

- Cá nhân hóa thông tin bài làm đối với các Biến, Hàm, Hằng, Bí danh,...

thông tin cá nhân hóa = sv tự chọn, VD; stt, tên, họ, viết tắt họ tên,...

- Cho NSD chọn số lương "Đĩa" => Comment # cho biết số "đĩa" tối đa chạy được

n = int(input("Quý vị nhap so Dia: ")) kiểu int

- In ra thời gian chạy ? import time

- Kiểm tra số lượng đĩa nhập trong #2 hợp lệ: n >0 && số nguyên [nên lặp = có thoát]

- IN RA SỐ THỨ TỰ LƯỢT CHUYỂN "ĐĨA"

### 3.2.3 Full Codes (mã lệnh lập trình)

|  |
| --- |
| def thapHaNoi\_CTNP\_27(*n*, *toaMot*, *toaHai*, *toaBa*):      if n == 1:          print("Chuyển từ", toaMot, "sang", toaBa)      else:          thapHaNoi\_CTNP\_27(n - 1, toaMot, toaBa, toaHai)          print("Chuyển từ", toaMot, "sang", toaBa)          thapHaNoi\_CTNP\_27(n - 1, toaHai, toaMot, toaBa)  # Nhập số đĩa từ người dùng  n = *int*(input("Nhập số đĩa: "))  # Gọi hàm tháp Hà Nội với các trụ  thapHaNoi\_CTNP\_27(n, "1", "2", "3") |

### 3.2.4 Kết quả thực nghiệm

+ Giá trị thực nghiệm: số đĩa n = 8

KẾT QUẢ:

|  |
| --- |
| Nhập số đĩa: 5  Chuyển từ 1 sang 3  Chuyển từ 1 sang 2  Chuyển từ 3 sang 2  Chuyển từ 1 sang 3  Chuyển từ 2 sang 1  Chuyển từ 2 sang 3  Chuyển từ 1 sang 3  Chuyển từ 1 sang 2  Chuyển từ 3 sang 2  Chuyển từ 3 sang 1  Chuyển từ 2 sang 1  Chuyển từ 3 sang 2  Chuyển từ 1 sang 3  Chuyển từ 1 sang 2  Chuyển từ 3 sang 2  Chuyển từ 1 sang 3  Chuyển từ 2 sang 1  Chuyển từ 2 sang 3  Chuyển từ 1 sang 3  Chuyển từ 2 sang 1  Chuyển từ 3 sang 2  Chuyển từ 3 sang 1  Chuyển từ 2 sang 1  Chuyển từ 2 sang 3  Chuyển từ 1 sang 3  Chuyển từ 1 sang 2  Chuyển từ 3 sang 2  Chuyển từ 1 sang 3  Chuyển từ 2 sang 1  Chuyển từ 2 sang 3  Chuyển từ 1 sang 3 |

### 3.2.5 Các bài tập tương tự

#### 3.2.5.1 Bài 1.1

In danh sách 100 số nguyên tố đầu tiên (chỉ chia hết cho 1 và chính nó: 2, ,3, 5, 7, 11, 13, 17,…)

|  |
| --- |
| def is\_prime\_27NgocPhung(*n*): # Kiểm tra nếu n nhỏ hơn 2, vì các số nguyên tố phải lớn hơn hoặc bằng 2      if n < 2:          return False      # Kiểm tra chia hết từ 2 đến căn bậc 2 của n (vì nếu n có ước nhỏ hơn căn bậc 2, ước lớn sẽ tìm thấy ở phía còn lại)      for i in range(2, *int*(n\*\*0.5) + 1): # Nếu n chia hết cho i, nghĩa là n không phải là số nguyên tố          if n % i == 0:              return False      return True # Nếu không có ước nào chia hết, tức là n là số nguyên tố  def list\_first\_100\_primes\_27NgocPhung():      primes = [] # Danh sách lưu các số nguyên tố      num = 2 # Bắt đầu từ số 2      # Tiếp tục cho đến khi tìm được 100 số nguyên tố      while len(primes) < 100:          if is\_prime\_27NgocPhung(num): # Kiểm tra xem num có phải là số nguyên tố không              primes.append(num) # Nếu là số nguyên tố, thêm vào danh sách primes          num += 1 # Tăng num lên 1 để kiểm tra số tiếp theo      return primes # Trả về danh sách các số nguyên tố đầu tiên  print(list\_first\_100\_primes\_27NgocPhung()) |

Kết quả thực nghiệm:

|  |
| --- |
| [2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97, 101, 103, 107, 109, 113, 127, 131, 137, 139, 149, 151, 157, 163, 167, 173, 179, 181, 191, 193, 197, 199, 211, 223, 227, 229, 233, 239, 241, 251, 257, 263, 269, 271, 277, 281, 283, 293, 307, 311, 313, 317, 331, 337, 347, 349, 353, 359, 367, 373, 379, 383, 389, 397, 401, 409, 419, 421, 431, 433, 439, 443, 449, 457, 461, 463, 467, 479, 487, 491, 499, 503, 509, 521, 523, 541] |

#### 3.2.5.2 Bài 1.2

Viết hàm đảo ngược số

|  |
| --- |
| # Định nghĩa hàm với tham số n, đại diện cho số nguyên cần đảo ngược  def reverse\_number\_27\_NgocPhung(*n*):      # Chuyển n thành chuỗi (str), sau đó đảo ngược chuỗi bằng cú pháp [::-1],      # rồi chuyển ngược lại thành số nguyên (int)      reversed\_num = *int*(*str*(n)[::-1])      # Trả về số nguyên đã được đảo ngược      return reversed\_num  # Gọi hàm và in ra kết quả đảo ngược của số 123 (kết quả là 321)  print(reverse\_number\_27\_NgocPhung(123)) |

Kết quả thực nghiệm:

|  |
| --- |
| 321 |

#### 3.2.5.3 Bài 1.3

Giải phương trình bậc nhất và bậc hai

|  |
| --- |
| import math  # Hàm giải phương trình bậc nhất ax + b = 0  def solve\_linear\_equation\_27NgocPhung(*a*, *b*):      if a == 0:          return "Vô nghiệm" if b != 0 else "Vô số nghiệm"      return -b / a  # Hàm giải phương trình bậc hai ax^2 + bx + c = 0  def solve\_quadratic\_equation\_27NgocPhung(*a*, *b*, *c*):      if a == 0:      # Nếu a = 0, phương trình trở thành bậc nhất          return solve\_linear\_equation\_27NgocPhung(b, c)      delta = b\*\*2 - 4 \* a \* c  # Tính biệt thức delta = b^2 - 4ac      if delta < 0:      # Nếu delta < 0, phương trình vô nghiệm (không có nghiệm thực)          return "Vô nghiệm"      elif delta == 0:      # Nếu delta = 0, phương trình có nghiệm kép          return -b / (2 \* a)      else:      # Nếu delta > 0, phương trình có 2 nghiệm phân biệt          sqrt\_delta = math.sqrt(delta)          x1 = (-b + sqrt\_delta) / (2 \* a)          x2 = (-b - sqrt\_delta) / (2 \* a)          return x1, x2  # Ví dụ sử dụng  print(solve\_linear\_equation\_27NgocPhung(2, -4))   # Giải phương trình 2x - 4 = 0, kết quả: 2  print(solve\_quadratic\_equation\_27NgocPhung(1, -3, 2))  # Giải phương trình x^2 - 3x + 2 = 0, kết quả: (2.0, 1.0) |

Kết quả thực nghiệm

|  |
| --- |
| 2.0  (2.0, 1.0) |

#### 3.2.5.4 Bài 1.4

|  |
| --- |
| def number\_to\_words\_27NgocPhung(*n*):      words = {  # Định nghĩa một từ điển chứa các chữ số từ 0 đến 9 và cách gọi trong tiếng Việt          '0': "Không", '1': "Một", '2': "Hai", '3': "Ba", '4': "Bốn",          '5': "Năm", '6': "Sáu", '7': "Bảy", '8': "Tám", '9': "Chín"}      # Dùng list comprehension để chuyển mỗi chữ số của n thành từ tiếng Việt tương ứng      # str(n) chuyển số n thành chuỗi, rồi ta duyệt qua từng ký tự (chữ số) trong chuỗi đó      # Mỗi chữ số được tra cứu trong từ điển words và chuyển thành từ tiếng Việt      return " ".join([words[digit] for digit in *str*(n)])  # Nối các từ lại với nhau bằng khoảng trắng  print(number\_to\_words\_27NgocPhung(123))  # Gọi hàm với tham số là 123 và in kết quả |

Kết quả thực nghiệm

|  |
| --- |
| Một Hai Ba |

## 3.3 LẬP TRÌNH PYTHON XỬ LÝ GIỌNG NÓI

### 3.3.1 Giới thiệu bài toán

Chủ đề lập trình python này là cơ sở của các ứng dụng về AI, ML, Deep ML,…

+ Các hệ thống này thường có 3 phần :

Part1 Input: Text, GUI, Speech….: chính là part1 của chủ đề này

Part các nội dung chuyên môn khác ..: Các kỹ thuật AI,…

Part2 Output: Voice, Text, GUI….: chính là part2 của chủ đề này

### 3.3.2 Các yêu cầu bài làm

**Yêu cầu bài tập**: **B2Ex2\_VoiceAssistant**

**Cá nhân hóa thông tin**:

+ Các biến, hàm, hằng số và bí danh có thể được cá nhân hóa theo thông tin của sinh viên (ví dụ: số thứ tự, tên, họ, viết tắt họ tên, ...).

**Phần 1: Nói -> Text**

1. Cho người sử dụng (NSD) chọn cách nhập âm thanh: Từ micro (MIC) hoặc từ file (.mp3, .wav).
2. Cho NSD chọn thời gian chờ (kiểm tra nhập từ 0 trở lên).
3. Cho NSD chọn thời gian nghe (kiểm tra nhập từ 1 trở lên).
4. Cho NSD chọn ngôn ngữ sử dụng (2 ký tự ngôn ngữ, ví dụ: vi, en).
5. Phát âm thanh song song với văn bản khi yêu cầu nhập lựa chọn.

**Phần 2: Text -> Nói**

1. Cho NSD nhập tên file lưu âm thanh:

Kiểm tra xem file đã tồn tại trong thư mục hiện tại chư

Nếu chưa, lưu file

Nếu đã có, yêu cầu NSD chọn phương án xử lý (sử dụng thư viện os):

Lưu chồng lên file cũ.

Tự động đổi tên file cũ theo định dạng yymmddhhmmss (sử dụng thư viện time), và thông báo cho NSD.

Tự động đổi tên file mới theo định dạng yymmddhhmmss (sử dụng thư viện time), và thông báo cho NSD.

Cho phép NSD tự đổi tên file (đảm bảo tên file mới phải khác tên file cũ).

### 3.3.3 Các thư viện Python liên quan

speech\_recognition: nhận diện giọng nói

gTTS: google Translate To Speech

playsound: thư viện phát ra âm thanh

Ngoài ra, thêm:

Os : thư viện hỗ trợ lưu file, mở file, đường dẫn,..

time: thư viện lấy thời gian = đồng hồ

google\_trans\_new: thư viện hỗ trợ mô phỏng Google Translate.

Pyaudio: thư viện cần thiết để sử dụng microphone

### 3.3.4 Full Codes (mã lệnh lập trình)

|  |
| --- |
| import speech\_recognition as sr  from gtts import gTTS  import playsound  r\_ngocphung\_27 = sr.Recognizer()  # Khởi tạo đối tượng Recognizer để nhận dạng giọng nói  with sr.Microphone() as Source\_ngocphung\_27:  # Mở microphone để nhận âm thanh      # Hiệu chỉnh mic để chuẩn bị nói      print("1. Chọn thoi gian cho là 1 s")      print("2. Chọn thoi gian cho là 2 s")      print("Bạn chọn 1 hay 2")      w\_ngocphung\_27 = *int*(input("Bạn chọn 1 hay 2: "))  # Chọn thời gian cho việc hiệu chỉnh nhiễu      print("Hiệu chỉnh nhiễu trước khi nói!")      r\_ngocphung\_27.adjust\_for\_ambient\_noise(Source\_ngocphung\_27, *duration*=w\_ngocphung\_27)  # Hiệu chỉnh nhiễu môi trường      print("Hiệu chỉnh nhiễu trước khi nói!")      r\_ngocphung\_27.adjust\_for\_ambient\_noise(Source\_ngocphung\_27, *duration*=1)  # Thực hiện lần nữa để ổn định môi trường      print("Nói tiếng Việt đi, sau 5s sẽ in ra văn bản!")      audio\_data\_ngocphung\_27 = r\_ngocphung\_27.record(Source\_ngocphung\_27, *duration*=5)  # Ghi âm lời nói trong 5 giây      print("KẾT QUẢ NHẬN DIỆN ..................")      try:          text\_ngocphung\_27 = r\_ngocphung\_27.recognize\_google(audio\_data\_ngocphung\_27, *language*="vi")  # Chuyển âm thanh thành văn bản      except:          text\_ngocphung\_27 = "Bạn nói gì nghe không rõ...!"  # Nếu không nhận diện được thì trả về lỗi      print("Bạn đã nói là: ", format(text\_ngocphung\_27))  # In ra văn bản nhận được  def main\_ngocphung\_27(*t\_ngocphung\_27*):  # Hàm chính để chuyển văn bản thành lời nói      ngocphung\_27 = gTTS(*text*=t\_ngocphung\_27, *lang*='vi')  # Sử dụng gTTS để chuyển văn bản thành tiếng nói      file\_name\_ngocphung\_27 = 'D:\PYPR (Python Programming)\G127CTNPNBAPlayer\B2Ex2\_VoiceAssistant\B2Ex2\_VoiceAssistant\_0.mp3'  # Đặt tên file lưu      ngocphung\_27.save(file\_name\_ngocphung\_27)  # Lưu file âm thanh      playsound.playsound(file\_name\_ngocphung\_27)  # Phát âm thanh vừa tạo  main\_ngocphung\_27(text\_ngocphung\_27)  # Gọi hàm để đọc văn bản đã nhận diện |

### 3.3.6 Kết quả thực nghiệm

A black screen with white text

Description automatically generated

File được in ra

A screenshot of a computer

Description automatically generated

### 3.3.7 Các bài toán tương tự

#### 3.3.7.1 B2Ex2\_VoiceAssistant\_1

Chương trình cho phép người dùng chọn nguồn âm thanh, có thể là từ một file âm thanh hoặc từ microphone. Khi người dùng chọn nguồn là microphone hoặc cung cấp một file âm thanh, chương trình sử dụng thư viện speech\_recognition để nhận diện giọng nói và chuyển đổi âm thanh thành văn bản. Tiếp theo, chương trình hỏi người dùng xem có muốn dịch văn bản từ tiếng Việt sang tiếng Anh hay không. Nếu người dùng chọn 'yes', chương trình sẽ sử dụng thư viện googletrans để dịch văn bản và hiển thị kết quả. Sau đó, chương trình yêu cầu người dùng quyết định có muốn chuyển văn bản thành giọng nói hay không. Nếu người dùng chọn 'yes', văn bản (bao gồm cả bản gốc và bản dịch) sẽ được chuyển thành âm thanh thông qua thư viện gTTS (Google Text-to-Speech) và lưu dưới dạng file MP3. Sau khi lưu file MP3, chương trình sẽ chuyển đổi file này thành file WAV chuẩn với tần số mẫu 16kHz và kênh mono (bằng thư viện pydub), đảm bảo định dạng âm thanh phù hợp với yêu cầu nhận diện giọng nói trong các ứng dụng sau này. Cuối cùng, tất cả các file âm thanh (cả MP3 và WAV) sẽ được lưu trữ tại thư mục chỉ định, đảm bảo rằng người dùng có thể dễ dàng truy cập và sử dụng lại chúng.

|  |
| --- |
| import speech\_recognition as sr  # Thư viện nhận diện giọng nói  import os  # Thư viện làm việc với hệ thống file  import time  # Thư viện làm việc với thời gian  from googletrans import Translator  # Thư viện dịch văn bản sử dụng Google Translate  from gtts import gTTS  # Thư viện chuyển văn bản thành giọng nói (text-to-speech)  from pydub import AudioSegment  # Nhập khẩu thư viện pydub  # Hàm chọn nguồn âm thanh (microphone hoặc file)  def choose\_audio\_source\_ngocphung\_27(*source*="microphone", *filename*=None):      if source == "file" and filename:  # Nếu chọn nguồn là file và có filename          return sr.AudioFile(filename)  # Trả về đối tượng AudioFile từ file âm thanh      elif source == "microphone":  # Nếu chọn nguồn là microphone          return sr.Microphone()  # Trả về đối tượng Microphone để thu âm từ microphone      else:  # Nếu nguồn không hợp lệ          raise *ValueError*("chỉ được nhập 'file' hoặc 'microphone'")  # Hàm nhận diện âm thanh từ nguồn (microphone hoặc file)  def recognize\_audio\_ngocphung\_27(*source*="microphone", *filename*=None):      recognizer = sr.Recognizer()  # Tạo một đối tượng Recognizer để nhận diện giọng nói      with choose\_audio\_source\_ngocphung\_27(source, filename) as source\_audio:  # Chọn nguồn âm thanh          print("Đang nghe... Vui lòng nói")  # Thông báo đang nghe          audio\_data = recognizer.listen(source\_audio)  # Lắng nghe và thu âm      try:          print("Nhận diện giọng nói...")  # Thông báo đang nhận diện giọng nói          text\_ngocphung\_27 = recognizer.recognize\_google(audio\_data, *language*="vi")  # Nhận diện giọng nói thành văn bản bằng Google Speech Recognition          return text\_ngocphung\_27  # Trả về văn bản nhận diện      except sr.UnknownValueError:  # Nếu không nhận diện được giọng nói          return "Không thể nhận diện giọng nói"  # Trả về thông báo lỗi      except sr.RequestError:  # Nếu có lỗi kết nối đến dịch vụ nhận diện          return "Lỗi kết nối với dịch vụ nhận diện"  # Trả về thông báo lỗi kết nối  # Hàm dịch văn bản từ ngôn ngữ nguồn (source\_lang) sang ngôn ngữ đích (target\_lang)  def translate\_text\_ngocphung\_27(*text\_ngocphung\_27*, *source\_lang*='vi', *target\_lang*='en'):      translator = Translator()  # Tạo đối tượng Translator để dịch văn bản      translated\_ngocphung\_27 = translator.translate(text\_ngocphung\_27, *src*=source\_lang, *dest*=target\_lang)  # Dịch văn bản      return translated\_ngocphung\_27.text  # Trả về văn bản đã dịch  # Hàm chuyển đổi từ MP3 sang WAV  def convert\_mp3\_to\_wav(*mp3\_file*, *wav\_file*):      sound = AudioSegment.from\_mp3(mp3\_file)  # Đọc file mp3      sound.export(wav\_file, *format*="wav")  # Chuyển đổi và lưu dưới định dạng wav      print(f"Đã chuyển đổi {mp3\_file} thành {wav\_file}")  # Hàm chuyển văn bản thành âm thanh (text-to-speech)  def text\_to\_speech\_ngocphung\_27(*text\_ngocphung\_27*, *filename*="output\_audio"):      tts = gTTS(*text*=text\_ngocphung\_27, *lang*="vi")  # Tạo đối tượng để chuyển văn bản thành giọng nói      timestamp = time.strftime("%Y%m%d%H%M%S")  # Lấy thời gian hiện tại để tạo tên file duy nhất      mp3\_file\_path = f"D:\\PYPR (Python Programming)\\Final Project\\Voice\\{filename}\_{timestamp}.mp3"  # Đường dẫn file MP3      wav\_file\_path = mp3\_file\_path.replace(".mp3", ".wav")  # Đường dẫn file WAV      if os.path.exists(mp3\_file\_path):  # Kiểm tra xem file MP3 đã tồn tại chưa          print("File đã tồn tại. Đổi tên hoặc xóa file cũ.")          os.rename(mp3\_file\_path, mp3\_file\_path.replace(".mp3", f"\_{timestamp}.mp3"))  # Đổi tên file nếu trùng tên      tts.save(mp3\_file\_path)  # Lưu âm thanh thành file MP3      print(f"Đã lưu âm thanh vào {mp3\_file\_path}")  # Thông báo đã lưu MP3      # Chuyển đổi file MP3 sang WAV và lưu      convert\_mp3\_to\_wav(mp3\_file\_path, wav\_file\_path)  # Chuyển MP3 thành WAV      print(f"Đã lưu âm thanh vào {wav\_file\_path}")  # Thông báo đã lưu WAV  # Hàm chính của trợ lý ảo (Voice Assistant)  def voice\_assistant\_ngocphung\_27():      print("Chọn nguồn âm thanh (file/microphone):")  # Yêu cầu người dùng chọn nguồn âm thanh      source\_ngocphung\_27 = input("Nhập 'file' hoặc 'microphone': ").lower()  # Lấy lựa chọn nguồn âm thanh từ người dùng      filename\_ngocphung\_27 = None  # Biến để lưu tên file (nếu nguồn là file)      if source\_ngocphung\_27 == "file":  # Nếu nguồn là file          filename\_ngocphung\_27 = input("Nhập tên file âm thanh (ví dụ: audio.wav): ")  # Yêu cầu người dùng nhập tên file âm thanh      text\_ngocphung\_27 = recognize\_audio\_ngocphung\_27(source\_ngocphung\_27, filename\_ngocphung\_27)  # Nhận diện âm thanh thành văn bản      print(f"Văn bản nhận diện: {text\_ngocphung\_27}")  # In ra văn bản nhận diện      print("Chuyển ngữ (Vi -> En)?")  # Hỏi người dùng có muốn dịch không      if input("Nhấn 'y' để tiếp tục, 'n' để dừng: ").lower() == 'y':  # Nếu người dùng chọn dịch          translated\_text\_ngocphung\_27 = translate\_text\_ngocphung\_27(text\_ngocphung\_27, *source\_lang*="vi", *target\_lang*="en")  # Dịch văn bản          print(f"Văn bản dịch: {translated\_text\_ngocphung\_27}")  # In ra văn bản đã dịch      print("Chuyển văn bản thành âm thanh? (y/n)")  # Hỏi người dùng có muốn chuyển văn bản thành âm thanh không      if input().lower() == 'y':  # Nếu người dùng chọn chuyển văn bản thành âm thanh          text\_to\_speech\_ngocphung\_27(text\_ngocphung\_27)  # Chuyển văn bản thành giọng nói  # Chạy ứng dụng Voice Assistant  voice\_assistant\_ngocphung\_27() |

##### 3.3.7.1 Trường hợp nhập ‘microphone’

A screen shot of a computer

Description automatically generated

🡪 4 file được xuất ra, 2 file tiếng việt (mp3,wav) và 2 file tiếng anh (mp3,wav)

A screenshot of a computer

Description automatically generated

##### 3.3.7.1 Trường hợp nhập ‘file’

A black background with white text

Description automatically generated

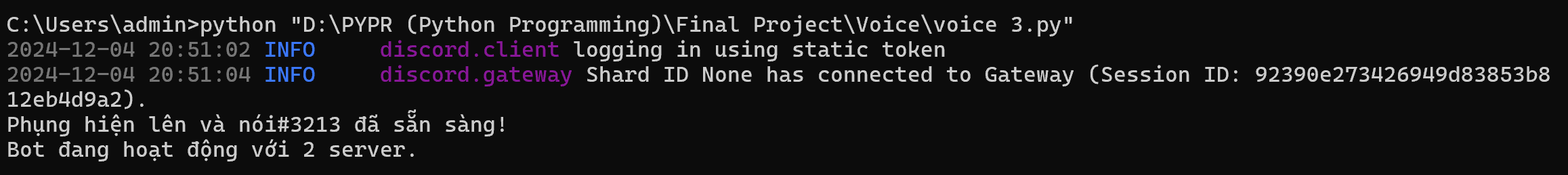
#### 3.3.7.2 B2Ex2\_VoiceAssistant\_2 [Ứng dụng vào phần mềm discord]

Bot Discord này được xây dựng để giúp người dùng giao tiếp với bot thông qua lệnh văn bản và âm thanh. Bot sử dụng thư viện discord.py để kết nối với server Discord và tham gia vào voice channel. Khi người dùng gửi tin nhắn chứa lệnh !p-join, bot sẽ gia nhập voice channel mà người gửi tin nhắn đang tham gia. Sau đó, người dùng có thể gửi tin nhắn văn bản cho bot thông qua lệnh !p-t, và bot sẽ chuyển tin nhắn này thành giọng nói bằng cách sử dụng thư viện gTTS (Google Text-to-Speech) để tạo file âm thanh, sau đó phát nó trong voice channel. Đặc biệt, bot hỗ trợ phát âm thanh bằng cả tiếng Việt và tiếng Anh, mang lại trải nghiệm tương tác phong phú cho người dùng. Các thư viện cần thiết cho việc này bao gồm discord.py để kết nối và tương tác với Discord, gTTS để chuyển văn bản thành âm thanh, và PyNaCl để bot có thể tham gia vào voice channel.

|  |
| --- |
| import discord  from discord.ext import commands  from gtts import gTTS  import os  import asyncio  from googletrans import Translator  # Dùng để dịch văn bản  # Đặt token bot Discord của bạn ở đây  TOKEN = 'MTI5NDYyNDQ0NzY4NDQxMTQyMw.GwSMUy.IRAr1xCmtCE6DsVUemaBFLsp-m\_dQeUdJh7Zf0'  # Thay thế bằng token của bạn  # Khởi tạo bot với quyền cần thiết  intents = discord.Intents.default()  intents.messages = True  # Để nhận tin nhắn từ kênh chat  intents.message\_content = True  # Đảm bảo quyền đọc nội dung tin nhắn được bật  intents.guilds = True  intents.voice\_states = True  # Cho phép bot sử dụng voice  bot = commands.Bot(*command\_prefix*='!p', *intents*=intents)  # Hàm dịch văn bản  def translate\_text(*text*, *source\_lang*='vi', *target\_lang*='en'):      translator = Translator()      translated = translator.translate(text, *src*=source\_lang, *dest*=target\_lang)      return translated.text  # Lệnh bot vào voice channel khi gõ "!p-join"  @bot.command(*name*='-join')  async def join(*ctx*):      if ctx.author.voice:          channel = ctx.author.voice.channel          if ctx.voice\_client is None:              try:                  await channel.connect()                  await ctx.send(f"Bot đã vào voice channel: {channel.name}")              except *Exception* as e:                  await ctx.send(f"Không thể kết nối vào voice channel: {e}")                  print(f"Lỗi kết nối vào voice channel: {e}")  # In lỗi vào console          else:              await ctx.send("Bot đã có mặt trong voice channel.")      else:          await ctx.send("Bạn cần phải ở trong voice channel để tôi vào.")  # Lệnh xử lý phát âm thanh khi có lệnh "!p-t"  @bot.command(*name*='t')  async def google(*ctx*, \*, *message*: *str*):      # Kiểm tra bot đã vào voice channel chưa      if ctx.voice\_client is None:          await ctx.send("Bot cần phải vào voice channel trước. Hãy dùng lệnh !p-join.")          return      # Dịch văn bản tiếng Việt sang tiếng Anh      translated\_text = translate\_text(message, *source\_lang*='vi', *target\_lang*='en')         # Chuyển văn bản thành giọng nói tiếng Việt      tts\_vi = gTTS(*text*=message, *lang*='vi')      tts\_en = gTTS(*text*=translated\_text, *lang*='en')      # Tạo file âm thanh tạm thời      tts\_vi.save("message\_vi.mp3")      tts\_en.save("message\_en.mp3")      # Lấy kênh voice của người dùng      voice\_client = ctx.voice\_client      # Phát giọng nói tiếng Việt      voice\_client.play(discord.FFmpegPCMAudio("message\_vi.mp3"), *after*=lambda *e*: print('Done playing Vietnamese', e))      while voice\_client.is\_playing():          await asyncio.sleep(1)      # Phát giọng nói tiếng Anh      voice\_client.play(discord.FFmpegPCMAudio("message\_en.mp3"), *after*=lambda *e*: print('Done playing English', e))      while voice\_client.is\_playing():          await asyncio.sleep(1)      # Xóa file âm thanh sau khi phát      os.remove("message\_vi.mp3")      os.remove("message\_en.mp3")  # Khi bot khởi động  @bot.event  async def on\_ready():      print(f'{bot.user} đã sẵn sàng!')      print(f"Bot đang hoạt động với {len(bot.guilds)} server.")  # Khởi chạy bot  bot.run(TOKEN) |

Kết quả thực nghiệm

Khởi động code



Gõ lệnh trên kênh chat và bot sẽ vào phòng trò chuyện, sau đó gõ lệnh !pt thì chat-bot sẽ nói

A screenshot of a computer

Description automatically generated

Thực hiện thành công chat bot sẽ nói và thông báo thành công bên terminal

A black screen with white text

Description automatically generated

## 3.4 LẬP TRÌNH GIAO DIỆN ĐỒ HỌA (GUI)

### 3.4.1 Giới thiệu bài toán

GUI (Graphical User Interface) là giao diện người dùng đồ họa, cho phép người dùng tương tác với phần mềm thông qua các hình ảnh, biểu tượng và các thành phần đồ họa khác, thay vì chỉ sử dụng dòng lệnh. Python hỗ trợ phát triển GUI qua nhiều thư viện khác nhau, cho phép bạn tạo ứng dụng đồ họa một cách dễ dàng và hiệu quả.

### 3.4.2 Các yêu cầu bài làm

Kết hợp với 2 bài tháp hà nội và

### 3.4.3 Các thư viện Python liên quan

speech\_recognition: Dùng để nhận diện giọng nói và chuyển đổi thành văn bản.

gtts: Chuyển đổi văn bản thành giọng nói (Text-to-Speech).

playsound: Phát tệp âm thanh (.mp3, .wav) trong Python.

tkinter: Thư viện GUI (giao diện người dùng đồ họa) để tạo cửa sổ, nút bấm, hộp thoại, v.v.

messagebox và simpledialog (từ tkinter): Tạo hộp thoại thông báo và yêu cầu nhập thông tin từ người dùng.

os: Quản lý hệ điều hành, như thao tác với tệp và thư mục.

### 3.4.4 Full code (mã lệnh lập trình

|  |
| --- |
| from tkinter import \*  from tkinter import messagebox  # Tạo một cửa sổ mới  frw27ctnp = Tk()  # Thêm tiêu đề cho cửa sổ  frw27ctnp.title('27 Cao Thị Ngọc Phụng \_ Lớp \_ HUCMUTE: ĐỒ ÁN HP: LẬP TRÌNH PYTHON')  # Đặt kích thước của cửa sổ  frw27ctnp.geometry('500x400')  lbl = Label(frw27ctnp, *text*="27\_CaoThiNgocPhung \n Đồ Án Học Phần: Lập trình Python \n SV Thực hiện: Cao Thị Ngọc Phụng, STT: 27 \n Trường: HCMUTE", *font*=("Arial Bold", 10))  lbl.grid(*column*=0, *row*=0)  # Hàm thoát chương trình  def Thoat():      traloi = messagebox.askquestion("Xác nhận", "Bạn có muốn thoát không (Y/N)?")      if traloi == "yes": frw27ctnp.destroy() # wn.quit()  # Nút thoát chương trình  btnThoat = Button(frw27ctnp, *text*="Thoát", *width*=10, *command*=Thoat)  btnThoat.place(*x*=100, *y*=120)  # Căn cứ vào kích thước form [wn.geometry("800x600")] để đặt vị trí Button "Thoát"  # Thiết lập nhãn Label thông tin sinh viên  lblText\_27ctnp = Label(frw27ctnp, *text*="Cao Thị Ngọc Phụng STT:27", *background*="yellow", *fg*="blue", *relief*=SUNKEN, *font*="Times 16", *borderwidth*=3, *width*=30, *height*=2)  # Đặt Label vừa thiết lập thông tin vào Form  lblText\_27ctnp.place(*x*=0, *y*=150)  # Thiết lập Entry để người dùng nhập vào  txtSource\_ctnp\_21110276 = Entry(frw27ctnp, *width*=30)  txtSource\_ctnp\_21110276.place(*x*=100, *y*=90)  # Lấy giá trị người dùng đã nhập vào textbox  a = txtSource\_ctnp\_21110276.get()  # Lấy giá trị người dùng đã nhập và cắt bỏ các khoảng trắng dư thừa  a = txtSource\_ctnp\_21110276.get().strip()  # Khởi chạy vòng lặp sự kiện của Tkinter  frw27ctnp.mainloop()  # Phải có dòng này để cửa sổ Tkinter hoạt động |

### 3.4.6 Kết quả thực nghiệm

A screenshot of a computer

Description automatically generated

### 3.4.7 Các bài toán tương tự

#### 3.4.7.1 Bài GUI 1

|  |
| --- |
| import speech\_recognition as sr  from gtts import gTTS  import playsound  import tkinter as tk  from tkinter import messagebox  import os  # Thư viện OS để tạo thư mục và lưu file  import time  OUT\_FILE = "D:/PYPR (Python Programming)/G127CTNPNBAPlayer/B6Ex3\_GUI/ngocphung\_27/ngocphung\_27\_GUI\_1.mp3"  # Thay đổi đường dẫn file âm thanh đầu ra  odir = 'D:/PYPR (Python Programming)/G127CTNPNBAPlayer/B6Ex3\_GUI/ngocphung\_27'  # Thay đổi thư mục lưu file âm thanh  count = 0  os.makedirs(odir, *exist\_ok*=True)  # Tạo thư mục nếu chưa tồn tại  # Hàm thoát chương trình  def Thoat():      traloi\_ctnp = messagebox.askquestion("Xác nhận", "Bạn có muốn thoát không (Y/N)?")      if traloi\_ctnp == "yes":  # Nếu người dùng chọn "yes", đóng cửa sổ          wn.destroy()  # Hàm nhận lệnh giọng nói và chuyển thành văn bản  def Lenh():      r\_21110276ctnp = sr.Recognizer()  # Khởi tạo đối tượng nhận diện giọng nói      with sr.Microphone() as source:  # Sử dụng microphone làm nguồn âm thanh          messagebox.showinfo("Nhắc nhở", "Hiệu chỉnh MIC trước khi ra lệnh bằng lời nói!")  # Nhắc người dùng hiệu chỉnh mic          r\_21110276ctnp.adjust\_for\_ambient\_noise(source, *duration*=1)  # Hiệu chỉnh theo tiếng ồn môi trường          messagebox.showinfo("Cảnh báo", "Ra lệnh bằng tiếng Việt trong 3 giây, bấm OK để bắt đầu.")  # Nhắc người dùng ra lệnh          audio\_data = r\_21110276ctnp.record(source, *duration*=3)  # Ghi lại âm thanh trong 3 giây          try:              tt\_phung = r\_21110276ctnp.recognize\_google(audio\_data, *language*="vi")  # Nhận dạng giọng nói thành văn bản          except sr.UnknownValueError:  # Nếu không nhận diện được giọng nói              tt\_phung = "Bạn nói gì nghe không rõ!"  # Thông báo lỗi          except sr.RequestError:  # Nếu không thể kết nối với dịch vụ nhận dạng              tt\_phung = "Không thể kết nối với dịch vụ nhận diện."          messagebox.showinfo("Bạn đã nói:", tt\_phung)  # Hiển thị văn bản nhận diện được          ctnp = gTTS(*text*=tt\_phung, *lang*='vi')  # Chuyển văn bản thành âm thanh          ctnp.save(OUT\_FILE)  # Lưu âm thanh vào file tại thư mục mới  # Hàm phát âm thanh đã lưu  def Xuat():      try:          playsound.playsound(OUT\_FILE)  # Phát âm thanh từ file      except *Exception* as e:  # Nếu có lỗi xảy ra trong quá trình phát âm thanh          messagebox.showerror("Lỗi", f"Không thể phát âm thanh: {e}")  # Hiển thị thông báo lỗi  # Tạo cửa sổ chính của ứng dụng  wn = tk.Tk()  wn.title("Cao Thị Ngọc Phụng STT: 27, LỚP\_HCMUTE, ĐỒ ÁN HỌC PHẦN: LẬP TRÌNH PYTHON, T9.2024")  wn.geometry("800x600")  # Đặt kích thước cửa sổ  wn.resizable(False, False)  # Không cho phép thay đổi kích thước cửa sổ  # Tạo nhãn (Label) hiển thị thông tin về đồ án  lblText = tk.Label(      wn,  *text*="Phân tích thống kê về cầu thủ bóng rổ trong mùa giải NBA thường niên 2023-2024",  *background*="green", *fg*="blue", *relief*=tk.SUNKEN,  *font*="Times 16", *borderwidth*=3, *width*=60, *height*=3  )  lblText.place(*x*=10, *y*=10)  # Tạo nút "Thoát"  btnThoat = tk.Button(wn, *text*="Thoát", *width*=10, *command*=Thoat)  btnThoat.place(*x*=100, *y*=200)  # Tạo nút "Lệnh = nói"  btnLenh = tk.Button(wn, *text*="Lệnh = nói", *width*=10, *command*=Lenh)  btnLenh.place(*x*=400, *y*=200)  # Tạo nút "ĐỌC"  btnDoc = tk.Button(wn, *text*="ĐỌC", *width*=10, *command*=Xuat)  btnDoc.place(*x*=500, *y*=200)  # Chạy vòng lặp sự kiện của tkinter  wn.mainloop() |

Kết quả thực nghiệm

A screenshot of a computer

Description automatically generated

Kiểm tra thấy file trong thư mục

A screenshot of a computer

Description automatically generated

Bấm đọc thì thì phát âm thanh giống nội dung trên

A screenshot of a computer

Description automatically generated

#### 3.4.7.2 Bài GUI 2

|  |
| --- |
| import tkinter as tk  from tkinter import messagebox, simpledialog  import speech\_recognition as sr  from gtts import gTTS  import playsound  import os  # Thư mục lưu tạm file âm thanh  OUT\_DIR = "27CaoThiNgocPhung"  os.makedirs(OUT\_DIR, *exist\_ok*=True)  # 1. THÁP HÀ NỘI  def thapHaNoi\_27ctnp(*n*, *toaMot*, *toaHai*, *toaBa*):      if n == 1:          print(f"Chuyển từ {toaMot} sang {toaBa}")      else:          thapHaNoi\_27ctnp(n - 1, toaMot, toaBa, toaHai)          print(f"Chuyển từ {toaMot} sang {toaBa}")          thapHaNoi\_27ctnp(n - 1, toaHai, toaMot, toaBa)  def run\_thapHaNoi\_ctnp():      n = *int*(simpledialog.askstring("Nhập số đĩa", "Nhập số đĩa:"))      print(f"Bắt đầu chuyển {n} đĩa:")      thapHaNoi\_27ctnp(n, "1", "2", "3")  # 2. VOICE ASSISTANT  def voice\_assistant\_21110276\_ctnp():      recognizer = sr.Recognizer()      with sr.Microphone() as source:          duration = *int*(simpledialog.askstring("Chọn thời gian", "Chọn thời gian hiệu chỉnh (giây):"))          messagebox.showinfo("Nhắc nhở", f"Hiệu chỉnh mic trong {duration} giây.")          recognizer.adjust\_for\_ambient\_noise(source, *duration*=duration)          messagebox.showinfo("Thông báo", "Bắt đầu nói trong 5 giây.")          audio\_data = recognizer.record(source, *duration*=5)          try:              text = recognizer.recognize\_google(audio\_data, *language*="vi")          except sr.UnknownValueError:              text = "Quý vị nói gì nghe không rõ!"          except sr.RequestError:              text = "Không thể kết nối với dịch vụ nhận diện."          messagebox.showinfo("Kết quả nhận diện", f"Quý vị đã nói: {text}")          text\_to\_speech\_ctnp(text)  def text\_to\_speech\_ctnp(*text*):      audio\_file = os.path.join(OUT\_DIR, "output.mp3")      tts = gTTS(*text*=text, *lang*='vi')      tts.save(audio\_file)      playsound.playsound(audio\_file)  # 3. EDA: PRE-PROCESSING  def eda\_preprocessing\_ctnp():      col\_count = *int*(simpledialog.askstring("Số cột", "Nhập số cột NULL cần loại bỏ:"))      messagebox.showinfo("Thông báo", f"Sẽ loại bỏ {col\_count} cột NULL nhiều nhất (mô phỏng).")      print(f"Mô phỏng: Loại bỏ {col\_count} cột NULL nhiều nhất.")  # GIAO DIỆN GUI CHÍNH  def main():      wn = tk.Tk()      wn.title("MainForm - Đồ Án Python")      wn.geometry("500x400")      # Nhãn tiêu đề      lbl\_title = tk.Label(          wn, *text*="Lập Trình Python - Đồ Án Học Phần",  *font*=("Arial", 16), *bg*="yellow", *fg*="blue", *height*=2      )      lbl\_title.pack(*fill*=tk.X)      # Nút Tháp Hà Nội      btn\_thapHaNoi = tk.Button(wn, *text*="Tháp Hà Nội", *width*=20, *command*=run\_thapHaNoi\_ctnp)      btn\_thapHaNoi.pack(*pady*=10)      # Nút Voice Assistant      btn\_voice = tk.Button(wn, *text*="Voice Assistant", *width*=20, *command*=voice\_assistant\_21110276\_ctnp)      btn\_voice.pack(*pady*=10)      # Nút EDA Pre-processing      btn\_eda = tk.Button(wn, *text*="EDA Pre-processing", *width*=20, *command*=eda\_preprocessing\_ctnp)      btn\_eda.pack(*pady*=10)      # Nút Thoát      btn\_exit = tk.Button(wn, *text*="Thoát", *width*=20, *command*=wn.quit)      btn\_exit.pack(*pady*=10)      wn.mainloop()  if \_\_name\_\_ == "\_\_main\_\_":      main() |

Kết quả thực nghiệm

A screenshot of a computer

Description automatically generated

**A screenshot of a computer

Description automatically generated**

File được lưu dưới tên ngocphung\_27\_GUI\_2.mp3

**A screenshot of a computer

Description automatically generated**

# CHƯƠNG 4 : PHÂN TÍCH THỐNG KÊ VỀ CẦU THỦ NBA CHO MÙA GIẢI THƯỜNG NÊN 2023-2024

## 4.1 Giới thiệu chủ đề EDA G127CTNPNBAPlayer

Vận dụng kỹ thuật lập trình EDA kết hợp với GUI và Voice Assistant cho phép dùng tập dữ liêu cung cấp thông tin chi tiết về tên cầu thủ, vị trí thi đấu (như trung phong, tiền phong hay hậu vệ), độ tuổi và đội bóng mà cầu thủ thi đấu. Bên cạnh đó, bộ dữ liệu còn ghi lại các chỉ số thống kê quan trọng như số trận đấu, số trận đá chính, thời gian thi đấu trung bình mỗi trận, và số điểm ghi được. Các chỉ số hiệu quả như tỷ lệ ghi điểm tổng thể (FG%), tỷ lệ ghi 3 điểm (3P%), tỷ lệ ghi 2 điểm (2P%), tỷ lệ ném phạt (FT%), cũng như các chỉ số phòng ngự như số lần cướp bóng (STL) và số lần block (BLK) đều được ghi lại chi tiết. Dữ liệu này có thể được sử dụng để phân tích và so sánh hiệu suất giữa các cầu thủ, giúp đánh giá mức độ đóng góp của mỗi cầu thủ trong đội bóng và hỗ trợ các nghiên cứu về chiến thuật thi đấu trong bóng rổ.

## 4.2 Giới thiệu bài toán (Yêu cầu đặt ra)

Bài toán phân tích dữ liệu này tập trung vào việc khám phá và đánh giá hiệu suất của các cầu thủ bóng rổ trong mùa giải NBA, thông qua việc phân tích các chỉ số thống kê trong quá khứ. Dữ liệu bao gồm các thông tin như tên cầu thủ, vị trí thi đấu, độ tuổi, đội bóng, và các chỉ số hiệu suất quan trọng như số trận đấu, số điểm ghi được, tỷ lệ ném bóng, cũng như các chỉ số phòng ngự. Thông qua việc thăm dò các mối quan hệ giữa các chỉ số này, chúng ta có thể dự đoán và phân tích khả năng đóng góp của từng cầu thủ vào thành tích chung của đội bóng trong các trận đấu sắp tới. Việc phân tích dữ liệu quá khứ không chỉ giúp đánh giá mức độ hiệu quả của mỗi cầu thủ mà còn hỗ trợ đưa ra những dự đoán về xu hướng thi đấu và chiến thuật trong tương lai.

## 4.3 Thông tin về file dữ liệu thực nghiệm EDA

* File dữ liệu gốc (Dataset): NBA\_2024\_per\_game(03-01-2024).csv <https://www.kaggle.com/datasets/bryanchungweather/nba-player-stats-dataset-for-the-2023-2024/data> Tác giả (Author) : <https://www.linkedin.com/in/bryan-chung-2a562b22a/>
* Thông tin về tập dữ liệu (DataSet)

+ Dạng file Excel .csv

+ Số records: 539 (quan sát tại giải đấu NBA mùa giải 2023-2024)

+ Thời gian quan sát: 24/10/2023 đến 14/04/2024

* Gồm 30 cột dữ liệu:
* Rk (Rank): Xếp hạng của cầu thủ trong bảng thống kê.
* Player: Tên cầu thủ.
* Pos (Position): Vị trí thi đấu của cầu thủ (ví dụ: PG - Point Guard, SG - Shooting Guard, SF - Small Forward, PF - Power Forward, C - Center).
* Age: Tuổi của cầu thủ trong mùa giải.
* Tm (Team): Đội bóng mà cầu thủ đang thi đấu.
* G (Games played): Số trận đấu mà cầu thủ tham gia.
* GS (Games started): Số trận mà cầu thủ bắt đầu trong đội hình chính.
* MP (Minutes played per game): Số phút trung bình mà cầu thủ chơi trong mỗi trận đấu.
* FG (Field goals per game): Số lần cầu thủ ghi điểm từ các cú ném bóng vào rổ mỗi trận.
* FGA (Field goal attempts per game): Số lần cầu thủ cố gắng ném bóng vào rổ mỗi trận.
* FG% (Field goal percentage): Tỷ lệ thành công của các cú ném bóng vào rổ.
* 3P (3-point field goals per game): Số lần cầu thủ ghi điểm từ các cú ném 3 điểm mỗi trận.
* 3PA (3-point field goal attempts per game): Số lần cầu thủ cố gắng ném 3 điểm mỗi trận.
* 3P% (3-point field goal percentage): Tỷ lệ thành công của các cú ném 3 điểm.
* 2P (2-point field goals per game): Số lần cầu thủ ghi điểm từ các cú ném 2 điểm mỗi trận.
* 2PA (2-point field goal attempts per game): Số lần cầu thủ cố gắng ném 2 điểm mỗi trận.
* 2P% (2-point field goal percentage): Tỷ lệ thành công của các cú ném 2 điểm.
* eFG% (Effective field goal percentage): Tỷ lệ thành công của các cú ném bóng, tính cả các cú ném 3 điểm.
* FT (Free throws per game): Số lần cầu thủ ghi điểm từ các cú ném phạt mỗi trận.
* FTA (Free throw attempts per game): Số lần cầu thủ cố gắng ném phạt mỗi trận.
* FT% (Free throw percentage): Tỷ lệ thành công của các cú ném phạt.
* ORB (Offensive rebounds per game): Số lần cầu thủ thu hồi bóng trong tình huống tấn công mỗi trận.
* DRB (Defensive rebounds per game): Số lần cầu thủ thu hồi bóng trong tình huống phòng ngự mỗi trận.
* TRB (Total rebounds per game): Số lần cầu thủ thu hồi bóng tổng cộng mỗi trận (bao gồm cả offensive và defensive rebounds).
* AST (Assists per game): Số lần cầu thủ chuyền bóng thành công cho đồng đội để ghi điểm mỗi trận.
* STL (Steals per game): Số lần cầu thủ cướp bóng từ đối phương mỗi trận.
* BLK (Blocks per game): Số lần cầu thủ chặn cú ném bóng của đối phương mỗi trận.
* TOV (Turnovers per game): Số lần cầu thủ mất bóng hoặc bị đối phương cướp bóng mỗi trận.
* PF (Personal fouls per game): Số lần cầu thủ phạm lỗi cá nhân mỗi trận.
* PTS (Points per game): Số điểm cầu thủ ghi được trung bình mỗi trận.

## 4.4 Phân tích yều cầu của chủ đề

Chủ đề lập trình Python này là nền tảng cho các ứng dụng liên quan đến Data Warehouse, Big Data, Phân tích Dữ liệu, Khoa học Dữ liệu, v.v.

Đây là bước xử lý cơ bản cho các lĩnh vực chuyên môn trên, chủ yếu tập trung vào giai đoạn tiền xử lý dữ liệu đầu vào (Input Preprocessing), bao gồm các công việc như xử lý giá trị NULL, lọc dữ liệu, tích hợp dữ liệu (Integration), và các thao tác khác.

## 4.5 Tiến hành thực hiện

Tiến trình phân tích dữ liệu thăm dò (EDA) thường được thực hiện qua 3 giai đoạn chính:

**GIAI ĐOẠN 1:** Nạp dữ liệu gốc (Primary Input Data Load): Bước 1 đến Bước 2  
**GIAI ĐOẠN 2:** Tiền xử lý dữ liệu (Pre-Processing): Bước 3 đến Bước 7  
**GIAI ĐOẠN 3:** Phân tích dữ liệu thăm dò (Exploratory Data Analysis - EDA): Bước 8 trở đi

Một số bước tiền xử lý dữ liệu đầu vào (input) trong hệ thống EDA bao gồm:  
[1] Xử lý giá trị NULL: ……………  
[2] Xử lý dữ liệu ngoại lệ (outliers): Khi dữ liệu quá khác biệt hoặc bị cô lập, gọi là "ngoại lệ" (isolated), là ……………

## 4.6 Các nền tảng kỹ thuật liên quan

### 4.6.1 Hệ thống thư viện sử dụng

NumPy (#Numeric Python): Thư viện hỗ trợ đại số tuyến tính.

Pandas (#Python Data Analysis Library): Thư viện xử lý và phân tích dữ liệu.

SciPy => stats: Thư viện cung cấp các công cụ thống kê (statistics) và là một phần trong bộ công cụ khoa học của Python (science libraries).

sklearn => preprocessing: Thư viện tiền xử lý dữ liệu (ví dụ: xử lý ngoại lệ, Isolated).

sklearn.feature\_selection => SelectKBest, chi2: Thư viện phân tích dữ liệu thăm dò, cung cấp các hàm chọn đặc trưng tốt nhất.

### 4.6.2 Codes nạp thư viện

|  |
| --- |
| import numpy as np  # Đại số tuyến tính (Numeric Python)  import pandas as pd  # Xử lý dữ liệu (Python Data Analysis)  from scipy import stats  # Thống kê (statistics), công cụ khoa học Python  from sklearn import preprocessing  # Tiền xử lý dữ liệu (Pre-processing)  from sklearn.feature\_selection import SelectKBest, chi2  # Phân tích dữ liệu thăm dò |

### 4.6.3 Tập dữ liệu thực nghiệm

#### 4.6.3.1 ****Bước 2: Tải tập dữ liệu****

|  |
| --- |
| df = pd.read\_csv('./NBAPLayer.csv') # Load the data set (Nạp tập dữ liệu)  print('Độ lớn của bảng [frame] dữ liệu thời tiết:',df.shape) # Display the shape of the data set (Vẽ biểu đồ tập dữ liệu)  print(df[0:5]) # Display data (Hiển thị dữ liệu dạng mảng 5 dòng đầu) |

**Kết quả:** Độ lớn của bảng [frame] dữ liệu cầu thủ: (538, 29)

A black and white screen with white text

Description automatically generated

### 4.6.4 Tiền xử lý (Pre-Processing)

#### 4.6.4.1 Bước 3

Bước 3: Xử lý CỘT dữ liệu NULL quá nhiều OR không có giá trị phân tích

|  |
| --- |
| print(df.count().sort\_values()) |

Kết quả

|  |
| --- |
| FT% 476 🡪 có 63 dòng NULL  3P% 489 🡪 có 43 dòng NULL  2P% 521 🡪 có 17 dòng NULL  FG% 530 🡪 có 8 dòng NULL  eFG% 530 🡪 có 8 dòng NULL  Pos 538  G 538  Player 538  FG 538  MP 538  GS 538  FGA 538  3PA 538  3P 538  Age 538  Tm 538  2PA 538  2P 538  FT 538  FTA 538  ORB 538  DRB 538  TRB 538  AST 538  STL 538  BLK 538  TOV 538  PF 538  PTS 538 |

A screenshot of a computer

Description automatically generated

**Nhận xét:**

Kết quả phân tích dữ liệu cho thấy có một số cột có tỷ lệ giá trị NULL cao, ảnh hưởng đến chất lượng và tính chính xác của quá trình phân tích dữ liệu (EDA). Cụ thể:

* Các cột FT%, 3P%, 2P%, FG%, eFG% có tỷ lệ NULL lần lượt là 63, 43, 17, 8 dòng NULL trong tổng số 538 dòng, chiếm tỷ lệ khá lớn trong dữ liệu. Những cột này có thể ảnh hưởng đến tính chính xác trong quá trình phân tích và dự đoán, nên cần được xem xét và có thể loại bỏ nếu không đóng góp nhiều vào mô hình phân tích.
* Các cột Pos, G, Player, FG, MP, GS, FGA, 3PA, 3P, Age, Tm, 2PA, 2P, FT, FTA, ORB, DRB, TRB, AST, STL, BLK, TOV, PF, PTS không có giá trị NULL và có đầy đủ dữ liệu cho tất cả các dòng, vì vậy chúng có thể được giữ lại và sử dụng trong các phân tích sau.

**Tiền xử lý (Pre-processing):**

* Với các cột có tỷ lệ NULL cao như FT%, 3P%, 2P%, FG%, eFG%, ta cần cân nhắc loại bỏ để tránh ảnh hưởng đến chất lượng của phân tích. Việc giữ lại các cột này có thể làm tăng độ phức tạp của mô hình mà không mang lại giá trị thực sự trong phân tích. Đây là các cột không đáng tin cậy vì chúng có quá nhiều giá trị thiếu.
* Các cột có đầy đủ dữ liệu (như Pos, G, Player, FG, MP, GS, FGA, ...) nên được giữ lại để đảm bảo tính toàn vẹn của dữ liệu.

**Kết luận:**

* Loại bỏ các cột có tỷ lệ NULL cao: FT%, 3P%, 2P%, FG%, eFG% vì chúng không cung cấp đủ thông tin cho phân tích.
* Giữ lại các cột có dữ liệu đầy đủ và có ý nghĩa như Pos, G, Player, FG, MP, GS, FGA, 3PA, 3P, Age, Tm, ....

|  |
| --- |
| df = df.drop(*columns*=['FT%', '3P%', '2P%', 'FG%', 'eFG%'], *axis*=1)  print(df.shape) # kiểm tra lại số lượng cột & dòng của df sau khi XL NULL cột |

Kết quả cho thấy đã bỏ đi 5 cột

(538, 29) 🡪 (538, 24)

|  |
| --- |
| (538, 24) |

A screenshot of a phone

Description automatically generated

#### 4.6.1.2 Bước 4

Bước 4: Xử lý DÒNG dữ liệu NULL

|  |
| --- |
| df = df.dropna(*how*='any')  print(df.shape) # kiểm tra lại số lượng cột & dòng của df sau khi XL NULL các dòng DL |

Kết quả không có dòng nào NULL

|  |
| --- |
| (538, 24) |

🡪 (538, 24) = drop: 538 – 538 = 538 dòng bị Drop

#### 4.6.1.3 Bước 5

Bước 5: Xử lý loại bỏ các giá trị ngoại lệ (cá biệt): isolated

|  |
| --- |
| z = np.abs(stats.zscore(df.\_get\_numeric\_data())) # Dò tìm và lấy các giá trị cá biệt trong tập dữ liệu gốc thông qua điểm z (z\_score)  print('MA TRAN Z-SCORE\n')  print(z) # in ra tập (ma trận) các giá trị z-score từ tập dữ liệu gốc |

**df.\_get\_numeric\_data()**:

* Lệnh này lấy ra tất cả các cột có kiểu dữ liệu là số (numeric) trong DataFrame df. Các cột có kiểu dữ liệu chuỗi (string) hoặc những cột không có giá trị số sẽ bị loại bỏ.
* Điều này giúp bạn chỉ làm việc với các cột có dữ liệu số, như điểm số (PTS), số phút thi đấu (MP), số lần ghi bàn (FG), v.v., trong trường hợp của bạn là dataset về bóng rổ.

**stats.zscore()**:

* Đây là một hàm từ thư viện scipy.stats, dùng để tính Z-Score của các giá trị trong một cột dữ liệu.
* Z-Score cho biết một giá trị (phần tử) trong dữ liệu cách bao xa so với trung bình (mean) của cột đó, tính bằng đơn vị độ lệch chuẩn (standard deviation). Công thức tính Z-Score là: Z=X−μσZ = \frac{X - \mu}{\sigma}Z=σX−

|  |
| --- |
| Age G GS MP FG FGA 3P \  0 0.441876 0.570821 0.774672 0.125027 0.047157 0.016714 0.479979  1 0.441876 0.476849 0.774672 0.096408 0.008423 0.021227 0.479979  2 0.441876 1.778489 0.774672 0.907283 1.209167 0.889364 1.020886  3 0.012239 0.288904 1.158337 1.487183 1.889527 1.709616 1.020886  4 0.668933 1.322601 0.065767 0.247022 0.202092 0.187450 0.277290  .. ... ... ... ... ... ... ...  533 2.055757 1.214655 0.774672 1.346109 0.860564 1.003188 1.020886  534 0.214818 0.946711 1.746644 1.725676 2.199396 2.563296 2.873641  535 0.214818 0.006987 0.270409 0.649711 0.511961 0.604804 0.804523  536 1.147527 0.288904 0.774672 0.850044 0.976765 0.946276 1.020886  537 0.012239 1.134656 1.914732 0.800324 0.843717 0.286817 1.020886  3PA 2P 2PA ... FTA ORB DRB \  0 0.365136 0.156983 0.203071 ... 0.052966 1.440654 0.481856  1 0.321866 0.205830 0.230040 ... 0.001312 1.440654 0.481856  2 1.143984 1.064195 0.552066 ... 0.921408 0.195559 0.259734  3 1.014176 2.794725 3.088771 ... 3.312247 1.565163 3.077422  4 0.283905 0.380335 0.444189 ... 0.704298 0.053460 0.418646  .. ... ... ... ... ... ... ...  533 1.143984 0.624571 0.713881 ... 0.704298 0.551498 1.001324  534 2.880068 1.475854 1.848190 ... 3.583635 0.426989 0.005120  535 0.841099 0.282641 0.336312 ... 0.704298 0.818106 0.217003  ...  536 0.131223 0.445270 0.864695 0.640100 0.804788 0.295307 0.933199  537 2.512742 0.393216 0.864695 2.019156 0.242172 1.382952 0.514546 |

A screenshot of a computer

Description automatically generated

Z-Score được sử dụng để xác định các giá trị cá biệt trong tập dữ liệu, bằng cách đo lường độ lệch chuẩn của mỗi giá trị so với trung bình. Cụ thể, một giá trị có Z-Score lớn hơn hoặc nhỏ hơn 3 thường được coi là **outlier** (giá trị cá biệt). Tuy nhiên, ngưỡng Z-Score = 3 không phải là cố định và có thể được điều chỉnh tùy thuộc vào phân phối của dữ liệu.Trong thực tế, bạn có thể thay đổi giá trị ngưỡng Z-Score (ví dụ: từ 2.5 đến 3.5) dựa trên phân phối của dữ liệu. Để xác định ngưỡng phù hợp, bạn có thể sử dụng **histogram** hoặc các biểu đồ phân phối khác để quan sát sự phân bố của Z-Score. Dựa trên biểu đồ này, nếu bạn nhận thấy phần lớn các giá trị nằm trong phạm vi gần trung bình và chỉ có một số ít giá trị xa trung bình, bạn có thể điều chỉnh ngưỡng để chỉ lọc ra các outliers thực sự. Tóm lại, mặc dù ngưỡng Z-Score = 3 là một chuẩn phổ biến, nhưng bạn hoàn toàn có thể điều chỉnh giá trị này dựa trên đặc điểm của dữ liệu và mục tiêu phân tích của mình.

**[1] NẾU ngưỡng là 3 (z < 3)**

|  |
| --- |
| df= df[(z < 3).all(*axis*=1)] # kiểm tra và chỉ giữ lại trong df các giá trị số liệu tưng ứng với z-score < 3  # {loại các giá trị >= 3} vì các giá trị z-score >=3 tướng ứng với số liệu quá khác biệt so với các số liệu còn lại (“cá biệt” = “ngoại lệ” = isolated}  print(df.shape) # xác định số dòng & cột dữ liệu sau khu xử lý các giá trị cá biệt |

**Thì kết quả**

|  |
| --- |
| (493, 24) |

A screenshot of a computer

Description automatically generated

[538, 24] 🡪 (493, 24)

45 rows =loại các dòng có giá trị ngoại lệ = còn lại => 493 dòng / 21 cột giá trị số /

=> Số dòng có giá trị ngoại lệ: 538 – 493 = 45 records (9.13 %)

**[2] NẾU ngưỡng là 54 (z < 3)**

Z-Score max là 54

|  |
| --- |
| df= df[(z < 54).all(*axis*=1)] #kiểm tra và chỉ giữ lại trong df các giá trị số liệu tưng ứng với z-score < 54  # {loại các giá trị >= 3} vì các giá trị z-score >=54 tướng ứng với số liệu quá khác biệt so với các số liệu còn lại (“cá biệt” = “ngoại lệ” = isolated}  print(df.shape) # xác định số dòng & cột dữ liệu sau khu xử lý các giá trị cá biệt |

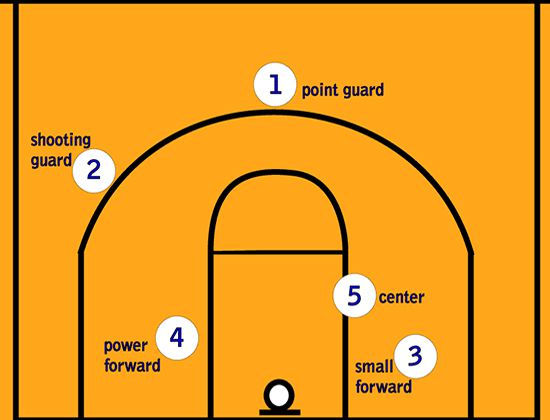
Thì kết quả là không xóa bất kì dòng nào

A screen shot of a computer

Description automatically generated

#### 4.6.1.4 Bước 6

Bước 6: RR THEO Mã hóa trược tiếp



Dựa vào các thuật ngữ vị trí trong môn bóng rổ và dựa vào tập tài liệu thì mã hóa như sau ở cột Pos

PG 🡪 1

SG 🡪 1

SF 🡪 0

PF 🡪 0

C 🡪 0

Đối với vị trí vòng ngoài 3 điểm ta sẽ cho là 1 và vòng trong 2 điểm là 0

Xóa những dòng không phải 5 dữ liệu trên

|  |
| --- |
| # Mã hóa các giá trị trong cột 'Pos' bằng cách sử dụng 'replace'  df['Pos'].replace({'PG': 1, 'SG': 1, 'SF': 0, 'PF': 0, 'C': 0}, *inplace*=True)  # Xóa các dòng có giá trị 'Pos' không nằm trong danh sách đã định  df = df[df['Pos'].isin([1, 2, 3, 4, 5])]  # Kiểm tra kết quả  print(df['Pos'])  # In ra 10 dòng đầu tiên của DataFrame để xem kết quả mã hóa |

Kết quả

A screenshot of a cell phone

Description automatically generated

#### 4.6.1.5 Bước 7

Bước 7: RR hóa theo khoảng / đoạn = Chuẩn hóa (Rời rạc hóa) tập dữ liệu Input dùng ..MaxMin

|  |
| --- |
| df\_num = df.drop(*columns*=['Player', 'Tm'], *axis*=1)  rr = preprocessing.MinMaxScaler()  # xác định thang đo sẽ dùng để RR hóa theo khoảng đều  rr.fit(df\_num)  # Áp dụng thang đo vào data frame [df] của đề tài (đã tiền xử lý đến Bước 5)  df\_num = pd.DataFrame(rr.transform(df\_num), *index*=df.index, *columns*=df\_num.columns)  # Chuyển đổi dữ liệu  df\_num.iloc[4:10]  # Lấy 5 dòng dữ liệu từ chỉ số 4 đến 10  print(df\_num)  # In ra DataFrame đã được chuẩn hóa |

Kết quả là

|  |
| --- |
| Pos Age G GS MP FG FGA 3P \  1 1.00 0.25 0.705882 0.000000 0.453297 0.262712 0.288136 0.106383  2 0.75 0.25 0.000000 0.000000 0.219780 0.000000 0.084746 0.000000  3 1.00 0.35 0.647059 0.657143 0.909341 0.677966 0.665254 0.000000  4 0.25 0.20 0.970588 0.285714 0.552198 0.220339 0.241525 0.255319  5 0.75 0.20 0.705882 0.142857 0.631868 0.355932 0.402542 0.340426  .. ... ... ... ... ... ... ... ...  533 0.75 0.80 0.176471 0.000000 0.093407 0.076271 0.059322 0.000000  534 0.00 0.30 0.852941 0.857143 0.978022 0.745763 0.855932 0.765957  535 1.00 0.30 0.558824 0.171429 0.293956 0.152542 0.148305 0.042553  536 1.00 0.60 0.647059 0.000000 0.236264 0.050847 0.072034 0.000000  537 1.00 0.35 0.911765 0.914286 0.711538 0.449153 0.347458 0.000000  3PA 2P ... FTA ORB DRB TRB \  1 0.163793 0.234234 ... 0.146552 0.377358 0.369565 0.439024  2 0.000000 0.000000 ... 0.000000 0.188679 0.217391 0.243902  3 0.025862 0.711712 ... 0.672414 0.396226 0.902174 0.845528  4 0.284483 0.126126 ... 0.034483 0.150943 0.184783 0.203252  5 0.413793 0.234234 ... 0.137931 0.245283 0.434783 0.439024  .. ... ... ... ... ... ... ...  533 0.000000 0.081081 ... 0.034483 0.075472 0.065217 0.081301  534 0.801724 0.468468 ... 0.715517 0.094340 0.271739 0.243902  535 0.060345 0.144144 ... 0.034483 0.283019 0.315217 0.357724  536 0.000000 0.054054 ... 0.094828 0.226415 0.195652 0.243902  537 0.000000 0.477477 ... 0.232759 0.603774 0.706522 0.788618  ...  536 0.086614 0.076923 0.032258 0.06 0.216667 0.051429   * 1. .094488 0.076923 0.419355 0.24 0.466667 0.348571 |

A screenshot of a computer

Description automatically generated

### 4.6.5 Phân tích dữ liệu thăm dò: EDA

#### 4.6.5.1 Bước 8

|  |
| --- |
| X = df\_num.loc[:, df\_num.columns != 'Pos']  # Tất cả các cột trừ cột 'Pos'  y = df\_num[['Pos']]  # Dữ liệu đầu ra là cột 'Pos'  # Loại bỏ các dòng có giá trị thiếu  selector = SelectKBest(chi2, *k*=3) # sd các hàm ... trong thư viện sklearn = Mô hình xác định các Thuộc tính quan trọng quyết định việc dự đoàn DL output = trích lọc Đặc trưng = Feature Extraction: k = 1...12 (đ/v bài này)  selector.fit(X, y) # Áp dụng mô hình trên vào ....  X\_new = selector.transform(X) # Chuyên DL Input teho mô hình  print(X\_new)  print(y) |

Kết quả như sau

A screenshot of a computer program

Description automatically generated

|  |
| --- |
| print('k cot quan trong nhat quyet dinh cho Vector Output')  print(X.columns[selector.get\_support(*indices*=True)]) # in ds các tt đặc trưng |

Kết quả như sau

A screen shot of a computer screen

Description automatically generated

#### 4.6.5.2 Bước 9

Qua bước 8 trên cho thấy 3 cột quan trọng là TOV, PF, PTS

|  |
| --- |
| df\_new = df\_num[['TOV','PF','PTS']]  # TOV (Turnovers per game): Số lần cầu thủ mất bóng hoặc bị đối phương cướp bóng mỗi trận.  # PF (Personal fouls per game): Số lần cầu thủ phạm lỗi cá nhân mỗi trận.  # PTS (Points per game): Số điểm cầu thủ ghi được trung bình mỗi trận.  X = df\_new[['PF']]  y = df\_new[['PTS']]  print(X)  print(y) |

Kết quả như sau

A screenshot of a computer screen

Description automatically generated

### 4.7 Full codes

|  |
| --- |
| ########################################################  # GIAI ĐOẠN 1: NẠP DỮ LIỆU GỐC (PRIMARY INPUT DATA LOAD)  ########################################################  #######################################  # Bước 1: Nạp các thư viện cần thiết  #######################################  import numpy as np #Numeric Python: Thư viện về Đại số tuyến tính tính  import pandas as pd #Python Analytic on Data System: For data processing (Thư viện xử lý dữ liệu)  from scipy import stats # thư viện cung cấp các công cụ thống kê [statistics] sub-lib của science python [các công cụ khoa học]  from sklearn import preprocessing # Thư viện tiền xử lý DL (XL ngoại lệ: Isolated) = thư viện chính of ML (trong môn PyPro chỉ làm Pre-Processing)  from sklearn.feature\_selection import SelectKBest, chi2 # Nạp hàm Thư viện hỗ trợ Mô hình phân tích dữ liệu thăm dò  ###############################################################  # Bước 2: Tải tập dữ liệu: Load the data set (Nạp tập dữ liệu)  # ./NBA\_2024\_per\_game(03-01-2024).csv  ##############################################################  df = pd.read\_csv('D:/PYPR (Python Programming)/G127CTNPNBAPlayer/G127CTNPNBAPLayer10EDA/NBA\_2024\_per\_game(03-01-2024).csv')  print('Độ lớn của bảng [frame] dữ liệu cầu thủ:',df.shape) # Display the shape of the data set (Vẽ biểu đồ tập dữ liệu)  print(df[0:5]) # Display data (Hiển thị dữ liệu dạng mảng 5 dòng đầu)  ###############################################  # # GIAI ĐOẠN 2: TIỀN XỬ LÝ (PRE-PROCESSING)  ###############################################  ############################################################################  # # Bước 3: Xử lý CỘT dữ liệu NULL quá nhiều OR không có giá trị phân tích  ############################################################################  # Checking for null values (Kiểm tra giá trị null = đếm số dòng có dữ liệu ứng từng thuộc# tính)  print(df.count().sort\_values()) #df.count(): đếm số lượng dòng có dữ liệu của df, .sort\_values() sx tăng dân  df = df.drop(*columns*=['FT%', '3P%', '2P%', 'FG%', 'eFG%'], *axis*=1)  print(df.shape) # kiểm tra lại số lượng cột & dòng của df sau khi XL NULL cột  #####################################  # # Bước 4: Xử lý DÒNG dữ liệu NULL  ###################################  # Removing null values (Xóa tất cả các dòng có giá trị null trong tập FRAME dữ liệu.)  df = df.dropna(*how*='any')  print(df.shape) # kiểm tra lại số lượng cột & dòng của df sau khi XL NULL các dòng DL  ##################################################################  # # Bước 5: Xử lý loại bỏ các giá trị ngoại lệ (cá biệt): isolated  ##################################################################  # #kiểm tra tập dữ liệu có bất kỳ ngoại lệ nào không  z = np.abs(stats.zscore(df.\_get\_numeric\_data())) # Dò tìm và lấy các giá trị cá biệt trong tập dữ liệu gốc thông qua điểm z (z\_score)  print('MA TRAN Z-SCORE\n')  print(z) # in ra tập (ma trận) các giá trị z-score từ tập dữ liệu gốc  df= df[(z < 3).all(*axis*=1)] # kiểm tra và chỉ giữ lại trong df các giá trị số liệu tưng ứng với z-score < 3  # {loại các giá trị >= 3} vì các giá trị z-score >=3 tướng ứng với số liệu quá khác biệt so với các số liệu còn lại (“cá biệt” = “ngoại lệ” = isolated}  print(df.shape) # xác định số dòng & cột dữ liệu sau khu xử lý các giá trị cá biệt  ##############################################################################  # # Bước 6: RR THEO Mã hóa trực tiếp  ##############################################################################  # Mã hóa các giá trị trong cột 'Pos' bằng cách sử dụng 'replace'  df['Pos'].replace({'PG': 1, 'SG': 1, 'SF': 0, 'PF': 0, 'C': 0}, *inplace*=True)  # Xóa các dòng có giá trị 'Pos' không nằm trong danh sách đã định  df = df[df['Pos'].isin([1, 2, 3, 4, 5])]# Kiểm tra kết quả  # Kiểm tra lại phân phối các giá trị trong cột 'Pos'  print(df['Pos'])  ####################################################################  # #Bước 7: RR hóa theo khoảng / đoạn = Chuẩn hóa (Rời rạc hóa) tập dữ liệu Input dùng ..MaxMin  ####################################################################  df\_num = df.drop(*columns*=['Player', 'Tm'], *axis*=1)  rr = preprocessing.MinMaxScaler()  # xác định thang đo sẽ dùng để RR hóa theo khoảng đều  rr.fit(df\_num)  # Áp dụng thang đo vào data frame [df] của đề tài (đã tiền xử lý đến Bước 5)  df\_num = pd.DataFrame(rr.transform(df\_num), *index*=df.index, *columns*=df\_num.columns)  # Chuyển đổi dữ liệu  df\_num.iloc[4:10]  # Lấy 5 dòng dữ liệu từ chỉ số 4 đến 10  print(df\_num)  # In ra DataFrame đã được chuẩn hóa    ################################################################################################################  # # GIAI ĐOẠN 3: PHÂN TÍCH DỮ LIỆU THĂM DÒ : EDA [CƠ SỞ = HỌC CÁC MÔN data Science, AI, ML và DeepML,... ]  ################################################################################################################  ####################################################################  # #Bước 8: Xác định mô hình trích lọc các thuộc tính đặc trưng: EDA  ####################################################################  X = df\_num.loc[:, df\_num.columns != 'Pos']  # Tất cả các cột trừ cột 'Pos'  y = df\_num[['Pos']]  # Dữ liệu đầu ra là cột 'Pos'  # Loại bỏ các dòng có giá trị thiếu  selector = SelectKBest(chi2, *k*=3) # sd các hàm ... trong thư viện sklearn = Mô hình xác định các Thuộc tính quan trọng quyết định việc dự đoàn DL output = trích lọc Đặc trưng = Feature Extraction: k = 1...12 (đ/v bài này)  selector.fit(X, y) # Áp dụng mô hình trên vào ....  X\_new = selector.transform(X) # Chuyên DL Input teho mô hình  print(X\_new)  print(y)  print('k cot quan trong nhat quyet dinh cho Vector Output')  print(X.columns[selector.get\_support(*indices*=True)]) # in ds các tt đặc trưng  ####################################################################  # #Bước 9: Xác định mô hình trích lọc các thuộc tính đặc trưng  ####################################################################  # # XĐ data frame = Chiếu lấy các thuộc tính đặc trưng đã xđ trong B8  df\_new = df\_num[['TOV','PF','PTS']]  # TOV (Turnovers per game): Số lần cầu thủ mất bóng hoặc bị đối phương cướp bóng mỗi trận.  # PF (Personal fouls per game): Số lần cầu thủ phạm lỗi cá nhân mỗi trận.  # PTS (Points per game): Số điểm cầu thủ ghi được trung bình mỗi trận.  ####################################################################  # #Bước 10: EDA theo nhu cầu thực tế => input vào các mô hình AI, ML,...  ####################################################################  X = df\_new[['PF']]  y = df\_new[['PTS']]  print(X)  print(y) |

# CHƯƠNG 5: LẬP TRÌNH PYTHON HỖ TRỢ CV NHẬN DIỆN HÌNH HỌC

## 5.1 Giới thiệu chủ đề lập trình

**Image Processing (Xử lý ảnh):**  
Thực hiện các thao tác xử lý đơn giản trên ảnh như:

* Quay, co giãn, lật, cắt ảnh
* Chuyển đổi ảnh giữa các hệ màu (ví dụ: RGB ↔ Grayscale)
* Histogram, rời rạc hóa ảnh, phân tích EDA
* Nhận diện biên ảnh và các thao tác xử lý cơ bản khác.

**Frames from Video (Cắt khung ảnh từ Video):**

* Cắt các khung ảnh (frames) từ video clip để phục vụ cho các phân tích hoặc xử lý tiếp theo.

## 5.2 Yêu cầu bài toán

1. Sv cá nhân hóa thông tin bài làm
2. YÊU CẦU 1: CHO PHÉP CHỌN FILE Video (Dùng text liệt kê & chọn: tham khảo: MauCode Choice.pdf): ...mp4 OR .mov,...
3. ..
4. YÊU CẦU 2: CHO PHÉP CHỌN PHƯƠNG ÁN XỬ LÝ TRÙNG CÁC FILE FRAMEs CẮT ĐƯỢC
5. + PA1: THAY ĐỔI TÊN các FILE FRAME THEO LẦN CHẠY
6. + PA2: LƯU CHÔNG LÊN FILEs CŨ
7. + PA3: CHO NSD CHỌN TÊN FILE CHO CÁC FAMREs CẮT ĐƯỢC, nên gợi ý NSD, Ex: <Tên\_file><Chỉ số>
8. + PA4: cHO PHÉP: CHỌN PATH LƯU CÁC FRAMEs CẮT ĐƯỢC & NEW FOLDER (nếu cần)
9. + SV TỰ THÊM CÁC PA KHÁC (nếu có)
10. YÊU CẦU 3: MỞ VIDEO = CHO NSD CHỌN THỜI ĐIỂM CẮT FANRE THEO Ý MUỐN
11. + x.đ độ dài
12. YÊU CẦU 4: CHO PHÉP CHỌN THỜI ĐIỂM BẮT ĐẦU & KẾT THÚC TRONG VIDEO GỐC = CẦN CẮT FRAMEs (cần lấy times duration của Video)
13. •YÊU CẦU 5: GÉP BÀI XỬ LÝ ẢNH VÀO BÀI NÀY=XỬ LÝ ẢNH TRÊN FRAMEs CẮT ĐƯỢC (dùng Text => chọn: tham khảo: MauCode Choice.pdf)

## 5.3 Các thư viện liên quan

**os**: Thư viện hỗ trợ tương tác với hệ điều hành như quản lý tệp và thư mục.

**cv2**: Thư viện xử lý ảnh và video mạnh mẽ với các thuật toán thị giác máy tính.

**tkinter**: Thư viện tạo giao diện người dùng đồ họa (GUI) trong Python.

**filedialog**: Cung cấp hộp thoại chọn tệp hoặc thư mục trong GUI của tkinter.

**numpy**: Thư viện xử lý mảng và tính toán khoa học hiệu quả trong Python.

## 5.3 Cắt frames ảnh từ Video Clips

### 5.3.1 Full code tạo video chứa những hình học

Đầu tiên, mã thiết lập các thông số cho video như độ phân giải của các frame, số lượng frame mỗi giây (FPS), và tổng số hình cần hiển thị trong video. Sau đó, chương trình tạo một loạt các frame trắng và vẽ lên các hình học (hình tròn, hình vuông và đường thẳng) với các thông số như vị trí, màu sắc và độ dày đường vẽ. Các frame này sau đó được lưu vào một video với tên và định dạng xác định. Video được lặp lại nhiều lần để tạo thành một đoạn video dài hơn.

|  |
| --- |
| import cv2  import numpy as np  import os  # Đặc tả các thông số cho video  frame\_width = 640  frame\_height = 480  fps = 1  # Mỗi frame sẽ hiển thị trong 1 giây  frame\_count = 3  # Tổng số hình bạn muốn hiển thị (1 hình = 1 frame)  # Đường dẫn lưu video vào thư mục cụ thể  save\_dir = r"D:\PYPR *(*Python Programming*)*\G127CTNPNBAPlayer\G127CTNPNBAPLayer20CV\ngocphung\_27"  # Kiểm tra và tạo thư mục nếu chưa tồn tại  if not os.path.exists(save\_dir):      os.makedirs(save\_dir)  # Khởi tạo video writer để lưu video  fourcc = cv2.VideoWriter\_fourcc(\*'mp4v')  # Định dạng video  video\_out = cv2.VideoWriter(os.path.join(save\_dir, 'ngocphung\_27\_create\_video.mp4'), fourcc, fps, (frame\_width, frame\_height))  # Tạo video với hình học cơ bản  frame\_list = []  # Danh sách chứa các frame hình học  # Tạo các frame với hình học  for i in range(frame\_count):      # Tạo một frame trắng      frame = np.ones((frame\_height, frame\_width, 3), *dtype*=np.uint8) \* 255        # Vẽ hình học tùy theo số frame      if i == 0:  # Vẽ hình tròn          center = (*int*(frame\_width / 2), *int*(frame\_height / 2))          radius = 50  # Kích thước hình tròn          color = (0, 255, 0)  # Màu xanh lá cây          thickness = 3          frame = cv2.circle(frame, center, radius, color, thickness)      elif i == 1:  # Vẽ hình vuông          top\_left = (50, 200)          bottom\_right = (frame\_width - 50, 300)          frame = cv2.rectangle(frame, top\_left, bottom\_right, (255, 0, 0), 2)  # Màu xanh dương      elif i == 2:  # Vẽ đường thẳng          line\_start = (50 + i \* 5, 50)          line\_end = (50 + i \* 5, frame\_height - 50)          frame = cv2.line(frame, line\_start, line\_end, (0, 0, 255), 2)  # Màu đỏ        # Thêm frame vào danh sách      frame\_list.append(frame)  # Số lượng vòng lặp trong video (lặp lại các hình học)  repeat\_count = 10  # Video sẽ lặp lại 10 lần  # Tạo video với các frame đã tạo trước đó  for \_ in range(repeat\_count):      for frame in frame\_list:          # Lưu từng frame vào video          video\_out.write(frame)          # Hiển thị frame          cv2.imshow("Frame", frame)          cv2.waitKey(1000)  # Đợi 1 giây (1000 ms) giữa các frame  # Giải phóng đối tượng VideoWriter và đóng tất cả các cửa sổ  video\_out.release()  cv2.destroyAllWindows()  print("Video đã được tạo và lưu thành công!") |

Kết quả video

A screenshot of a computer

Description automatically generated

## 5.4 Full codes bài làm

Sau khi tạo xong video, mã tiếp tục mở video vừa tạo ra và áp dụng phương pháp nhận diện hình học. Cụ thể, mã sử dụng hàm **cv2.HoughCircles** để nhận diện các hình tròn trong video. Quá trình nhận diện chuyển đổi các frame của video sang ảnh grayscale (đen trắng) để tăng độ chính xác khi tìm kiếm các hình tròn. Nếu tìm thấy hình tròn trong một frame, các vòng tròn này sẽ được vẽ lại trên video, và chương trình tiếp tục kiểm tra các frame tiếp theo để nhận diện các hình tròn, vuông, hoặc đường thẳng khác nếu cần.

|  |
| --- |
| import os  # Thêm dòng này để import thư viện os  import cv2  import tkinter as tk  from tkinter import filedialog  import numpy as np  # B2: HÀM MỞ file Clips= cắt frames  def OpenFile():      # biến toàn cục      global filepath      # Hộp thoại mở thư mục      filepath = filedialog.askopenfilename(*title*="Chọn Clip File", *filetypes*=(("Clips File (.mp4)", "\*.mp4"), ("Movie File (.mov)", "\*.mov")))      lblfile.configure(*text*="File Video: %s" % filepath)  # B3: HÀM cắt frames và nhận diện hình học  def FramesCap():      cap = cv2.VideoCapture(filepath)      count = 0  # biến đếm số khung hình -> bắt đầu từ số 0      output\_dir = r"D:/PYPR *(*Python Programming*)*/G127CTNPNBAPlayer/G127CTNPNBAPLayer20CV/ngocphung\_27"  # Đường dẫn thư mục lưu ảnh        # Kiểm tra nếu thư mục chưa tồn tại, tạo thư mục mới      if not os.path.exists(output\_dir):          os.makedirs(output\_dir)      while cap.isOpened():  # trong khi Video clip đang còn phát          ret, frame = cap.read()  # chụp ra một khung hình: khung chụp được lưu vào biến frame          if not ret:              break            # Tiền xử lý hình ảnh (chuyển sang ảnh xám)          gray = cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY)          blurred = cv2.GaussianBlur(gray, (5, 5), 0)          edges = cv2.Canny(blurred, 50, 150)          # Tìm các hình vuông (hình chữ nhật) bằng cách tìm các contour          contours, \_ = cv2.findContours(edges, cv2.RETR\_TREE, cv2.CHAIN\_APPROX\_SIMPLE)          for contour in contours:              # Lọc các contour quá nhỏ (không phải là hình học)              if cv2.contourArea(contour) < 500:                  continue              # Xấp xỉ contour thành một đa giác              epsilon = 0.04 \* cv2.arcLength(contour, True)              approx = cv2.approxPolyDP(contour, epsilon, True)              # Kiểm tra nếu là hình vuông (hoặc hình chữ nhật)              if len(approx) == 4:                  cv2.drawContours(frame, [approx], -1, (0, 255, 0), 3)  # Vẽ hình vuông/chữ nhật          # Phát hiện hình tròn          circles = cv2.HoughCircles(gray, cv2.HOUGH\_GRADIENT, 1, *minDist*=50, *param1*=50, *param2*=30, *minRadius*=10, *maxRadius*=100)            if circles is not None:              circles = np.round(circles[0, :]).astype("int")              for (x, y, r) in circles:                  cv2.circle(frame, (x, y), r, (0, 0, 255), 4)  # Vẽ hình tròn                  cv2.rectangle(frame, (x - 5, y - 5), (x + 5, y + 5), (0, 128, 255), -1)          # Lưu khung hình vào thư mục đã chỉ định          output\_path = os.path.join(output\_dir, f"Khung{count}.jpg")  # Đường dẫn đầy đủ để lưu ảnh          cv2.imwrite(output\_path, frame)  # Lưu ảnh vào thư mục          # Hiển thị khung hình với các hình học được nhận diện          cv2.imshow('Khung Hinh', frame)            count += 1  # Tăng chỉ count lên 1 để chuẩn bị lưu khung hình kế tiếp          # Chờ gõ phím kết thúc là phím 'q'          if cv2.waitKey(10) & 0xFF == ord('q'):              break      cap.release()  # Giải phóng đối tượng VideoCapture      cv2.destroyAllWindows()  # Đóng tất cả các cửa sổ      lblfile.configure(*text*="Tổng số frames: %d" % count)  # B2: THIẾT LẬP & KHỞI TẠO ĐỐI TƯỢNG FORM  wf = tk.Tk()  wf.title("CẮT KHUNG ẢNH TỪ VIDEO CLIPs VÀ NHẬN DIỆN HÌNH HỌC")  wf.geometry("900x500")  wf.resizable(tk.FALSE, tk.FALSE)  # Thiết lập label  tk.Label(wf, *text*="Chọn file Video:").place(*x*=10, *y*=15)  # Thiết lập 1 button  btnOpenFile = tk.Button(wf, *text*="Open File", *command*=OpenFile)  btnOpenFile.place(*x*=120, *y*=15)  # Thiết lập label hiện tên file  lblfile = tk.Label(wf, *text*="", *relief*=tk.SUNKEN)  lblfile.place(*x*=120, *y*=50)  # Thiết lập 1 button  btnFrCut = tk.Button(wf, *text*="Cắt frames và nhận diện hình học", *command*=FramesCap)  btnFrCut.place(*x*=200, *y*=15)  wf.mainloop() |

## 5.5 Kết quả

- chọn đường dẫn cho file video

A computer screen shot of a computer

Description automatically generated

- Sau khi nhấn Cắt frames và nhận diện hình học sẽ hiện ra kết quả các frame

A screenshot of a computer

Description automatically generated

- Đây là kết quả các ảnh từ video

A screenshot of a computer

Description automatically generated

# CHƯƠNG 6: LẬP TRÌNH PYTHON GAME CAR RACING

## 6.1 Giới thiệu chủ đề Game Car Racing

Game Car Racing (Đua xe) là một thể loại game phổ biến trong ngành công nghiệp trò chơi điện tử, nơi người chơi tham gia vào các cuộc đua xe tốc độ cao trên các đường đua đầy thử thách. Mục tiêu của trò chơi là điều khiển chiếc xe vượt qua các đối thủ hoặc đạt được thành tích tốt nhất trong thời gian ngắn nhất. Với gameplay đơn giản nhưng lôi cuốn, thể loại này thu hút người chơi bởi cảm giác hồi hộp, tốc độ và sự cạnh tranh. Các game đua xe thường được trang bị đồ họa ấn tượng, âm thanh sống động và các chế độ chơi đa dạng như đua đường phố, đua xe thể thao, hoặc đua xe địa hình. Những trò chơi này không chỉ mang lại sự giải trí mà còn giúp người chơi phát triển kỹ năng điều khiển, phản xạ nhanh và khả năng xử lý tình huống.

## 6.2 Lý thuyết đơn giản về Lập trình Game Car Racing

Vòng lặp game: Trong các game, thường có một vòng lặp vô tận while True: (hoặc while not QUIT) để điều khiển trò chơi cho đến khi người chơi thoát. Vòng lặp này thực hiện ba thao tác chính:

* Lập (vẽ) màn hình (hoặc cửa sổ) trạng thái của game.
* Nhận sự kiện (Event): Lắng nghe và xử lý các sự kiện như phím bấm, chuột, v.v.
* Thay đổi màn hình: Cập nhật và vẽ lại các đối tượng trên cửa sổ game.

Surface: Trong Pygame, Surface là khung (hoặc lớp) trạng thái của game, tương tự như các lớp (layers) trong media. Nó là một "lớp" ảnh trong suốt, và nhiều surface có thể được vẽ chồng lên nhau.

* Trong ví dụ dưới đây, biến DISPLAYSURF là một surface đặc biệt, có kích thước bằng cửa sổ game và nằm dưới cùng, là nơi bạn sẽ vẽ các đối tượng của game.

Sử dụng thư viện Pygame: Để bắt đầu, bạn cần khởi tạo thư viện Pygame bằng lệnh pygame.init() và sau đó sử dụng pygame.display.update() để cập nhật và hiển thị các nội dung được vẽ trên màn hình.

## 6.3 Các thư viện liên quan

Thư viện Pygame dùng để phát triển các trò chơi 2D, cung cấp các công cụ để xử lý đồ họa, âm thanh và sự kiện trong game.

Thư viện sys cung cấp các chức năng tương tác với hệ điều hành và quản lý các tham số của chương trình như tham số dòng lệnh và thoát chương trình.

Thư viện random dùng để tạo các số ngẫu nhiên, hữu ích trong việc tạo ra các sự kiện ngẫu nhiên trong game.

## 6.4 Mô tả thuật toán

huật toán của game đua xe này có thể được tóm tắt như sau:

**1. Khởi tạo (Initialization)**

* **Khởi tạo Pygame**: Sử dụng pygame.init() để khởi tạo thư viện Pygame.
* **Thiết lập cửa sổ game**: Xác định kích thước cửa sổ (400x600), tải ảnh nền, xe và vật cản.
* **Khởi tạo các đối tượng**:
  + **Background**: Đối tượng nền có khả năng cuộn.
  + **Car**: Đối tượng xe có thể di chuyển theo các hướng.
  + **Obstacles**: Các vật cản xuất hiện ngẫu nhiên trên các làn đường.
  + **Score**: Đối tượng điểm số, hiển thị điểm của người chơi.

**2. Quy trình chơi game**

* **Game Start**:
  + Hiển thị màn hình bắt đầu với chữ "RACING" và yêu cầu nhấn phím "space" để bắt đầu chơi.
* **Game Play**:
  + Xe di chuyển theo các phím mũi tên (trái, phải, lên, xuống).
  + Các vật cản được sinh ra và di chuyển từ trên xuống dưới.
  + Người chơi phải tránh các vật cản bằng cách điều khiển xe.
  + Điểm số tăng dần theo thời gian.
* **Kiểm tra va chạm (Collision Detection)**:
  + Nếu xe va chạm với vật cản, game kết thúc.
* **Game Over**:
  + Khi game kết thúc, hiển thị màn hình "GAME OVER" và yêu cầu nhấn "space" để chơi lại.

**3. Các lớp chính**

* **Background**: Quản lý việc cuộn nền (tốc độ cuộn và cập nhật vị trí).
* **Car**: Quản lý di chuyển của xe và vẽ xe lên màn hình.
* **Obstacles**: Quản lý vật cản, tạo và di chuyển chúng.
* **Score**: Cập nhật và hiển thị điểm số của người chơi.
* **Collision Detection**: Kiểm tra va chạm giữa xe và vật cản.

**4. Vòng lặp chính (Main Loop)**

* Chạy vòng lặp chính với các bước sau:
  + Kiểm tra sự kiện (nhấn phím, đóng cửa sổ).
  + Cập nhật vị trí của nền, xe và vật cản.
  + Vẽ các đối tượng lên màn hình.
  + Kiểm tra va chạm và cập nhật điểm số.

**5. Game Over**

* Khi người chơi va chạm với vật cản, hiển thị màn hình "GAME OVER" và cho phép chơi lại.

**6. Kết thúc**

* Khi người chơi thoát khỏi game, chương trình dừng và đóng cửa sổ game.

## 6.5 Giao diện trò chơi

Xe người chơi



Xe chướng ngại vật



Hình nền

A road with white lines and green trees

Description automatically generated

## 6.4 Full codes bài làm

|  |
| --- |
| import pygame, sys, random  # Nhập các thư viện pygame, sys và random  from pygame.locals import \*  # Nhập các hằng số và các sự kiện từ pygame  WINDOWWIDTH = 400  # Đặt chiều rộng cửa sổ game  WINDOWHEIGHT = 600  # Đặt chiều cao cửa sổ game  pygame.init()  # Khởi tạo pygame  FPS = 60  # Số khung hình trên giây (Frames per second)  fpsClock = pygame.time.Clock()  # Đặt đồng hồ lặp theo nhịp FPS  BGSPEED = 1.5  # Tốc độ cuộn nền  BGIMG = pygame.image.load(r'D:\PYPR *(*Python Programming*)*\G127CTNPNBAPlayer\G127CTNPNBAPLayer30Game*\b*ack\_ground*.*png')  # Tải ảnh nền  DISPLAYSURF = pygame.display.set\_mode((WINDOWWIDTH, WINDOWHEIGHT))  # Tạo cửa sổ game với kích thước đã định  pygame.display.set\_caption('stt Ho Tên = Ex3.0: Game = Game ĐUA XE')  # Đặt tiêu đề cho cửa sổ game  class Background():  # Lớp nền của game      def \_\_init\_\_(*self*):  *self*.x = 0  # Vị trí x của nền  *self*.y = 0  # Vị trí y của nền  *self*.speed = BGSPEED  # Tốc độ cuộn nền  *self*.img = BGIMG  # Hình nền  *self*.width = *self*.img.get\_width()  # Chiều rộng của nền  *self*.height = *self*.img.get\_height()  # Chiều cao của nền      def draw(*self*):  # Vẽ nền          DISPLAYSURF.blit(*self*.img, (*int*(*self*.x), *int*(*self*.y)))  # Vẽ nền tại vị trí x, y          DISPLAYSURF.blit(*self*.img, (*int*(*self*.x), *int*(*self*.y-*self*.height)))  # Vẽ nền tiếp theo      def update(*self*):  # Cập nhật vị trí nền  *self*.y += *self*.speed  # Di chuyển nền xuống dưới          if *self*.y > *self*.height:  # Nếu nền đã cuộn hết thì reset lại vị trí  *self*.y -= *self*.height  X\_MARGIN = 80  # Khoảng cách bên trái và phải của xe  CARWIDTH = 40  # Chiều rộng của xe  CARHEIGHT = 60  # Chiều cao của xe  CARSPEED = 3  # Tốc độ di chuyển của xe  CARIMG = pygame.image.load(r'D:\PYPR *(*Python Programming*)*\G127CTNPNBAPlayer\G127CTNPNBAPLayer30Game\car\_icon\_1*.*png')  # Hình ảnh xe  class Car():  # Lớp xe      def \_\_init\_\_(*self*):  *self*.width = CARWIDTH  # Đặt chiều rộng cho xe  *self*.height = CARHEIGHT  # Đặt chiều cao cho xe  *self*.x = (WINDOWWIDTH-*self*.width)/2  # Vị trí x của xe (giữa màn hình)  *self*.y = (WINDOWHEIGHT-*self*.height)/2  # Vị trí y của xe (giữa màn hình)  *self*.speed = CARSPEED  # Tốc độ di chuyển của xe  *self*.surface = pygame.Surface((*self*.width, *self*.height))  # Tạo một bề mặt cho xe  *self*.surface.fill((255, 255, 255))  # Đặt màu trắng cho xe      def draw(*self*):  # Vẽ xe          DISPLAYSURF.blit(CARIMG, (*int*(*self*.x), *int*(*self*.y)))  # Vẽ hình ảnh xe lên màn hình      def update(*self*, *moveLeft*, *moveRight*, *moveUp*, *moveDown*):  # Cập nhật vị trí của xe          if moveLeft:  # Nếu người chơi nhấn phím trái  *self*.x -= *self*.speed  # Di chuyển xe sang trái          if moveRight:  # Nếu người chơi nhấn phím phải  *self*.x += *self*.speed  # Di chuyển xe sang phải          if moveUp:  # Nếu người chơi nhấn phím lên  *self*.y -= *self*.speed  # Di chuyển xe lên          if moveDown:  # Nếu người chơi nhấn phím xuống  *self*.y += *self*.speed  # Di chuyển xe xuống            # Giới hạn di chuyển xe          if *self*.x < X\_MARGIN:  *self*.x = X\_MARGIN  # Xe không được ra ngoài màn hình bên trái          if *self*.x + *self*.width > WINDOWWIDTH - X\_MARGIN:  *self*.x = WINDOWWIDTH - X\_MARGIN - *self*.width  # Xe không được ra ngoài màn hình bên phải          if *self*.y < 0:  *self*.y = 0  # Xe không được ra ngoài màn hình trên          if *self*.y + *self*.height > WINDOWHEIGHT:  *self*.y = WINDOWHEIGHT - *self*.height  # Xe không được ra ngoài màn hình dưới  LANEWIDTH = 60  # Chiều rộng mỗi làn đường  DISTANCE = 200  # Khoảng cách giữa các xe cản trở  OBSTACLESSPEED = 2  # Tốc độ của các vật cản  CHANGESPEED = 0.001  # Tốc độ tăng dần của vật cản  OBSTACLESIMG = pygame.image.load(r'D:\PYPR *(*Python Programming*)*\G127CTNPNBAPlayer\G127CTNPNBAPLayer30Game\car\_icon\_2*.*png')  # Hình ảnh vật cản  class Obstacles():  # Lớp vật cản      def \_\_init\_\_(*self*):  *self*.width = CARWIDTH  # Đặt chiều rộng cho vật cản  *self*.height = CARHEIGHT  # Đặt chiều cao cho vật cản  *self*.distance = DISTANCE  # Khoảng cách giữa các vật cản  *self*.speed = OBSTACLESSPEED  # Tốc độ của vật cản  *self*.changeSpeed = CHANGESPEED  # Tốc độ thay đổi của vật cản  *self*.ls = []  # Danh sách các vật cản          for i in range(5):  # Khởi tạo 5 vật cản              y = -CARHEIGHT - i \* *self*.distance  # Đặt vị trí ban đầu của vật cản              lane = random.randint(0, 3)  # Chọn ngẫu nhiên làn đường  *self*.ls.append([lane, y])  # Thêm vật cản vào danh sách      def draw(*self*):  # Vẽ các vật cản          for i in range(5):  # Lặp qua tất cả các vật cản              x = *int*(X\_MARGIN + *self*.ls[i][0] \* LANEWIDTH + (LANEWIDTH - *self*.width) / 2)  # Tính toán vị trí x của vật cản              y = *int*(*self*.ls[i][1])  # Lấy vị trí y của vật cản              DISPLAYSURF.blit(OBSTACLESIMG, (x, y))  # Vẽ vật cản lên màn hình      def update(*self*):  # Cập nhật vị trí của các vật cản          for i in range(5):  # Lặp qua tất cả các vật cản  *self*.ls[i][1] += *self*.speed  # Di chuyển vật cản xuống dưới  *self*.speed += *self*.changeSpeed  # Tăng dần tốc độ vật cản          if *self*.ls[0][1] > WINDOWHEIGHT:  # Nếu vật cản ra ngoài màn hình  *self*.ls.pop(0)  # Xóa vật cản đầu tiên              y = *self*.ls[3][1] - *self*.distance  # Đặt vị trí mới cho vật cản              lane = random.randint(0, 3)  # Chọn ngẫu nhiên làn đường  *self*.ls.append([lane, y])  # Thêm vật cản vào cuối danh sách  class Score():  # Lớp điểm số      def \_\_init\_\_(*self*):  *self*.score = 0  # Khởi tạo điểm số      def draw(*self*):  # Vẽ điểm số          font = pygame.font.SysFont('consolas', 30)  # Chọn font chữ          scoreSuface = font.render('Score: '+*str*(*int*(*self*.score)), True, (0, 0, 0))  # Tạo bề mặt điểm số          DISPLAYSURF.blit(scoreSuface, (10, 10))  # Vẽ điểm số lên màn hình      def update(*self*):  # Cập nhật điểm số  *self*.score += 0.02  # Tăng điểm theo thời gian  def rectCollision(*rect1*, *rect2*):  # Kiểm tra va chạm giữa 2 hình chữ nhật      if rect1[0] <= rect2[0] + rect2[2] and rect2[0] <= rect1[0] + rect1[2] and rect1[1] <= rect2[1] + rect2[3] and rect2[1] <= rect1[1] + rect1[3]:          return True      return False  def isGameover(*car*, *obstacles*):  # Kiểm tra nếu game over      carRect = [car.x, car.y, car.width, car.height]  # Vị trí và kích thước của xe      for i in range(5):  # Lặp qua các vật cản          x = *int*(X\_MARGIN + obstacles.ls[i][0] \* LANEWIDTH + (LANEWIDTH - obstacles.width) / 2)  # Tính toán vị trí x của vật cản          y = *int*(obstacles.ls[i][1])  # Lấy vị trí y của vật cản          obstaclesRect = [x, y, obstacles.width, obstacles.height]  # Vị trí và kích thước của vật cản          if rectCollision(carRect, obstaclesRect):  # Kiểm tra va chạm              return True      return False  def gameOver(*bg*, *car*, *obstacles*, *score*):  # Hàm game over      font = pygame.font.SysFont('consolas', 60)  # Chọn font chữ      headingSuface = font.render('GAMEOVER', True, (255, 0, 0))  # Tạo chữ 'GAMEOVER'      headingSize = headingSuface.get\_size()  # Lấy kích thước chữ      font = pygame.font.SysFont('consolas', 20)  # Chọn font chữ nhỏ cho hướng dẫn      commentSuface = font.render('Press "space" to replay', True, (0, 0, 0))  # Tạo chữ hướng dẫn      commentSize = commentSuface.get\_size()  # Lấy kích thước chữ hướng dẫn      while True:          for event in pygame.event.get():              if event.type == pygame.QUIT:  # Nếu đóng cửa sổ game                  pygame.quit()                  sys.exit()              if event.type == pygame.KEYUP:  # Nếu nhấn phím                  if event.key == K\_SPACE:  # Nhấn phím space để chơi lại                      return          bg.draw()  # Vẽ nền          car.draw()  # Vẽ xe          obstacles.draw()  # Vẽ vật cản          score.draw()  # Vẽ điểm số          DISPLAYSURF.blit(headingSuface, (*int*((WINDOWWIDTH - headingSize[0]) / 2), 100))  # Vẽ chữ 'GAMEOVER'          DISPLAYSURF.blit(commentSuface, (*int*((WINDOWWIDTH - commentSize[0]) / 2), 400))  # Vẽ hướng dẫn          pygame.display.update()  # Cập nhật màn hình          fpsClock.tick(FPS)  # Điều chỉnh tốc độ khung hình  def gameStart(*bg*):  # Hàm bắt đầu game      bg.\_\_init\_\_()  # Khởi tạo lại nền      font = pygame.font.SysFont('consolas', 60)  # Chọn font chữ      headingSuface = font.render('RACING', True, (255, 0, 0))  # Tạo chữ 'RACING'      headingSize = headingSuface.get\_size()  # Lấy kích thước chữ      font = pygame.font.SysFont('consolas', 20)  # Chọn font chữ nhỏ cho hướng dẫn      commentSuface = font.render('Press "space" to play', True, (0, 0, 0))  # Tạo chữ hướng dẫn      commentSize = commentSuface.get\_size()  # Lấy kích thước chữ hướng dẫn      while True:          for event in pygame.event.get():              if event.type == pygame.QUIT:  # Nếu đóng cửa sổ game                  pygame.quit()                  sys.exit()              if event.type == pygame.KEYUP:  # Nếu nhấn phím                  if event.key == K\_SPACE:  # Nhấn phím space để bắt đầu chơi                      return          bg.draw()  # Vẽ nền          DISPLAYSURF.blit(headingSuface, (*int*((WINDOWWIDTH - headingSize[0]) / 2), 100))  # Vẽ chữ 'RACING'          DISPLAYSURF.blit(commentSuface, (*int*((WINDOWWIDTH - commentSize[0]) / 2), 400))  # Vẽ hướng dẫn          pygame.display.update()  # Cập nhật màn hình          fpsClock.tick(FPS)  # Điều chỉnh tốc độ khung hình  def gamePlay(*bg*, *car*, *obstacles*, *score*):  # Hàm chơi game      car.\_\_init\_\_()  # Khởi tạo lại xe      obstacles.\_\_init\_\_()  # Khởi tạo lại vật cản      bg.\_\_init\_\_()  # Khởi tạo lại nền      score.\_\_init\_\_()  # Khởi tạo lại điểm số      moveLeft = moveRight = moveUp = moveDown = False  # Khởi tạo trạng thái di chuyển      while True:          for event in pygame.event.get():              if event.type == pygame.QUIT:  # Nếu đóng cửa sổ game                  pygame.quit()                  sys.exit()              if event.type == KEYDOWN:  # Nếu nhấn phím                  if event.key == K\_LEFT:  # Nhấn phím trái để di chuyển trái                      moveLeft = True                  if event.key == K\_RIGHT:  # Nhấn phím phải để di chuyển phải                      moveRight = True                  if event.key == K\_UP:  # Nhấn phím lên để di chuyển lên                      moveUp = True                  if event.key == K\_DOWN:  # Nhấn phím xuống để di chuyển xuống                      moveDown = True              if event.type == KEYUP:  # Nếu nhả phím                  if event.key == K\_LEFT:  # Nhả phím trái                      moveLeft = False                  if event.key == K\_RIGHT:  # Nhả phím phải                      moveRight = False                  if event.key == K\_UP:  # Nhả phím lên                      moveUp = False                  if event.key == K\_DOWN:  # Nhả phím xuống                      moveDown = False          if isGameover(car, obstacles):  # Kiểm tra nếu game over              return          bg.draw()  # Vẽ nền          bg.update()  # Cập nhật nền          car.draw()  # Vẽ xe          car.update(moveLeft, moveRight, moveUp, moveDown)  # Cập nhật vị trí xe          obstacles.draw()  # Vẽ vật cản          obstacles.update()  # Cập nhật vật cản          score.draw()  # Vẽ điểm số          score.update()  # Cập nhật điểm số          pygame.display.update()  # Cập nhật màn hình          fpsClock.tick(FPS)  # Điều chỉnh tốc độ khung hình  def main():  # Hàm chính      bg = Background()  # Tạo đối tượng nền      car = Car()  # Tạo đối tượng xe      obstacles = Obstacles()  # Tạo đối tượng vật cản      score = Score()  # Tạo đối tượng điểm số      gameStart(bg)  # Bắt đầu game      while True:          gamePlay(bg, car, obstacles, score)  # Chơi game          gameOver(bg, car, obstacles, score)  # Khi game kết thúc thì hiển thị game over  if \_\_name\_\_ == '\_\_main\_\_':  # Nếu chương trình được chạy trực tiếp      main()  # Gọi hàm chính |

## 6.5 Kết quả

Giao diện của trò chơi

A video game screen shot

Description automatically generated A screenshot of a video game

Description automatically generated A screenshot of a video game

Description automatically generated

# CHƯƠNG 7: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN CỦA ĐỒ ÁN HỌC PHẦN

## 7.1. Những kết quả đạt được

Đồ án học phần đã thực hiện thành công việc lập trình Python cho dự án G127CTNPNBAPlayer, với các tính năng nổi bật bao gồm. Giao diện đồ họa (GUI): Đã thiết kế và triển khai giao diện người dùng thân thiện, trực quan, giúp người dùng dễ dàng tương tác với các tính năng của ứng dụng. Xử lý giọng nói trợ lý ảo (Voice Assistant): Tính năng trợ lý ảo đã được tích hợp, cho phép người dùng ra lệnh và điều khiển ứng dụng thông qua giọng nói. Phân tích khám phá trên tập dữ liệu thực nghiệm: Đã thực hiện các phân tích khám phá trên tập dữ liệu thực nghiệm, bao gồm việc vẽ đồ thị (Plot) và hiển thị các kết quả phân tích.

Computer Vision (CV): Ứng dụng đã triển khai các kỹ thuật Computer Vision, phục vụ cho việc nhận diện và xử lý hình ảnh trong các bài toán thực tiễn. Game: Đã phát triển một trò chơi đơn giản sử dụng Pygame, mang lại trải nghiệm giải trí cho người dùng.

## 7.2. Hạn chế & Hướng khắc phục các hạn chế

Liên kết voice assistant để ra lệnh phân tích khám phá bằng lời nói: Mặc dù tính năng trợ lý ảo đã được tích hợp, nhưng vẫn chưa hoàn thiện khả năng nhận diện và ra lệnh thực hiện các phân tích dữ liệu bằng giọng. Chưa hoàn thiện một số tính năng trong game: Trò chơi đua xe được phát triển mới chỉ hoàn thiện các tính năng cơ bản, chưa có các yếu tố nâng cao như âm thanh, hiệu ứng đặc biệt, và khả năng mở rộng. Để khắc phục những hạn chế này, đồ án sẽ tiếp tục được phát triển với các hướng đi sau. Liên kết voice assistant với các tính năng phân tích khám phá: Cải thiện khả năng nhận diện giọng nói để người dùng có thể ra lệnh phân tích dữ liệu bằng lời nói. Cải thiện trò chơi và thêm các tính năng mới: Phát triển thêm các chức năng như âm thanh, hiệu ứng hình ảnh, và mở rộng nội dung game để tăng tính hấp dẫn.

## 7.3. Hướng mở rộng ĐỒ ÁN HỌC PHẦN

Với nền tảng đã được xây dựng, đồ án sẽ tiếp tục được mở rộng và phát triển với các ứng dụng thực tiễn trong nhiều lĩnh vực. Các hướng mở rộng tiềm năng bao gồm, ứng dụng trợ lý ảo vào các lĩnh vực khác: Tiếp tục cải tiến và tích hợp trợ lý ảo vào các ứng dụng khác như chăm sóc sức khỏe, giáo dục, và các dịch vụ hỗ trợ khách hàng. Ứng dụng phân tích dữ liệu trong các ngành nghề cụ thể: Phát triển các tính năng phân tích dữ liệu cho các lĩnh vực như tài chính, marketing, và khoa học dữ liệu. Mở rộng trò chơi với các tính năng nâng cao: Thêm các tính năng mạng để người chơi có thể thi đấu trực tuyến, hoặc xây dựng các cấp độ mới, tính năng chiến đấu, và nhiều chế độ chơi phong phú hơn. Nhìn chung, đồ án học phần sẽ tiếp tục được hoàn thiện và mở rộng để không chỉ mang lại những sản phẩm hữu ích trong lĩnh vực nghiên cứu mà còn phục vụ các ứng dụng thực tiễn trong đời sống.

# 

# DANH MỤC TÀI LIỆU THAM KHẢO

1. Võ Xuân Thể (2023), Tài liệu học tập Lập trình Python, Trường Đại học Sư phạm kỹ thuật Tp.HCM, Tài liệu lưu hành nội bộ được cung cấp theo từng buổi học
2. Võ Xuân Thể (2023), Bài tập thực hành Lập trình Python, Trường Đại học Sư phạm kỹ thuật Tp.HCM, Tài liệu lưu hành nội bộ được cung cấp theo từng buổi học
3. https://www.kaggle.com/datasets/trnduythanhkhttt/housepricinghcm: nguồn dataFrame được sử dụng trong đề tài này: giá nhà tại Tp.HCM….
4. https://online-video-cutter.com: cắt video phục vụ bài hỗ trợ CV