



Manual Tecnico

Lenguajes Formales y De Programacion
Práctica 1

Josué Salvador Sánchez Portomarin
201800968

INDICE

INTRODUCCIÓN	ARCHIVOS DEL PROYECTOR	CÓDIGO FUENTE	CONCLUSIONES
--------------	------------------------------	------------------	--------------

INTRODUCCIÓN

Esta práctica tiene como objetivo central la implementación de soluciones lógicas por medio del lenguaje de programación Python, que permitan una gestión más eficaz y ágil del manejo de vida, alimentación y recreación de una mascota.

El sistema permitirá cargar archivos con la información pertinente a cada una de las mascotas

ARCHIVOS DEL PROYECTO

La extensión de los archivos .py le indica al sistema operativo que esta trabajando en el lenguaje Python. A continuación describiremos la función de los archivos utilizados durante la practica.

Este archivo controla el flujo principal del proyecto, desde este se debe ejecutar el proyecto para inicializarlo.

Este archivo permite almacenar y manipular a través de sus funciones toda información cargada por medio de los archivos de prueba.

Estos archivos permiten controlar toda la información de las bodegas dentro de nuestra base de datos.



main.py

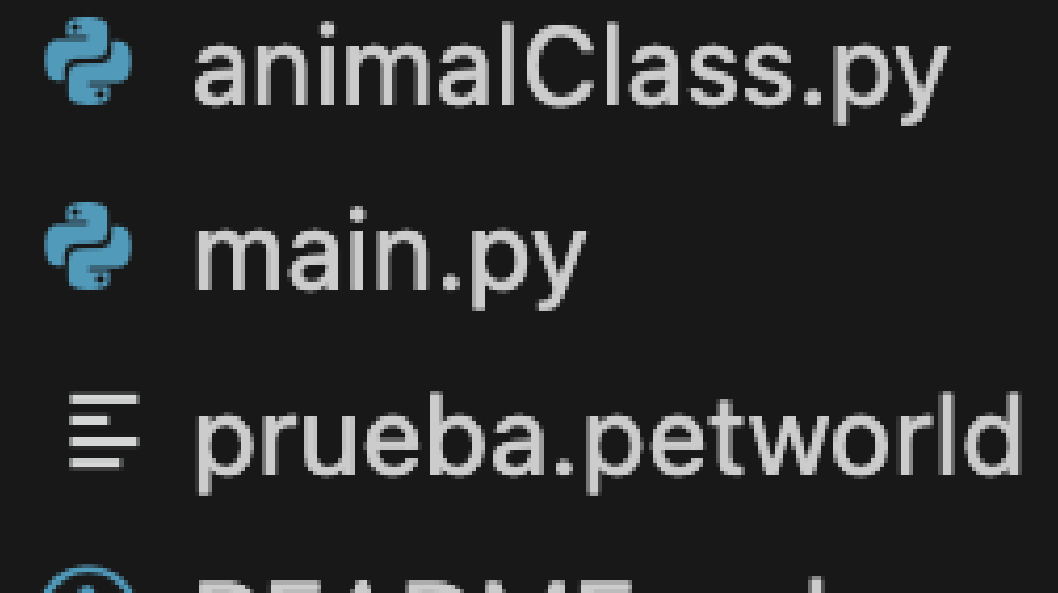


classProducts.py







Test Files

Imagen de los archivos
utilizados para la realización del
proyecto



A screenshot of a file explorer window with a dark background. It displays a list of files and folders. The first two items are Python files, each preceded by a blue Python logo icon: 'animalClass.py' and 'main.py'. The third item is a folder named 'prueba.petworld', preceded by a grey icon of three horizontal lines. A fourth item is partially visible at the bottom, preceded by a blue circular icon.

-  animalClass.py
-  main.py
-  prueba.petworld
- 

Código Fuente

```
from os import system
from datetime import date
from datetime import datetime

#import all my classes
from animalClass import *
```

```
> def viewList(): ...

> def chargeFile(title): ...

> def writeFileLines(parraf): ...

> def writeFile(text): ...

> def createAnimal(name): ...

> def feedAnimal(name, plus): ...

> def playWithAnimal(name, plus): ...

> def searchAnimal(name): ...

#--> VIEW USER MENUS
#view student data
> def studentData(): ...

#create principal menu
> def principalMenu(): You, ay
```

```
#--> VIEW USER MENUS
#view student data
> def studentData(): ...

#create principal menu
> def principalMenu(): You, ay

#VARS
animal=animalList()

studentData()
```

ARCHIVO MAIN.PY

Importacion de la clases y metodos nativos de python

```
from os import system
from datetime import date
from datetime import datetime

#import all my classes
from animalClass import *
```



Esta línea de código permite reutilizar algunas funciones del archivo animalClass dentro del archivo main.py

from indica la clase de donde se tomará la importación.

import le indica a python la acción que debe realizar con la clase.

* el asterisco indica que debe importar todos los metodos y elementos de la clase.

ARCHIVO MAIN.PY

Dentro del archivo mian.py se manejaron las siguientes variables:

```
#VARS  
animal=animalList()  
  
studentData()
```

- animal: Crea la instancia de la clase para manipular a la mascota.
- studentData: Hace visible el primer menu del programa.

ARCHIVO MAIN.PY

Para trabajar con funciones el lenguaje Python utiliza la palabra reservada **def**.

Funciones principales del archivo main.py.

ARCHIVO MAIN.PY

Función para cargar el archivo de entrada "chargeFile()".

Nombre de la función
(parametros)

*Un parámetro es una variable con
información que la función recibe*

Palabra Reservada para
las funciones

```
#Functions
def chargeFile(title):
    print("\n*** ",title," ***\n")
    #save rout at var
    rout=input("Ingresa la dirección del archivo: ")
    #open file
    with open(rout, encoding="utf-8") as file:
        return file.readlines()
```

Solicitud de la ruta del
archivo al usuario

Apertura del archivo de
entrada

ARCHIVO MAIN.PY

Funcion para cargar el archivo .txt.

Nombre de la funcion a llamar, con parametro titulo

a travez de la ruta ingresada por el usuario utiliza un rt para leer el archivo
otros metodos:

- r --> read
- w --> write
- a --> append
- x --> create

```
def chargeFile(title):  
    print("\n*** ",title," ***\n")  
    #save rout at var  
    rout = input("> Ingresa la ruta del archivo: ")  
    file = open(rout,'rt',encoding='utf-8')  
    readAllLines = file.readlines()  
  
    return readAllLines
```

Devuelve la lista de lineas que contiene el archivo

ARCHIVO MAIN.PY

Funcion para generar el menu principal de la aplicación.

Inico de la aplicacion,
llama a la funcion
menuPrincipal (menu
para cargar el archivo)

"print()" nos
permite imprimir
el texto en la
consola

"\n" Realiza un
salto de linea en la
consola

Int(input()) solicita un dato de
entrada al usuario

```
#view student data
def studentData():
    print("\n=====")
    print("==          Practic 1 - Datos del Estudiante          ==")
    print("=====")
    print("==")
    print("==  Lenguajes Foramales y de Programación  ==")
    print("==  Sección: A+                               ==")
    print("==  Nombre:                                   ==")
    print("==      Josué Salvador Sánchez Portomatin    ==")
    print("==  Carné:                                    ==")
    print("==      201800968                             ==")
    print("==")
    print("===== Ingeniria en Ciencisa y Sistemas =====")

#request user option
input("\n> Presiona enter para continuar: ")
system('clear')
principalMenu()
```

ARCHIVO MAIN.PY

If anidados para las opciones del menu, estas le permiten al sistema saber que linea de acción tomar según la petición de un usuario

```
# filter action to realise
if(x[0]=="Crear_Gato"): ...
elif(x[0]=="Dar_de_Comer"): ...
elif(x[0]=="Jugar"): ...
elif(x[0]=="Resumen_Mascota"): ...
elif(x[0]=="Resumen_Global"): ...
else:
    print("** opcion invalida **")
```

ARCHIVO MAIN.PY

OPCIÓN = 1

Crea una mascota y almacena la informacion en la base de datos

```
# filter action to realise
if(x[0]=="Crear_Gato"):
    name= x[1].split("\n")
    #validate cat exist
    exist = searchAnimal(name[0])
    if(exist == None):
        createAnimal(name[0])
    else:
        print(f"El nombre {name[0]} ya existe y no puede ser registrado dos veces")
```

```
elif(x[0]=="Dar_de_Comer"):
    data = x[1].split(",")
    name = data[0]
    peso = data[1].split('\n')

    plus = 12 + float(peso[0])
    feedAnimal(name, plus)
```

Busca una mascota y si esta existe la alimenta

OPCION = 2

ARCHIVO MAIN.PY

OPCIÓN = 3

Busca una mascota y si esta existe juega con ella disminuyendo su energia

```
elif(x[0]=="Jugar"):  
    data = x[1].split(",")  
    name = data[0]  
    time = data[1].split('\n')  
  
    #operate data  
    plus = float(time[0]) * 0.1  
    playWithAnimal(name, plus)
```

```
elif(x[0]=="Resumen_Global"):  
    today = date.today()  
    now = datetime.now()  
    dateNow = f"{today.day}/{today.month}/{today.year}, {now.hour}:{now.minute}"  
    print(f"{dateNow} "+"-".rjust(24, '-')+" Resumen Global "+"-".ljust(24, '-')  
    writeFile(f"{dateNow} "+"-".rjust(24, '-')+" Resumen Global "+"-".ljust(24,  
    print("\n")  
    viewList()  
    print("\n")  
    print("-".rjust(50, '-'))  
    writeFile("-".rjust(70, '-'))  
    animal.draw()
```

Muestra un registro de una mascota existe

OPCION = 4

ARCHIVO MAIN.PY

OPCIÓN = 5

Muestra y guarda el resumen de todas las mascotas de como se encuentra su estado actual.

```
elif(x[0]=="Resumen_Global"):
    today = date.today()
    now = datetime.now()
    dateNow = f"{today.day}/{today.month}/{today.year}, {now.hour}:{now.minute}"
    print (f"{dateNow} "+"-".rjust(24, '-')+f" Resumen Global "+"-".ljust(24, '-')
    writeFile(f"{dateNow} "+"-".rjust(24, '-')+f" Resumen Global "+"-".ljust(24, '-')
    print("\n")
    viewList()
    print("\n")
    print("-".rjust(50, '-'))
    writeFile("-".rjust(70, '-'))
    animal.draw()
else:
    print("** opcion invalida **")
```


Intanciamos la clase para manejar los productos como objetos

Instancia de la
clase

```
#save product in the class
def __init__(self, name, amount, price, location):
    self.name = name
    self.amount= amount
    self.price=price
    self.location=location
```

Atributos del
objeto producto

ARCHIVO CLASSPRODUCT.PY

Esta funcion actualiza los productos dentro de la bodega

```
from os import system
import graphviz;
You, hace 24 horas | 1 author (You)
class NodoGato: ...

You, hace 17 minutos | 1 author (You)
class animalList:
    def __init__(self): ...

    def addAnimal(self, dataGato=[]): ...

    #function to feed a cat
    def feed(self, name, plus): ...

    #function to play with a cat
    def play(self, name, plus): ...

    #function to validate cat exist
    def valAnimal(self, name):| You, hace 2

    def draw(self): ...

    def globalList(self): ...

    def printListAnimals(self): ...
```

Administra cada una de las funciones vistas anteriormente

Funcion para agregar una mascota

Recorre la lista, al llegar al ultimo elemnto agrega una mascota

```
def addAnimal(self, dataGato=[]):  
    #sino existe ningun elemento en la lista  
    if self.inicio == None:  
        aux = NodoGato(dataGato)  
        self.inicio = aux  
        self.final = aux  
    #si la lista NO esta vacia hace lo next  
    else:  
        aux = NodoGato(dataGato)  
        self.final.next=aux  
        self.final=aux
```

Funcion para alimentar una mascota

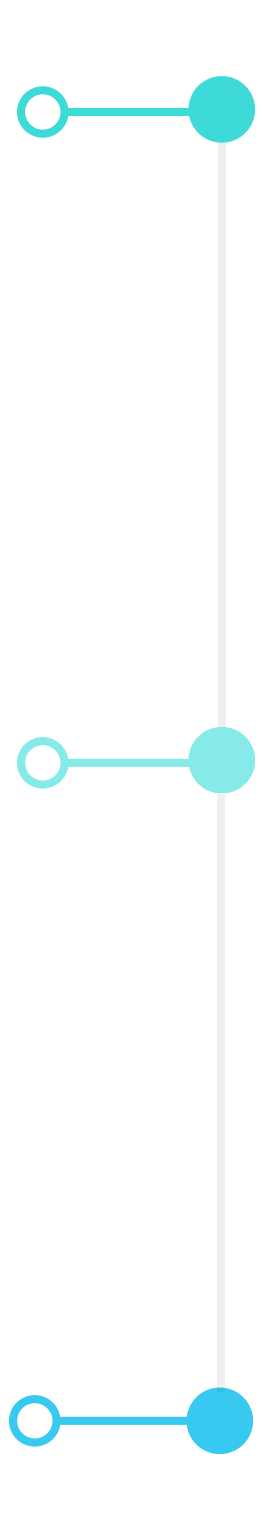
Recorre la lista, busca a la mascota y realiza las operaciones pertinentes

```
#function to feed a cat
def feed(self, name, plus):
    actual = self.inicio
    cont = 1
    while actual!=None:
        if(actual.dataGato[1]==name):
            if(float(actual.dataGato[2])>0):
                actual.dataGato[2] = float(actual.dataGato[2])+plus
                return actual.dataGato
            cont = cont + 1
            actual=actual.next
    return actual
```

Funcion para jugar una mascota

Recorre la lista, busca a la mascota y realiza las operaciones pertinentes

```
#function to play with a cat
def play(self, name, plus):
    actual = self.inicio
    cont = 1
    while actual!=None:
        if(actual.dataGato[1]==name):
            if(float(actual.dataGato[2])>0):
                actual.dataGato[2] = float(actual.dataGato[2])-plus
                return actual.dataGato
            cont = cont + 1
            actual=actual.next
    return actual
```

- 
- A vertical timeline on the left side of the slide. It consists of a light gray vertical line with three horizontal segments extending to the left. Each segment ends with a teal circle. The top circle is a solid teal dot, while the middle and bottom circles are teal outlines.
- Python es un lenguaje de programación versátil que se adapta a una amplia gama de aplicaciones, Su sintaxis simple y legible permite a los programadores escribir código de manera más rápida y eficiente, lo que conduce a una mayor productividad en el desarrollo de proyectos.
 - permite a los programadores aprovechar soluciones preconstruidas y evita tener que crear componentes desde cero, acelerando el proceso de desarrollo y reduciendo posibles errores.
 - La facilidad de aprendizaje y la legibilidad del código también contribuyen a que programadores de diferentes niveles de experiencia puedan adoptar el lenguaje con relativa rapidez y lograr resultados significativos en sus proyectos.