

TRƯỜNG ĐẠI HỌC KỸ THUẬT CÔNG NGHIỆP

KHOA ĐIỆN TỬ

Bộ môn: Công nghệ Thông tin



BÀI TẬP KẾT THÚC MÔN HỌC
MÔN HỌC
LẬP TRÌNH PYTHON

Thái Nguyên - 2025

TRƯỜNG ĐẠI HỌC KỸ THUẬT CÔNG NGHIỆP

KHOA ĐIỆN TỬ

Bộ môn: Công nghệ Thông tin



BÀI TẬP KẾT THÚC MÔN HỌC
MÔN HỌC
LẬP TRÌNH PYTHON

HỌ TÊN	:	NGUYỄN ĐỨC DƯƠNG
MSV	:	K225480106093
LỚP	:	K58KTP
GIÁO VIÊN HƯỚNG DẪN	:	TS. NGUYỄN VĂN HUY



QR_YOUTUBE

Thái Nguyên - 2025

PHIẾU GIAO ĐỀ TÀI
MÔN HỌC: LẬP TRÌNH PYTHON
BỘ MÔN : CÔNG NGHỆ THÔNG TIN

Sinh viên: Nguyễn Đức Dương

Ngành: Kỹ thuật máy tính

MSV: K225480106093

Lớp: K58KTP

Giáo viên hướng dẫn: TS. Nguyễn Văn Huy

Ngày giao đề: 20/05/2025

Ngày hoàn thành: 09/06/2025

Tên đề tài : Xây ứng dụng Critter Caretaker (Chapter 8) cho phép tạo, cho ăn, cho ngủ critter qua giao diện.

Yêu cầu :

1. Đầu vào – đầu ra:

- Đầu vào: Tên critter từ Entry, các nút hành động.
- Đầu ra: Trạng thái critter (hunger, boredom) hiển thị trên Label.

2. Tính năng yêu cầu:

- Class Critter với attributes và methods.
- GUI: Entry tạo critter, Buttons: “Tạo”, “Cho ăn”, “Chơi”, “Ngủ”.
- Cập nhật trạng thái real-time.
- Bắt lỗi khi chưa tạo critter.

3. Kiểm tra & kết quả mẫu:

- Tạo “Bob” → Hunger=0, Boredom=0.
- Nhấn “Cho ăn” → Hunger giảm, Label cập nhật.

GIÁO VIÊN HƯỚNG DẪN
(Ký và ghi rõ họ tên)

NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN

.....

.....

.....

.....

.....

.....

.....

.....

.....

Thái Nguyên, ngày....tháng.....năm 2025

GIÁO VIÊN HƯỚNG DẪN

(Ký ghi rõ họ tên)

MỤC LỤC

DANH MỤC HÌNH ẢNH	3
DANH MỤC BẢNG	4
LỜI NÓI ĐẦU	5
CHƯƠNG I: GIỚI THIỆU	6
1. Tổng quan về Python	6
1.1. Python là gì ?	6
1.2. Ứng dụng của Python	6
1.3. Lợi thế của Python	8
2. Mô tả đề tài	9
2.1. Giới thiệu đề tài	9
2.2. Tính năng của chương trình	9
2.3. Thách thức	10
2.4. Các kiến thức sử dụng	11
CHƯƠNG II. CƠ SỞ LÝ THUYẾT	12
2.1. Ngôn ngữ lập trình sử dụng	12
2.2. Các kiến thức sử dụng trong chương trình	12
2.2.1. Kiểu dữ liệu List	12
CHƯƠNG III. THIẾT KẾ VÀ XÂY DỰNG CHƯƠNG TRÌNH	16
3.1. Sơ đồ khối hệ thống	16
3.1.1. mô tả các module chính trong chương trình	16
3.1.2. Biểu đồ phân cấp chức năng	17
3.2. Sơ đồ khối các thuật toán chính	19
3.2.1. Thuật toán: Tạo Critter	19
3.2.2. Thuật toán: Chọn Critter	20
3.2.3. Thuật toán: Cho Feed/Play/Sleep	21
3.2.4. Thuật toán: Hiển thị trạng thái Critter	22
3.3. Cấu trúc dữ liệu	23
3.3.1. Bảng Critter	23
3.3.2. Danh sách Critter	23

3.4. Chương trình	24
3.4.1. Các hàm trong chương trình chính	24
CHƯƠNG IV: THỰC NGHIỆM VÀ KẾT LUẬN	26
4.1. Thực nghiệm	26
4.1.1. Chức năng tạo Critter và Lưu vào danh sách	26
4.1.2. Chức năng hành động của Critters và Tâm trạng của Critter...26	
4.2. Kết luận	27
4.2.1. Sản phẩm đã làm đã làm được những gì?	27
4.2.2. Những điều học được	27
4.2.3. Hướng cải tiến trong tương lai	28
TÀI LIỆU THAM KHẢO	29

DANH MỤC HÌNH ẢNH

Hình 1. Ngôn ngữ Python

Hình 2. Tự động hoá và Phát triển phần mềm

Hình 3. Lợi thế của Python

Hình 4. Cấu trúc kiểu dữ liệu list sử dụng trong chương trình

Hình 5. Các lệnh sử dụng liên quan đến kiểu dữ liệu list

Hình 6. Mô phỏng danh sách các critters

Hình 7. Ví dụ để tạo Tkinter

Hình 8. Giao diện Tkinter

Hình 9. Cú pháp Pack()

Hình 10. Ví dụ về việc sử dụng Pack()

Hình 11. Biểu đồ phân cấp chức năng

Hình 12. Sơ đồ thuật toán tạo Critter

Hình 13. Sơ đồ thuật toán chọn Critter

Hình 14. Sơ đồ thuật toán ăn/chơi/ngủ

Hình 15. Sơ đồ thuật toán hiển thị trạng thái Critter

Hình 16. Cấu trúc hàm tạo lớp Critter

Hình 17. Cấu trúc hàm hành động của Critter

Hình 18. Cấu trúc hàm xác định tâm trạng Critter

Hình 19. Cấu trúc hàm tạo Critter

Hình 20. Cấu trúc hàm chọn Critter trong List

Hình 21. Cấu trúc hàm hiển thị trạng thái Critter

Hình 22. Giao diện tạo Critter và List

Hình 23. Giao diện hiển thị tâm trạng và các hành động với Critter

Hình 24. Điều kiện của từng trạng thái

DANH MỤC BẢNG

Bảng 1. Danh mục Widget sử dụng

Bảng 2. Cấu trúc dữ liệu của Critter

LỜI NÓI ĐẦU

Trong thời đại công nghệ phát triển mạnh mẽ như hiện nay, việc ứng dụng lập trình để mô phỏng các hành vi sinh động và tương tác gần gũi với người dùng ngày càng được chú trọng. Đặc biệt, việc kết hợp giữa giao diện đồ họa (GUI) và mô hình đối tượng trong lập trình Python giúp các sản phẩm phần mềm trở nên trực quan, dễ sử dụng và mang tính giáo dục – giải trí cao.

Đề tài "Xây dựng ứng dụng Critter Caretaker bằng Python và Tkinter" được thực hiện với mục tiêu tạo ra một ứng dụng đơn giản nhưng mang tính tương tác, giúp người dùng chăm sóc một hoặc nhiều sinh vật ảo (gọi là Critter). Thông qua các hành động như cho ăn, cho chơi và cho ngủ, người dùng có thể theo dõi sự thay đổi trong tâm trạng và trạng thái của Critter, từ đó hiểu hơn về hành vi phản hồi trong lập trình và mối liên hệ giữa dữ liệu và giao diện.

Ứng dụng được xây dựng hoàn toàn bằng ngôn ngữ Python với thư viện Tkinter – một công cụ phổ biến dùng để phát triển giao diện người dùng. Trong quá trình thực hiện, người viết không chỉ áp dụng các kiến thức về lập trình hướng đối tượng, xử lý sự kiện mà còn rèn luyện kỹ năng thiết kế giao diện, tư duy thuật toán và xử lý tình huống khi phát triển phần mềm.

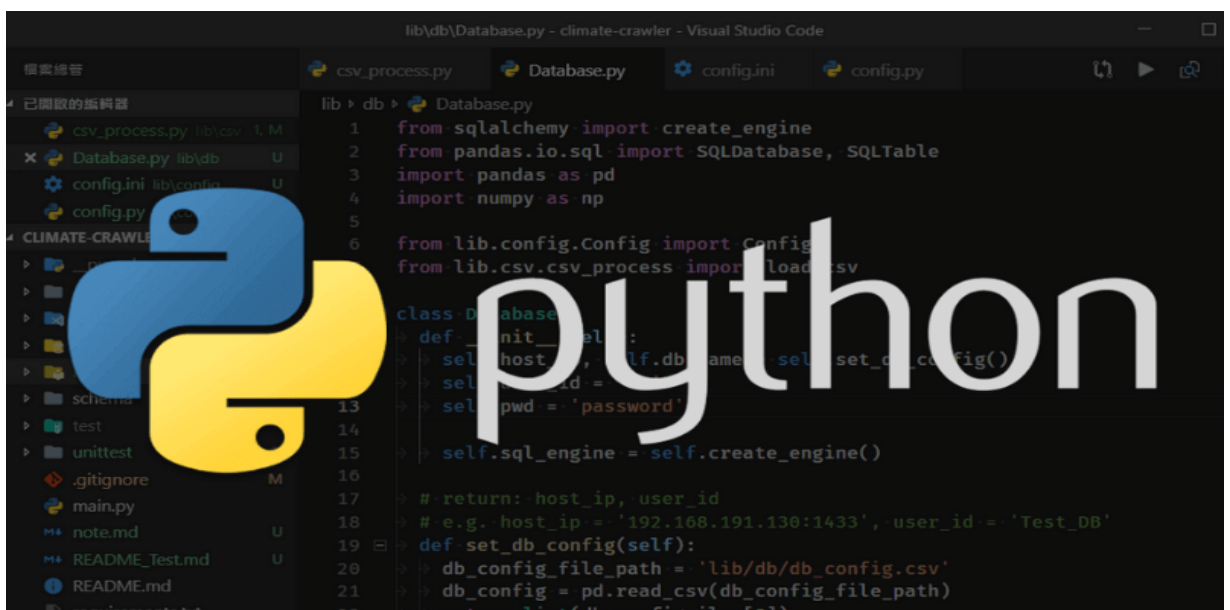
Đề tài này đồng thời là một bước đệm giúp người học tiến gần hơn tới việc xây dựng các ứng dụng quy mô lớn hơn trong tương lai, có thể mở rộng ra nhiều lĩnh vực như trò chơi, phần mềm giáo dục, chăm sóc thú ảo, hoặc các hệ thống tương tác đơn giản cho người dùng.

CHƯƠNG I: GIỚI THIỆU

1. Tổng quan về Python

1.1. Python là gì ?

- Đầu tiên, Python là gì? **Python là ngôn ngữ lập trình máy tính bậc cao** thường được sử dụng để xây dựng trang web và phần mềm, tự động hóa các tác vụ và tiến hành phân tích dữ liệu. Python là ngôn ngữ có mục đích chung, nghĩa là nó có thể được sử dụng để tạo nhiều chương trình khác nhau và không chuyên biệt cho bất kỳ vấn đề cụ thể nào.



Hình 1. Ngôn ngữ Python

1.2. Ứng dụng của Python

a) Phân tích dữ liệu và học máy

- Python đã trở thành một yếu tố chính trong khoa học dữ liệu, cho phép các nhà phân tích dữ liệu và các chuyên gia khác sử dụng ngôn ngữ này để thực hiện các phép tính thống kê phức tạp, tạo trực quan hóa dữ liệu, xây dựng thuật toán học máy, thao tác và phân tích dữ liệu cũng như hoàn thành các nhiệm vụ khác liên quan đến dữ liệu.

- Python có thể xây dựng nhiều dạng trực quan hóa dữ liệu khác nhau, chẳng hạn như biểu đồ đường và thanh, biểu đồ hình tròn, biểu đồ 3D. Python cũng có một số thư viện cho phép các lập trình viên viết chương trình để phân tích dữ liệu và học máy nhanh hơn và hiệu quả hơn, như **TensorFlow** và **Keras**.

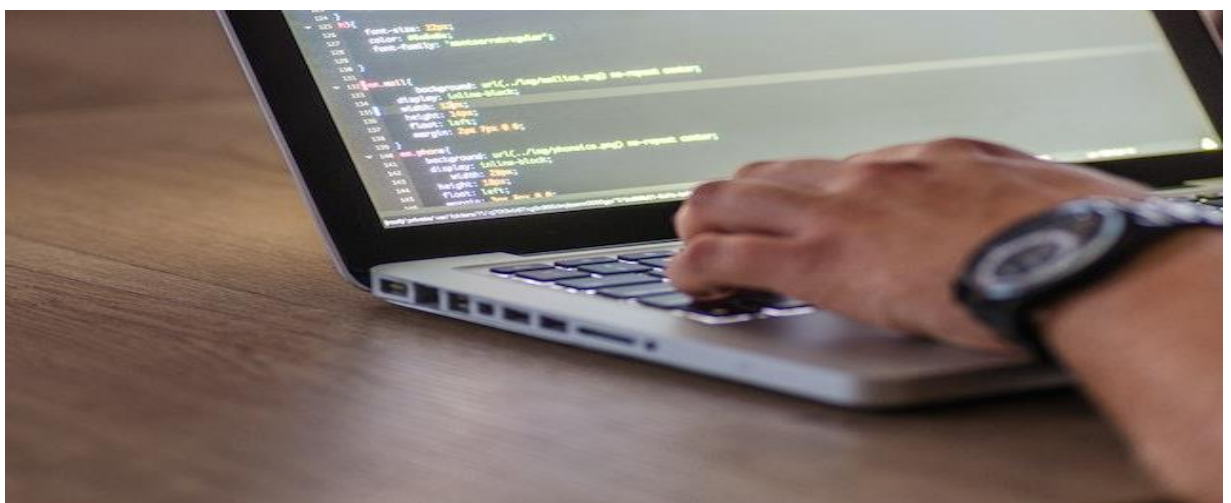
b) Phát triển Web

- Python thường được sử dụng để phát triển **back-end** của trang web hoặc ứng dụng — những phần mà người dùng không nhìn thấy. Vai trò của Python trong phát triển web có thể bao gồm gửi dữ liệu đến và đi từ máy chủ, xử lý dữ liệu và giao tiếp với cơ sở dữ liệu, định tuyến URL và đảm bảo tính bảo mật. Python cung cấp một số khuôn khổ để phát triển web. Những cái thường được sử dụng bao gồm **Django** và **Flask**.

- Một số công việc phát triển web sử dụng Python bao gồm kỹ sư phụ trợ, nhà phát triển Python, kỹ sư phần mềm và kỹ sư **DevOps**.

c) Tự động hoá và Phát triển phần mềm

- Nếu bạn thấy mình thực hiện một nhiệm vụ lặp đi lặp lại, bạn có thể làm việc hiệu quả hơn bằng cách tự động hóa nó bằng Python. Quá trình viết code được sử dụng để xây dựng các quy trình tự động này được gọi là viết script. Trong thế giới mã hóa, tự động hóa có thể được sử dụng để kiểm tra lỗi trên nhiều tệp, chuyển đổi tệp, thực hiện phép toán đơn giản và loại bỏ các bản sao trong dữ liệu.



Hình 2. Tự động hoá và Phát triển phần mềm

- Python thậm chí có thể được sử dụng bởi những người mới bắt đầu để tự động hóa các tác vụ đơn giản trên máy tính—chẳng hạn như đổi tên tệp, tìm và tải xuống nội dung trực tuyến hoặc gửi email hoặc văn bản theo khoảng thời gian mong muốn.

- Trong phát triển phần mềm, Python có thể hỗ trợ các tác vụ như kiểm soát bản dựng, theo dõi lỗi và thử nghiệm. Với Python, các nhà phát triển phần mềm có thể tự động kiểm tra các sản phẩm hoặc tính năng mới. Một số công cụ Python được sử dụng để kiểm thử phần mềm bao gồm Green và Requestium.

1.3. Lợi thế của Python

- Python là một ngôn ngữ dễ đọc và đơn giản để hiểu cho các nhà phát triển chưa bao giờ viết code. Do đó, cộng đồng người dùng Python không ngừng phát triển và lớn mạnh. Có rất nhiều học giả và giáo sư trong cộng đồng người dùng Python.

- Vì vậy, khi xảy ra sự cố, nhà phát triển có thể tập trung vào vấn đề đó và nhận trợ giúp từ những người khác trong cộng đồng mà không phải lo lắng về sự phức tạp của ngôn ngữ.

- Python là một ngôn ngữ lập trình miễn phí và mở. Giấy phép nguồn mở được **OSI** phê chuẩn mà Python được phát triển theo đó làm cho Python trở thành ngôn ngữ tự do sử dụng và phân phối, kể cả cho mục đích thương mại. Nó sẽ làm giảm chi phí của bạn để bảo trì.

- Trong khi các nhà phát triển có thể chia sẻ, sao chép và thay đổi nó. Đối với cộng đồng Python, đây là cơ hội để chia sẻ kiến thức với các chuyên gia cấp dưới.



Hình 3. Lợi thế của Python

2. Mô tả đề tài

2.1. Giới thiệu đề tài

- Đề tài yêu cầu xây dựng một ứng dụng giao diện đồ họa (GUI) sử dụng ngôn ngữ lập trình Python kết hợp với thư viện Tkinter trong môi trường *Visual Studio Code*.

- Ứng dụng được thiết kế nhằm mô phỏng quá trình chăm sóc một Critter – sinh vật ảo, cho phép người dùng tạo Critter mới, cho ăn, cho ngủ và theo dõi trạng thái của Critter thông qua giao diện trực quan. Các thông tin như mức độ đói, chán của Critter sẽ được cập nhật và hiển thị trên màn hình, từ đó tạo ra trải nghiệm tương tác gần gũi và sinh động với người dùng.

- **Đầu vào:** Tên critter từ Entry cho phép người dùng nhập tên cho critter, các nút hành động bao gồm: cho ăn (Feed), ngủ (Sleep), chơi (Play), người dùng có thể sử dụng các nút hành động để tác động đến critter.
- **Đầu ra:** Trạng thái critter (hunger, boredom) hiển thị trên Label, dựa trên hành động mà người dùng thực hiện như chơi, ngủ, ăn sẽ làm thay đổi 2 trạng thái đói (*Hunger*) và chán (*Boredom*).

2.2. Tính năng của chương trình

- Sử dụng lớp Critter để mô phỏng sinh vật ảo, với các thuộc tính như đói và chán, cùng các hành vi như ăn, ngủ và chơi.
- Cung cấp giao diện đồ họa thân thiện bằng thư viện tkinter, cho phép người dùng tạo mới Critter và chọn Critter hiện có để chăm sóc.
- Hiển thị danh sách các Critter đã tạo bằng *Listbox*, cho phép chọn và tương tác trực tiếp với từng Critter riêng biệt.
- Cho phép người dùng thực hiện các hành động như cho ăn, cho chơi, cho ngủ Critter thông qua các nút điều khiển, đồng thời cập nhật trạng thái Critter sau mỗi hành động.
- Hiển thị thông tin trạng thái hiện tại (đói, chán) cùng với tâm trạng của Critter bằng các Label rõ ràng, giúp người dùng dễ theo dõi.
- Xử lý lỗi nhập liệu và lựa chọn chưa hợp lệ, ví dụ: cảnh báo khi người dùng chưa nhập tên Critter hoặc chưa chọn Critter để chăm sóc.
- Thiết kế giao diện trực quan với bố cục rõ ràng và màu sắc sinh động, giúp tăng trải nghiệm người dùng khi tương tác với sinh vật ảo.

2.3. Thách thức

a) *Quản lý trạng thái Critter theo thời gian.*

- Hunger và Boredom phải thay đổi theo thời gian – tức là mỗi hành động người dùng thực hiện (kể cả nói chuyện) đều khiến chỉ số tăng lên nếu không chăm sóc.
- Cần đảm bảo **Logic __pass_time()** được gọi đúng lúc và không bỏ sót.

b) *Đảm bảo tâm trạng phản ánh đúng trạng thái thực*

- Tâm trạng của Critter được tính toán dựa trên tổng ***hunger + boredom***, nhưng phải có các mức cụ thể như: "rất vui", "bình thường", "khó chịu", "giận dữ"...
- Cần xây dựng ngưỡng đánh giá hợp lý để người dùng dễ hiểu và có động lực chăm sóc.

c) *Giới hạn giá trị chỉ số*

- Phải đảm bảo ***hunger*** và ***boredom*** không âm hoặc vượt quá mức nhất định (gây lỗi).
- Dùng `max(0, ...)` hoặc đặt giới hạn hợp lý trong các hàm giảm chỉ số.

d) *Xử lý khi người dùng chưa tạo Critter*

- Nếu người dùng nhấn “Cho ăn”, “Chơi” hoặc “Ngủ” khi chưa khởi tạo Critter:
- Chương trình có thể bị lỗi ***AttributeError*** hoặc ***NoneType***.
- Cần chặn tương tác hoặc hiện ***MessageBox*** cảnh báo (trong GUI) hoặc thông báo trong console.

e) *Giao diện người dùng (nếu dùng GUI)*

- Cần bố trí hợp lý các nút, Label, Entry để thao tác trực quan.
- Đảm bảo nội dung Label cập nhật real-time sau mỗi hành động.
- Tránh việc người dùng nhấn nhiều lần gây sai trạng thái.

f) *Tính mở rộng*

- Việc thêm nhiều Critter (Critter Farm) cần xây dựng danh sách quản lý (list of objects).
- Mỗi Critter có thể cần giao diện riêng hoặc một menu chọn thú cần tương tác.

2.4. Các kiến thức sử dụng

- Để hoàn thành đề tài, cần vận dụng các kiến thức sau:

- **Ngôn ngữ lập trình Python:** Vận dụng cú pháp cơ bản của Python để xây dựng lớp `Critter`, thao tác với chuỗi, danh sách và điều kiện logic. Áp dụng kiến thức về hàm, đối tượng, và xử lý ngoại lệ (try-except) để đảm bảo tính ổn định của chương trình khi người dùng nhập sai hoặc thao tác chưa hợp lệ.
- **Thư viện Tkinter:** Tạo giao diện đồ họa bằng thư viện Tkinter, sử dụng các widget như `Frame`, `Label`, `Entry`, `Button`, `Listbox` để thiết kế bố cục rõ ràng, thân thiện với người dùng. Tùy biến font chữ, màu sắc và hành vi hiển thị để nâng cao trải nghiệm.
- **Quản lý trạng thái:** Áp dụng mô hình hướng đối tượng để quản lý thông tin và trạng thái của Critter. Sử dụng các phương thức như `feed()`, `play()` và `sleep()` để thay đổi các chỉ số trạng thái như `'hunger'` và `'boredom'`.
- **Xử lý sự kiện:** Gắn các sự kiện tương tác với người dùng như chọn Critter từ danh sách (`ListboxSelect`) hoặc nhấn nút (`Button command`) để phản hồi phù hợp. Tự động cập nhật giao diện sau mỗi hành động.
- **Quản lý môi trường phát triển:** Phát triển và chạy chương trình trong môi trường Visual Studio Code. Biết cách thiết lập môi trường ảo, sử dụng tiện ích mở rộng như Pylance để hỗ trợ gợi ý cú pháp và gỡ lỗi hiệu quả.
- **Kỹ năng gỡ lỗi:** Kiểm tra logic xử lý trạng thái Critter và giao diện khi thực hiện các thao tác liên tục. Xử lý các tình huống như không nhập tên Critter, chưa chọn Critter, giúp đảm bảo chương trình không bị lỗi trong quá trình chạy.

CHƯƠNG II. CƠ SỞ LÝ THUYẾT

2.1. Ngôn ngữ lập trình sử dụng

- Chương trình được xây dựng bằng **Python 3**, một ngôn ngữ bậc cao, dễ học, cú pháp gọn gàng và hỗ trợ tốt cho lập trình hướng đối tượng cũng như lập trình giao diện người dùng (GUI). Python rất phù hợp với mô hình mô phỏng và tương tác như chương trình Critter Caretaker.

2.2. Các kiến thức sử dụng trong chương trình

2.2.1. Kiểu dữ liệu List

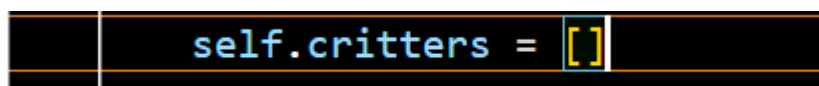
- **Kiểu dữ liệu List trong Python** là một collection lưu trữ các phần tử theo thứ tự đã cho, có thể thay đổi. Cho phép chứa dữ liệu trùng lặp. List có cấu trúc dữ liệu mà có khả năng lưu giữ các kiểu dữ liệu khác nhau.

- **List trong Python** là cấu trúc dữ liệu mà có khả năng lưu giữ các kiểu dữ liệu khác nhau.

- **List trong Python** là thay đổi (mutable), nghĩa là Python sẽ không tạo một List mới nếu bạn sửa đổi một phần tử trong List.

- **List** là một container mà giữ các đối tượng khác nhau trong một thứ tự đã cho. Các hoạt động khác nhau như chèn hoặc xóa có thể được thực hiện trên List.

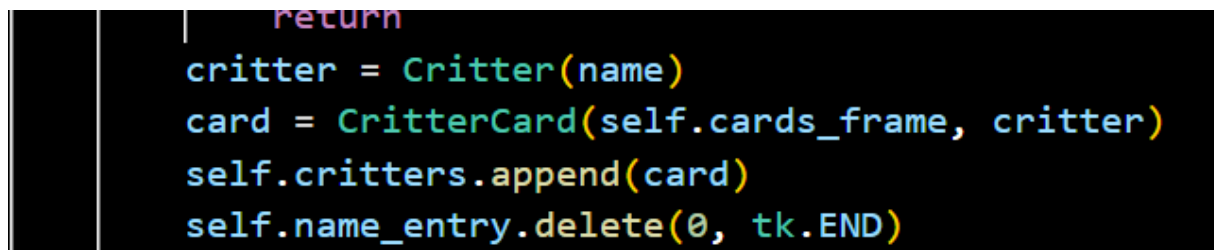
- **List trong Python** được viết với dấu ngoặc vuông [].

A screenshot of a code editor showing the line `self.critters = []`. The text is in a light blue font on a dark background. The opening square bracket is highlighted with a yellow box.

Hình 4. Cấu trúc kiểu dữ liệu list sử dụng trong chương trình

- Câu lệnh trên được sử dụng để tạo 1 list rỗng để chứa các critterscard

- Sau đó để thêm các critterscard vào List ta sử dụng cấu trúc lệnh sau:

A screenshot of a code editor showing a block of Python code. The code is as follows:

```
return  
critter = Critter(name)  
card = CritterCard(self.cards_frame, critter)  
self.critters.append(card)  
self.name_entry.delete(0, tk.END)
```

The code is in a light blue font on a dark background. The word 'return' is in pink.

Hình 5. Các lệnh sử dụng liên quan đến kiểu dữ liệu list



Hình 6. Mô phỏng danh sách các critters

2.2.2. Thư viện Tkinter – Tạo giao diện người dùng (GUI)

a) Giới thiệu về Tkinter

- Tkinter là thư viện GUI tiêu chuẩn cho Python. Tkinter trong Python cung cấp một cách nhanh chóng và dễ dàng để tạo các ứng dụng GUI. Tkinter cung cấp giao diện hướng đối tượng cho bộ công cụ Tk GUI.

b) Các bước để tạo 1 ứng dụng Tkinter

1. import Tkinter module.
2. Tạo cửa sổ chính của ứng dụng GUI.
3. Thêm một hoặc nhiều widget nói trên vào ứng dụng GUI.
4. Gọi vòng lặp sự kiện chính để các hành động có thể diễn ra trên màn hình máy tính của người dùng.

- Ví dụ:

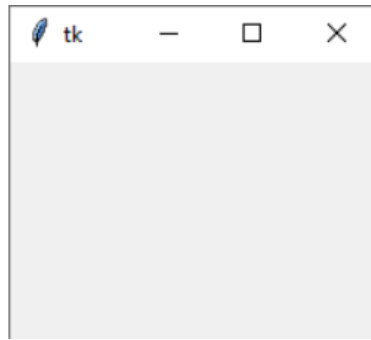
```

1  # import Tkinter module
2  from tkinter import *
3  # Tạo cửa sổ chính của ứng dụng GUI
4  top = Tk()
5  # Gọi vòng lặp sự kiện chính để các hành động có thể diễn ra trên màn hình máy tính
6  top.mainloop()

```

Hình 7. Ví dụ để tạo Tkinter

- Kết quả:



Hình 8. Giao diện Tkinter

c) Các Widget của Tkinter sử dụng trong chương trình

1	Button	Button được sử dụng để thêm nhiều nút khác nhau vào ứng dụng python.
2	Entry	Entry được sử dụng để hiển thị trường văn bản một dòng cho người dùng. Nó thường được sử dụng để nhập các giá trị của người dùng.
3	Frame	Frame có thể được định nghĩa là một vùng chứa mà có thể chứa một hoặc nhiều widget khác.
4	Label	Label là một văn bản được sử dụng để hiển thị một số thông báo hoặc thông tin cho các widget khác.
5	ListBox	ListBox được sử dụng để hiển thị danh sách các tùy chọn cho người dùng.
6	MessageBox	Nó được sử dụng để hiển thị hộp thông báo trong các ứng dụng desktop.

Bảng 1. Danh mục Widget sử dụng

d) Phương thức pack() trong Python Tkinter

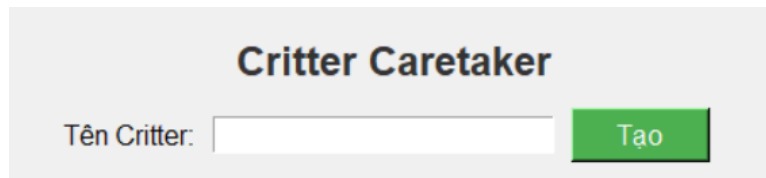
- Phương thức pack() được sử dụng để tổ chức widget theo khối. Vị trí các widget được thêm vào ứng dụng python bằng phương thức pack() có thể được kiểm soát bằng cách sử dụng các tùy chọn khác nhau được chỉ định trong lệnh gọi phương thức.

- Cú pháp:

```
1 | widget.pack(options)
```

Hình 9. Cú pháp Pack()

- Kết quả:

A screenshot of a Tkinter window titled "Critter Caretaker". The window has a light gray background. At the top, the title "Critter Caretaker" is displayed in a bold, black font. Below the title, there is a label "Tên Critter:" followed by a white text input field with a thin gray border. To the right of the input field is a green button with the text "Tạo" in white.

Hình 10. Ví dụ về việc sử dụng Pack()

CHƯƠNG III. THIẾT KẾ VÀ XÂY DỰNG CHƯƠNG TRÌNH

3.1. Sơ đồ khối hệ thống

3.1.1. mô tả các module chính trong chương trình

- Chương trình được chia thành hai phần chính tương ứng với hai lớp và phần khởi tạo giao diện:

a) Lớp Critter – Xử lý logic đối tượng

- Đây là module đại diện cho từng sinh vật (Critter) trong trò chơi. Lớp này định nghĩa các thuộc tính và hành vi cơ bản của Critter:

❖ Thuộc tính:

- name: Tên của Critter.
- hunger: Mức độ đói (số nguyên).
- boredom: Mức độ chán (số nguyên)

❖ Phương thức:

- __str__(): Trả về chuỗi mô tả trạng thái hiện tại của Critter.
- feed(): Giảm mức đói.
- play(): Giảm chán, tăng đói.
- sleep(): Tăng đói và chán (mô phỏng nghỉ ngơi).
- get_mood(): Trả về trạng thái cảm xúc hiện tại dựa vào độ đói và chán.

b) Lớp CritterApp – Xử lý giao diện và tương tác

- Đây là phần xử lý giao diện người dùng (GUI) và các thao tác tương tác với Critter. Lớp này quản lý toàn bộ cửa sổ chính, các nút bấm, danh sách Critter, và vùng hiển thị trạng thái.

- Chức năng chính:

- Tạo giao diện bằng Tkinter (Label, Entry, Button, Listbox, Frame).
- Lưu trữ danh sách các Critter đã tạo.
- Gắn sự kiện với các nút để cho ăn, chơi, ngủ.
- Hiển thị tâm trạng và trạng thái của Critter được chọn.

c) Hàm main (if __name__ == "__main__":) – Khởi động ứng dụng

- Tạo đối tượng Tk() làm cửa sổ chính.
- Tạo một đối tượng CritterApp truyền vào Tk().
- Gọi mainloop() để bắt đầu vòng lặp giao diện.

d) Mối quan hệ giữa các Module

- Hai lớp Critter và CritterApp tương tác chặt chẽ với nhau:

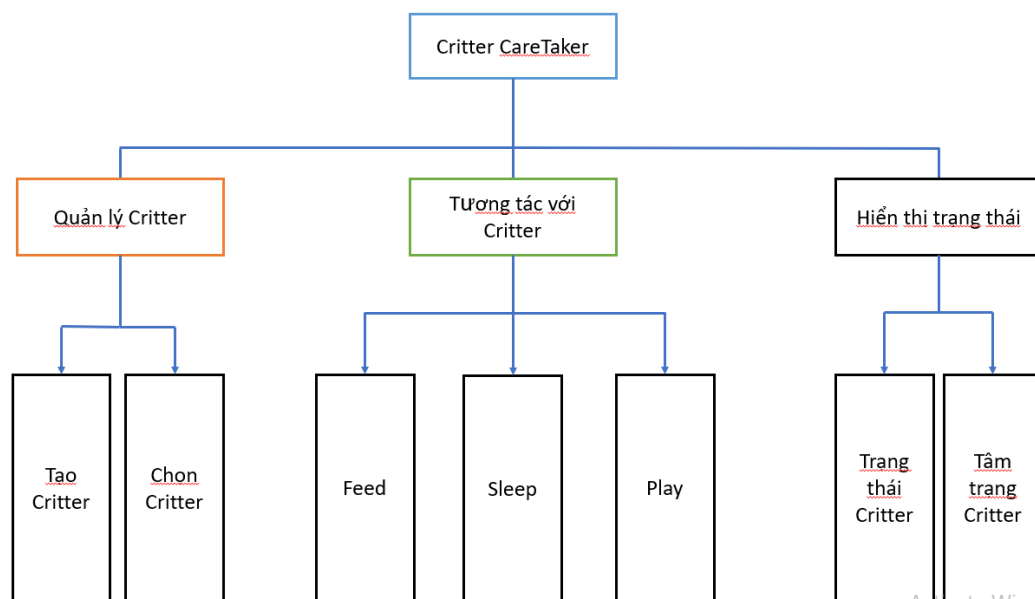
- CritterApp là lớp điều khiển giao diện, trực tiếp tạo và quản lý nhiều đối tượng Critter.
- Mỗi khi người dùng tương tác (ví dụ cho ăn, chơi...), CritterApp gọi đến các phương thức tương ứng trong lớp Critter, từ đó cập nhật trạng thái và hiển thị lại lên giao diện.

e) Ưu điểm của cấu trúc phân chia Module

- Tách biệt logic và giao diện: Việc chia ra hai lớp giúp mã nguồn dễ đọc, dễ bảo trì.
- Dễ mở rộng: Có thể bổ sung thêm thuộc tính cho Critter (ví dụ: năng lượng, tuổi...) mà không ảnh hưởng đến giao diện.
- Tái sử dụng được: Lớp Critter có thể sử dụng lại trong các chương trình khác như mô phỏng, game nhỏ...

3.1.2. Biểu đồ phân cấp chức năng

- Sơ đồ phân cấp chức năng của chương trình Critter Caretaker mô tả cách các chức năng được chia thành từng nhóm rõ ràng, từ chức năng tổng thể đến các chức năng con cụ thể. Đây là phương pháp giúp phân tích hệ thống theo hướng từ tổng quan đến chi tiết, đảm bảo dễ phát triển, mở rộng và bảo trì.
- Biểu đồ phân cấp chức năng mô tả các chức năng chính của chương trình, được trình bày như sau:



Hình 11. Biểu đồ phân cấp chức năng

- Giải thích:

a) Chức năng tổng thể: Critter Caretaker

- Đây là chức năng chính của toàn bộ ứng dụng, đóng vai trò là điểm khởi đầu cho mọi hoạt động.
- Từ đây, chương trình chia thành ba nhóm chức năng chính: **Quản lý Critter**, **Tương tác với Critter**, và **Hiển thị trạng thái**.

b) Nhóm chức năng: Quản lý Critter

- Tạo Critter: Cho phép người dùng nhập tên để tạo một đối tượng Critter mới. Tên được hiển thị trong danh sách và lưu trong bộ nhớ.
- Chọn Critter: Khi có nhiều Critter, người dùng có thể chọn một cá thể cụ thể để tương tác. Việc chọn được thực hiện thông qua danh sách (Listbox).

c) Nhóm chức năng: Tương tác với Critter

- Cho ăn: Giảm mức độ đói của Critter. Nếu không được cho ăn thường xuyên, Critter sẽ trở nên mệt mỏi.
- Chơi: Giảm sự buồn chán của Critter nhưng làm tăng cảm giác đói.

- Ngủ: Tăng cả đói và chán, nhưng mang tính chất phục hồi (có thể mở rộng trong tương lai).

- Các hành động này làm thay đổi trạng thái nội tại của Critter (hunger, boredom), từ đó ảnh hưởng đến tâm trạng của nó.

d) Nhóm chức năng: Hiển thị trạng thái

- Trạng thái Critter: Hiển thị thông tin chi tiết như tên, mức độ đói và chán của Critter đang được chọn.
- Tâm trạng: Hiển thị tâm trạng tổng quát dựa trên trạng thái bên trong của Critter (vui vẻ, tức giận, mệt mỏi), giúp người dùng nhận biết cần chăm sóc như thế nào.

3.2. Sơ đồ khối các thuật toán chính

- Các thuật toán chính trong chương trình bao gồm:

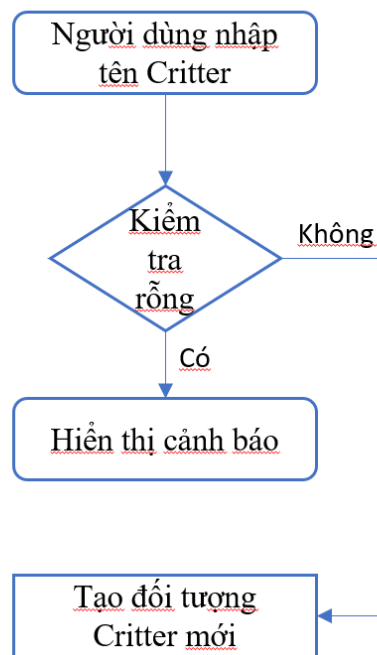
3.2.1. Thuật toán: Tạo Critter

- Mục đích: Tạo đối tượng Critter mới từ tên do người dùng nhập.

- Bước thực hiện:

- Người dùng nhập tên vào ô **Entry**.
- Kiểm tra tên có bị để trống không.
- Nếu trống → hiển thị cảnh báo.
- Nếu hợp lệ → tạo đối tượng Critter mới, thêm vào danh sách critters, và hiển thị trên **Listbox**.

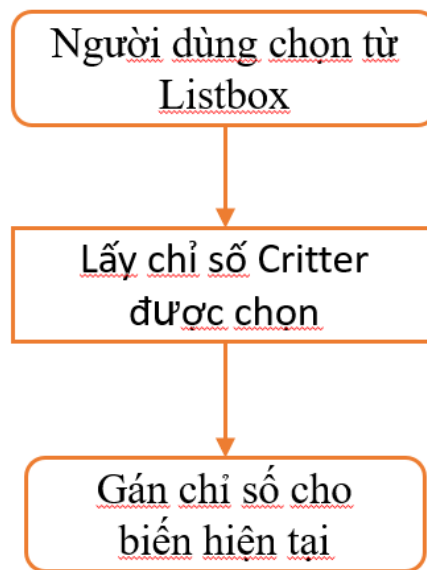
- Ý nghĩa: Đảm bảo không tạo ra Critter không tên và duy trì danh sách quản lý.



Hình 12. Sơ đồ thuật toán tạo Critter

3.2.2. Thuật toán: Chọn Critter

- Mục đích: Ghi nhận Critter mà người dùng chọn để tương tác.
- Bước thực hiện:
 - Khi người dùng nhấp vào một dòng trong Listbox, hệ thống lấy chỉ số dòng đó.
 - Lưu chỉ số này vào biến ***current_index*** để biết người dùng đang làm việc với Critter nào.
- Ý nghĩa: Đảm bảo các hành động như cho ăn, chơi, ngủ được thực hiện đúng đối tượng.



Hình 13. Sơ đồ thuật toán chọn Critter

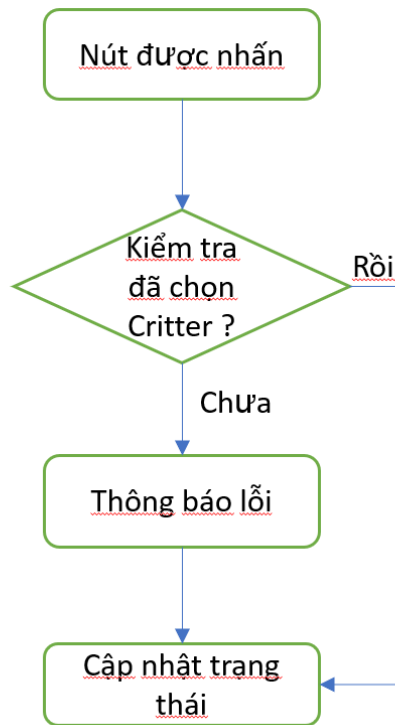
3.2.3. Thuật toán: Cho Feed/Play/Sleep

- Mục đích: Cập nhật trạng thái Critter khi người dùng nhấn nút tương ứng.

- Bước thực hiện:

- Kiểm tra xem người dùng đã chọn Critter chưa (`current_index` khác `None`).
- Nếu chưa chọn → hiện thông báo lỗi.
- Nếu đã chọn:
- Gọi phương thức tương ứng (*`feed()`*, *`play()`*, *`sleep()`*) của Critter đang chọn.
- Cập nhật trạng thái trên giao diện bằng *`display_status()`*.

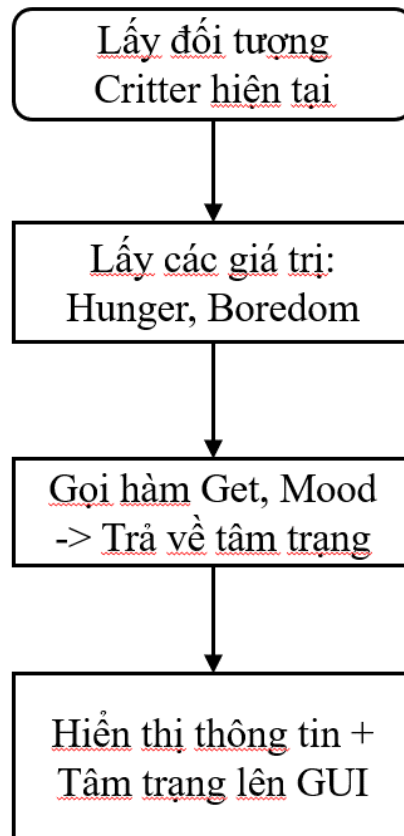
- Ý nghĩa: Cập nhật trạng thái sinh học và tâm trạng của Critter đúng theo thao tác.



Hình 14. Sơ đồ thuật toán ăn/choi/ngủ

3.2.4. Thuật toán: Hiển thị trạng thái Critter

- Mục đích: Hiển thị thông tin chi tiết về Critter được chọn lên GUI.
- Bước thực hiện:
 - Lấy đối tượng Critter theo chỉ số `current_index`.
 - Truy xuất thuộc tính `hunger`, `boredom`, gọi hàm `get_mood()` để lấy tâm trạng.
 - Hiển thị đầy đủ thông tin lên hai nhãn (`status_label` và `mood_label`).
- Ý nghĩa: Cung cấp thông tin trực quan, giúp người dùng theo dõi tình trạng Critter.



Hình 15. Sơ đồ thuật toán hiển thị trạng thái Critter

3.3. Cấu trúc dữ liệu

3.3.1. Bảng Critter

Tên trường	Kiểu dữ liệu
name	String
hunger	Integer
boredom	Integer

Bảng 2. Cấu trúc dữ liệu của Critter

3.3.2. Danh sách Critter

- Tên biến: self.critters
- Kiểu: List
- Mỗi phần tử: một đối tượng Critter có đầy đủ thông tin như trên.
- Vai trò: lưu tất cả các Critter được tạo bởi người dùng trong phiên làm việc hiện tại.

3.4. Chương trình

3.4.1. Các hàm trong chương trình chính

a) Hàm khởi tạo lớp Critter

```
class Critter:
    def __init__(self, name):
        self.name = name
        self.hunger = 0
        self.boredom = 0
```

Hình 16. Cấu trúc hàm tạo lớp Critter

- Dùng để khởi tạo một đối tượng Critter với tên, mức đói (**hunger**) và mức chán (**boredom**) ban đầu bằng 0

b) Hàm hành động của Critter

```
def feed(self): ...
def play(self): ...
def sleep(self): ...
```

Hình 17. Cấu trúc hàm hành động của Critter

- feed(): Giảm hunger đi 1 đơn vị (không giảm dưới 0).
- play(): Tăng hunger lên 2, giảm boredom đi 1 (không giảm dưới 0).
- sleep(): Tăng cả hunger và boredom thêm 1.

c) Hàm xác định tâm trạng

```
def get_mood(self):
```

Hình 18. Cấu trúc hàm xác định tâm trạng Critter

- Trả về tâm trạng hiện tại của Critter dựa trên chỉ số **hunger** và **boredom**

d) Hàm tạo Critter

```
def create_critter(self):
```

Hình 19. Cấu trúc hàm tạo Critter

- Lấy tên người dùng nhập, tạo đối tượng **Critter**, thêm vào danh sách **self.critters**, và cập nhật **Listbox**.

e) Hàm chọn Critter trong danh sách

```
def select_critter(self, event):
```

Hình 20. Cấu trúc hàm chọn Critter trong List

- Khi người dùng chọn một **Critter** trong **Listbox**, hàm lấy chỉ số và cập nhật giao diện hiển thị thông tin.

f) Hàm hiển thị trạng thái

```
def display_status(self):
```

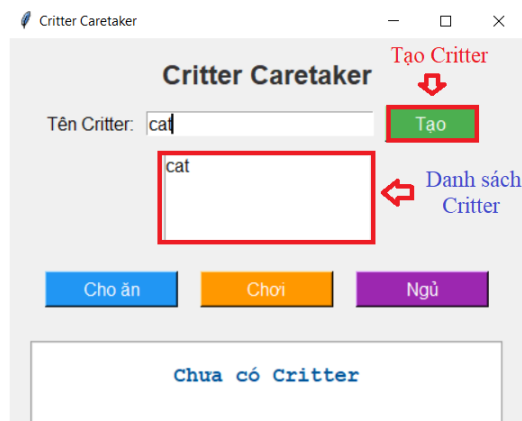
Hình 21. Cấu trúc hàm hiển thị trạng thái Critter

- Cập nhật thông tin Critter đang chọn gồm: tên, mức đói, chán, và tâm trạng lên Label giao diện.

CHƯƠNG IV: THỰC NGHIỆM VÀ KẾT LUẬN

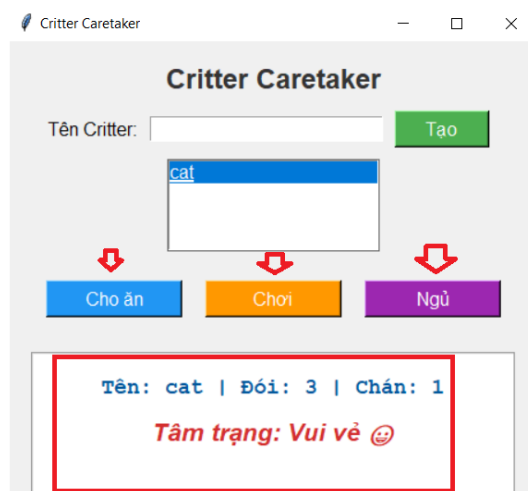
4.1. Thực nghiệm

4.1.1. Chức năng tạo Critter và Lưu vào danh sách



Hình 22. Giao diện tạo Critter và List

4.1.2. Chức năng hành động cho từng Critter và Tâm trạng của Critter



Hình 23. Giao diện hiển thị tâm trạng và các hành động với Critter

```
def get_mood(self):
    if self.hunger > 5 or self.boredom > 5:
        return "Kiệt sức 😫"
    elif self.hunger > self.boredom:
        return "Mệt mỏi 😩"
    elif self.boredom > self.hunger:
        return "Tức giận 😡"
    else:
        return "Vui vẻ 😊"
```

Hình 24. Điều kiện của từng trạng thái

4.2. Kết luận

4.2.1. Sản phẩm đã làm đã làm được những gì?

- Xây dựng thành công ứng dụng Critter Caretaker bằng Python và thư viện Tkinter.
- Giao diện trực quan, dễ sử dụng, cho phép:
 - Tạo nhiều Critter khác nhau.
 - Cho Critter ăn, chơi, ngủ và cập nhật trạng thái tương ứng.
 - Hiện thị trạng thái và tâm trạng của Critter theo thời gian thực.
 - Quản lý dữ liệu Critter thông qua danh sách và sự kiện chọn.

4.2.2. Những điều học được

- Hiểu và áp dụng kiến thức về lập trình hướng đối tượng trong Python.
- Làm quen với thư viện Tkinter, thiết kế và tổ chức giao diện người dùng (GUI).
- Biết cách sử dụng các widget cơ bản như Entry, Button, Listbox, Label, Frame.
- Nắm được cách gắn sự kiện, xử lý hành vi người dùng và cập nhật giao diện động.
- Rèn luyện tư duy tổ chức mã, phân chia chức năng rõ ràng giữa logic và giao diện.

4.2.3. Hướng cải tiến trong tương lai

- Bổ sung hình ảnh 2D cho Critter: mỗi Critter sẽ có hình minh họa và biểu cảm rõ ràng theo tâm trạng (vui, giận, buồn) thay vì chỉ hiển thị icon cảm xúc đơn giản.
- Thêm hiệu ứng âm thanh khi người dùng thực hiện các hành động như cho ăn, chơi, ngủ để tạo trải nghiệm chân thực và sinh động hơn.
- Thiết kế Critter hoạt hình 2D thay vì chỉ dùng văn bản: biểu cảm khuôn mặt, động tác chuyển động cơ bản khi thay đổi trạng thái.
- Lưu trữ dữ liệu Critter vào file (JSON/cơ sở dữ liệu) để giữ lại tiến trình người chơi giữa các phiên sử dụng.
- Thêm các yếu tố nâng cao: chỉ số sức khỏe, năng lượng, cấp độ phát triển (level), hoặc hệ thống điểm thưởng.
- Xây dựng một thư viện Critter đa dạng với hình dáng, màu sắc khác nhau để người dùng có thể lựa chọn.

TÀI LIỆU THAM KHẢO

1. Cuốn sách ”*Python Programming for the Absolute Beginner- 3rd Edition*”
2. Cuốn sách: “*Đường vào lập trình Python*“ của tác giả “TS. Nguyễn Ngọc Giang”.
3. <https://codelearn.io>
4. <https://www.codecademy.com/>
5. https://www.youtube.com/watch?v=NZj6LI5a9vc&list=PL33lvabfs1xczCv2BA0SaNJHu_VXsFtg