Nombre: miguel angel menodza Espinoza

Genericidad:

Ejercicio1:

class Caja:  
 def \_\_init\_\_(self):  
 self.contenido = None  
 def guardar(self, objeto):  
 self.contenido = objeto  
 def obtener(self):  
 return self.contenido  
# Prueba  
caja\_texto = Caja()  
caja\_texto.guardar("Hola mundo")  
caja\_numero = Caja()  
caja\_numero.guardar(123  
print("CajaTexto:", caja\_texto.obtener())  
print("CajaNumero:", caja\_numero.obtener())

Suedocodigo:

Clase Caja<TIPO>

Atributo contenido

Método guardar(objeto: TIPO)

contenido ← objeto

Método obtener() : TIPO

Retornar contenido

Inicio

cajaTexto ← nueva Caja<String>

cajaTexto.guardar("Hola mundo")

cajaNumero ← nueva Caja<Integer>

cajaNumero.guardar(123)

Imprimir cajaTexto.obtener()

Imprimir cajaNumero.obtener()

Fin

Ejecicio3

class Catalogo:

def \_\_init\_\_(self):

self.elementos = []

def agregar(self, elemento):

self.elementos.append(elemento)

def buscar(self, elemento):

return elemento in self.elementos

# Prueba

libros = Catalogo()

libros.agregar("El Principito")

print("Buscar libro:", libros.buscar("El Principito"))

productos = Catalogo()

productos.agregar("Celular")

print("Buscar producto:", productos.buscar("Celular"))

seudocódigo:

Clase Catalogo<TIPO>

Lista elementos

Método agregar(elemento: TIPO)

elementos.agregar(elemento)

Método buscar(elemento: TIPO) : Booleano

Retornar elemento en elementos

Inicio

libros ← nuevo Catalogo

libros.agregar("El Principito")

Imprimir libros.buscar("El Principito")

productos ← nuevo Catalogo

productos.agregar("Celular")

Imprimir productos.buscar("Celular")

Fin

Ejercicio5:

class Pila:

def \_\_init\_\_(self):

self.elementos = []

def apilar(self, elemento):

self.elementos.append(elemento)

def desapilar(self):

return self.elementos.pop()

def mostrar(self):

print(self.elementos)

# Prueba

pila\_texto = Pila()

pila\_texto.apilar("uno")

pila\_texto.apilar("dos")

pila\_texto.mostrar()

pila\_texto.desapilar()

pila\_texto.mostrar()

pila\_numeros = Pila()

pila\_numeros.apilar(10)

pila\_numeros.apilar(20)

pila\_numeros.mostrar()

seudocódigo:

Clase Pila<TIPO>

Lista elementos

Método apilar(elemento: TIPO)

elementos.agregar(elemento)

Método desapilar() : TIPO

Retornar elementos.quitarUltimo()

Método mostrar()

Imprimir elementos

Inicio

pilaTexto ← nueva Pila

pilaTexto.apilar("uno")

pilaTexto.apilar("dos")

pilaTexto.mostrar()

pilaTexto.desapilar()

pilaTexto.mostrar()

pilaNumeros ← nueva Pila

pilaNumeros.apilar(10)

pilaNumeros.apilar(20)

pilaNumeros.mostrar()

Fin

Persistencias:

Ejercicio1:

class Empleado:

def \_\_init\_\_(self, nombre, edad, salario):

self.nombre = nombre

self.edad = edad

self.salario = salario

class ArchivoEmpleado:

def \_\_init\_\_(self, nomA):

self.nomA = nomA

self.empleados = []

def crearArchivo(self):

self.empleados = []

def guardarEmpleado(self, e):

self.empleados.append(e)

def buscaEmpleado(self, nombre):

for e in self.empleados:

if e.nombre == nombre:

return e

return None

def mayorSalario(self, sueldo):

for e in self.empleados:

if e.salario > sueldo:

return e

return None

seudocodigo:

Clase Empleado:

Atributos:

nombre: String

edad: int

salario: float

Métodos:

Constructor(nombre, edad, salario)

Clase ArchivoEmpleado:

Atributos:

nomA: String

empleados: Lista de Empleado

Métodos:

Constructor(n)

crearArchivo()

guardarEmpleado(e)

buscaEmpleado(n)

mayorSalario(sueldo)

Ejercicio3:

class Cliente:

def \_\_init\_\_(self, id, nombre, telefono):

self.id = id

self.nombre = nombre

self.telefono = telefono

class ArchivoCliente:

def \_\_init\_\_(self, nomA):

self.nomA = nomA

self.clientes = []

def guardarCliente(self, c):

self.clientes.append(c)

def buscarCliente(self, id):

for cli in self.clientes:

if cli.id == id:

return cli

return None

def buscarCelularCliente(self, id):

for cli in self.clientes:

if cli.id == id:

return (cli.nombre, cli.telefono)

return None

seudocodigo:

Clase Cliente:

id: int

nombre: String

telefono: int

Clase ArchivoCliente:

nomA: String

clientes: Lista de Cliente

Métodos:

guardarCliente(c)

buscarCliente(c)

buscarCelularCliente(c)

ejercicio5:

class Medicamento:

def \_\_init\_\_(self, nombre, cod, tipo, precio):

self.nombre = nombre

self.codMedicamento = cod

self.tipo = tipo

self.precio = precio

class Farmacia:

def \_\_init\_\_(self, nombreFarmacia, sucursal, direccion):

self.nombreFarmacia = nombreFarmacia

self.sucursal = sucursal

self.direccion = direccion

self.medicamentos = []

def agregarMedicamento(self, m):

self.medicamentos.append(m)

def mostrarMedicamentos(self, tipo):

return [m for m in self.medicamentos if m.tipo == tipo]

def buscaMedicamento(self, nombre):

for m in self.medicamentos:

if m.nombre == nombre:

return m

return None

class ArchFarmacia:

def \_\_init\_\_(self, na):

self.na = na

self.farmacias = []

def agregarFarmacia(self, f):

self.farmacias.append(f)

def mostrarMedicamentosTosSucursal(self, x):

for f in self.farmacias:

if f.sucursal == x:

return f.mostrarMedicamentos("tos")

def mostrarSucursalGolpex(self):

result = []

for f in self.farmacias:

for m in f.medicamentos:

if m.nombre == "Golpex":

result.append((f.sucursal, f.direccion))

return result

seudocodigo:

Clase Medicamento:

nombre, codMedicamento, tipo, precio

Clase Farmacia:

nombreFarmacia, sucursal, direccion, medicamentos[100]

Clase ArchFarmacia:

na: String

Lista de Farmacia

Métodos:

listar()

mostrarMedicamentosResfrios()

precioMedicamentoTos()

mostrarMedicamentosMenorTos()