Inicializador

from data\_base\_manager.data\_base\_manager import DataBaseManager

from menu\_manager.menu\_manager import CoreApp

if \_\_name\_\_ == "\_\_main\_\_":                      #Verificação se este fecheiro foi corrido diretamento no PYTHON ( \_name\_ variavel de interpretador)

    store\_db = DataBaseManager('base\_de\_dados/dados.json')

    store\_db.import\_dados()

    app = CoreApp(store\_db)

    app.run()

Menu Manager

import os

from enum import Enum

from data\_base\_manager.data\_base\_manager import DataBaseManager

from data\_base\_orcamentos.data\_base\_orcamentos import DataBaseOrcamentos

class MenuCodes(Enum):

    EXIT = 0

    MAIN\_MENU = 1

    DATA\_BASE\_MENU = 2

    ORCAMENTO\_MENU = 6

    ADD\_ORCAMENTO = 7

    EDIT\_ORCAMENTO = 8

class CoreApp:

    def \_\_init\_\_(self, store\_db):

        self.active\_menu = MenuCodes.MAIN\_MENU

        self.store\_db = store\_db

        self.orcamento\_ativo = None

    def run(self):

        while self.active\_menu != MenuCodes.EXIT:

            self.next\_menu()

    def delete\_from\_base\_dados(self):                                   #Eliminar elementos da base de dados

        del\_db\_code = input('Introduza o código do elemento a eliminar ou introduza 0 para sair: ')

        if del\_db\_code == '0':

            print('Cancelou a sua operação!')

            return

        for path, \_, file\_names in os.walk('orcamentos'):

            for file in file\_names:

                o = DataBaseOrcamentos(path+'/'+file)

                o.import\_dados()

                if o.check\_code(del\_db\_code):

                    print(f'Este artigo encontra-se ativo no orçamento {file}! Impossível eliminar.')

                    return

        self.store\_db.remove(del\_db\_code)

    def add\_to\_base\_dados(self):                                        #Inserir elementos da base de dados

        insert\_db\_code = input('\nIntroduza o código do artigo desejado:  ')

        if not insert\_db\_code =='0':

            insert\_db\_name = input('Introduza o nome do artigo: ')

            try:

                insert\_db\_price = float(input('Introduza o preço do artigo: '))

            except ValueError:

                print('O preço inserido não é válido!!')

                return

            result = self.store\_db.insert(insert\_db\_code, insert\_db\_name, insert\_db\_price)

            if result:

                print('\nO seu artigo foi adicionada com sucesso.')

                print(insert\_db\_code)

                print(insert\_db\_name)

                print(insert\_db\_price)

        else:

            print('Cancelou a sua operação!')

            return

    def add\_to\_orcamento(self):

        insert\_orcamento\_code = input('\nIntroduza o código do artigo desejado ou insira 0 para sair:  ')

        if not insert\_orcamento\_code =='0':

            insert\_orcamento\_quantity =int(input('Introduza a quantidade desejada: '))

            self.orcamento\_ativo.insert(insert\_orcamento\_code, insert\_orcamento\_quantity, self.store\_db)

            print('Artigo inserido com sucesso.')

        else:

            print('Cancelou a sua operação!')

            return

    def delete\_from\_orcamento(self):

        del\_orcamento\_code = input('Introduza o código do elemento a eliminar ou introduza 0 para sair: ')

        if not del\_orcamento\_code == '0':

            self.orcamento\_ativo.remove(del\_orcamento\_code)

        else:

            print('Cancelou a sua operação!')

            return

    def delete\_orcamento(self):

        name\_file\_to\_remove = 'orcamentos\\' + input('Insira o nome do orçamento ou insira 0 para voltar: ')

        if name\_file\_to\_remove == 0:

            return

        if not os.path.isfile(name\_file\_to\_remove):

            print('Não foi possivel encontrar o orçamento, por favor vericar os orçamentos existentes.')

            return

        orcamento\_delete = DataBaseOrcamentos(name\_file\_to\_remove)

        orcamento\_delete.import\_dados()

        self.active\_menu = MenuCodes.ORCAMENTO\_MENU

        orcamento\_delete.calcutate\_totals()

        print(orcamento\_delete)

        print('\nDeseja mesmo eliminar? ')

        print('                1 - Sim.')

        print('Qualquer caracter - Não.')

        confirm\_del = input('')

        if confirm\_del == '1':

            print('Orçamento eliminado com sucesso.')

            os.remove(name\_file\_to\_remove)

    def next\_menu(self):                                                #Menu

        if self.active\_menu == MenuCodes.MAIN\_MENU:

            self.print\_main\_menu()

            user\_input = input()

            if user\_input == '0':

                self.active\_menu = MenuCodes.EXIT

                return

            if user\_input == '1': # password?

                self.active\_menu = MenuCodes.DATA\_BASE\_MENU

                return

            if user\_input == '2':

                self.active\_menu = MenuCodes.ORCAMENTO\_MENU

                return

            print('Opção desconhecida!')

            return

        if self.active\_menu == MenuCodes.DATA\_BASE\_MENU:                    #Gestão de artigos

            self.print\_menu\_base\_dados()

            user\_input = input()

            if user\_input == '0':

                self.active\_menu = MenuCodes.MAIN\_MENU

                return

            if user\_input == '1':

                self.add\_to\_base\_dados()

                return

            if user\_input == '2':

                self.delete\_from\_base\_dados()

                return

            if user\_input == '3':

                print(self.store\_db)

                return

            if user\_input == '4':

                self.store\_db.save()

                return

            print('Opção desconhecida!')

            return

        if self.active\_menu == MenuCodes.ORCAMENTO\_MENU:

            self.print\_main\_menu\_orcamento()

            user\_input = input()

            if user\_input == '0':

                self.active\_menu = MenuCodes.MAIN\_MENU

                return

            if user\_input == '1':

                self.active\_menu = MenuCodes.ADD\_ORCAMENTO

                self.orcamento\_ativo = DataBaseOrcamentos()

                return

            if user\_input == '2':

                name\_file = 'orcamentos\\' + input('Insira o nome do orçamento ou insira 0 para voltar: ')

                if name\_file == 0:

                    return

                if not os.path.isfile(name\_file):

                    print('Não foi possivel encontrar o orçamento, por favor vericar os orçamentos existentes.')

                    return

                self.orcamento\_ativo = DataBaseOrcamentos(name\_file)

                self.orcamento\_ativo.import\_dados()

                self.active\_menu = MenuCodes.EDIT\_ORCAMENTO

                return

            if user\_input == '3':

                for path, \_, file\_names in os.walk('orcamentos'):

                    for file in file\_names:

                        o = DataBaseOrcamentos(path+'/'+file)

                        o.import\_dados()

                        o.calcutate\_totals()

                        print(file)

                return

            if user\_input == '4':

                pass

                return

            if user\_input == '5':

                self.delete\_orcamento()

                return

            print('Opção desconhecida!')

            return

        if self.active\_menu == MenuCodes.ADD\_ORCAMENTO:                                 #Criar novo orçamento - finish

            self.print\_menu\_add\_orcamento()

            user\_input = input()

            if user\_input == '0':

                if self.save():

                    self.orcamento\_ativo.save()

                self.active\_menu = MenuCodes.ORCAMENTO\_MENU

                self.orcamento\_ativo = None

                return

            if user\_input == '1':

                self.add\_to\_orcamento()

                return

            if user\_input == '2':

                self.delete\_from\_orcamento()

                return

            if user\_input == '3':

                self.orcamento\_ativo.calcutate\_totals()

                print(self.orcamento\_ativo)

                return

            if user\_input == '4':

                print(self.store\_db)

                return

            if user\_input == '5':

                if self.save():

                    self.orcamento\_ativo.save()

                return

            print('Opção desconhecida!')

            return

        if self.active\_menu == MenuCodes.EDIT\_ORCAMENTO:                        #Editar orçamento do ficheiro

            self.print\_menu\_edit\_orcamento()

            user\_input = input()

            if user\_input == '0':

                if self.save():

                    self.orcamento\_ativo.save()

                self.active\_menu = MenuCodes.ORCAMENTO\_MENU

                self.orcamento\_ativo = None

                return

            if user\_input == '1':

                self.add\_to\_orcamento()

                return

            if user\_input == '2':

                self.delete\_from\_orcamento()

                return

            if user\_input == '3':

                self.orcamento\_ativo.calcutate\_totals()

                print(self.orcamento\_ativo)

                return

            if user\_input == '4':

                pass

                return

            if user\_input == '5':

                if self.save():

                    self.orcamento\_ativo.save()

                return

            print('Opção desconhecida!')

    def print\_main\_menu(self):                                          #Menu principal

        print('\n--MENU PRINCIPAL--')

        print('1 - Gestão de artigos.')

        print('2 - Gestão de orçamentos.')

        print('0 - Sair.\n')

    def print\_menu\_base\_dados(self):                                    #Menu data base

        print('\n--GESTÃO DE ARTIGOS--')

        print("1 - Inserir novo artigo.")

        print('2 - Eliminar artigo.')

        print('3 - Ver Base de dados.')

        print("4 - Guardar.")

        print('0 - Sair.\n')

    def print\_main\_menu\_orcamento(self):                                #Menu orcamento

        print('\n--GESTÃO DE ORÇAMENTOS--')

        print('1 - Criar novo orçamento')

        print('2 - Editar orçamento')

        print('3 - Ver orçamentos existentes')

        print('4 - Ver orçamento especifico')

        print('5 - Eliminar orçamento')

        print('0 - Sair\n')

    def print\_menu\_add\_orcamento(self):                                 #Menu Adicionar orcamento

        print('\n--ADICIONAR NOVO ORÇAMENTO--')

        print('1 - Inserir novo artigo no orçamento')

        print('2 - Eliminar artigo do orçamento')

        print('3 - Ver orçamento')

        print('4 - Ver artigos disponíveis')

        print('5 - Guardar orçamento')

        print('0 - Sair\n')

    def print\_menu\_edit\_orcamento(self):                                #Menu Adicionar artigos

        print('\n--EDITAR ORÇAMENTO--')

        print('1 - Inserir novo artigo no orçamento')

        print('2 - Eliminar artigo do orçamento')

        print('3 - Ver orçamento')

        print('4 - Ver artigos disponíveis')

        print('5 - Guardar orçamento')

        print('0 - Sair\n')

    def save(self):                                                   #Menu salvar

        print('Deseja salvar? ')

        print('\n1 - Sim.')

        print('Qualquer caracter - Não.')

        save\_input = input()

        if save\_input == '1':

            return True

        return False

Data base manager

import json

class DataBaseManager:

    def \_\_init\_\_(self, ficheiro):

        self.ficheiro\_db = ficheiro

        # Convenção para propriedades e métodos privados (python não tem metodos privados)

        self.\_max\_name\_size = 10

        self.\_dados = {}

    def get\_item(self, code):

        if self.check\_code(code):

            return self.\_dados[code]

        return None

    def \_\_str\_\_(self):

        string = f' {"Code":8s}  |  {"Name":{self.\_max\_name\_size}s}  |  {"Price":6s}\n'

        for key, value in self.\_dados.items():

            string += self.\_elem\_to\_str(key, value)

        return string

    def \_elem\_to\_str(self, code, value, use\_name\_length=False):

        name\_size = self.\_max\_name\_size

        if use\_name\_length:

            name\_size = len(value["Name"])

        return f' {code:8s}  |  {value["Name"]:{name\_size}s}  |  {value["Price"]:8.2f}€\n'

    def \_update\_max\_name\_size(self):

        if len(self.\_dados) == 0:

            self.\_max\_name\_size = 10

            return

        self.\_max\_name\_size = max([len(item["Name"]) for code, item in self.\_dados.items()])

    def get\_elem\_as\_str(self, code):

        if not self.check\_code(code):

            print(f"O código {code} não existe.")

            return ""

        return self.\_elem\_to\_str(code, self.\_dados[code], True)

    def insert(self, code, item\_name, price):

        if self.check\_code(code):

            print(f"O código {code} já existe, foi impossível adicionar.")

            return False

        self.\_dados[code] = {"Name": item\_name, "Price": price}

        self.\_max\_name\_size = max(len(item\_name), self.\_max\_name\_size)

        return True

    def remove(self, code):

        if not self.check\_code(code):

            print(f"O código {code} não existe.")

            return

        del self.\_dados[code]

        self.\_update\_max\_name\_size()

    def update(self, code, name, price):

        if not self.check\_code(code):

            print(f"O código {code} não existe.")

            return False

        self.\_dados[code] = {"Name": name, "Price": price}

        self.\_update\_max\_name\_size()

        return True

    def check\_code(self, code):

        if code in self.\_dados:

            return True

        return False

    def import\_dados(self):

        with open(self.ficheiro\_db, encoding='utf-8') as data\_base:    #Importação do ficheiro dados.json e codificação dos caracteres

            self.\_dados = json.load(data\_base)

        self.\_update\_max\_name\_size()

    def save(self):

        with open(self.ficheiro\_db, "w") as json\_file:

            json.dump(self.\_dados, json\_file, indent=4)

Data Base Orçamentos

import json

import datetime

class DataBaseOrcamentos:

    def \_\_init\_\_(self, ficheiro=None):

        if ficheiro is None:

            self.ficheiro\_db\_orcamento = self.new\_name()

        else:

            self.ficheiro\_db\_orcamento = ficheiro

        # Convenção para propriedades e métodos privados (python não tem metodos privados)

        self.\_max\_name\_size = 10

        self.total = 0

        self.total\_impostos = 0

        self.\_dados\_orcamento = {}

    def new\_name(self):

        time = datetime.datetime.now()

        return 'orcamentos\\' + time.strftime('%Y-%m-%d\_%Hh-%Mm-%Ss') + '.json'

    def import\_dados(self):

        with open(self.ficheiro\_db\_orcamento, encoding='utf-8') as data\_base:    #Importação do ficheiro dados.json e codificação dos caracteres

            self.\_dados\_orcamento = json.load(data\_base)

        self.\_update\_max\_name\_size()

    def check\_code(self, code):

        if code in self.\_dados\_orcamento:

            return True

        return False

    def get\_elem\_as\_str(self, code):

        if not self.check\_code(code):

            print(f"O código {code} não existe.")

            return ""

        return self.\_elem\_to\_str(code, self.\_dados\_orcamento[code], True)

    def insert(self, code, quantity, store\_db):

        if self.check\_code(code):

            print(f"O código {code} já existe, vamos substituir pela nova quantidade.")

        if not store\_db.check\_code(code):

            print('O código não exite em loja!')

            return False

        item = store\_db.get\_item(code)

        self.\_dados\_orcamento[code] = {

                                        'Name': item['Name'],

                                        'Quantity': quantity,

                                        'Price': item['Price']

                                        }

        self.\_max\_name\_size = max(len(store\_db.get\_item(code)["Name"]), self.\_max\_name\_size)

        return True

    def remove(self, code):

        if not self.check\_code(code):

            print(f"O código {code} não existe.")

            return

        del self.\_dados\_orcamento[code]

        self.\_update\_max\_name\_size()

    def save(self):

        with open(self.ficheiro\_db\_orcamento, "w") as json\_file:

            json.dump(self.\_dados\_orcamento, json\_file, indent=4)

    def is\_empty(self):

        return not self.\_dados\_orcamento.keys()

#periféricos do código

    def \_elem\_to\_str(self, code, item, use\_name\_length=False):

        name\_size = self.\_max\_name\_size

        if use\_name\_length:

            name\_size = len(item["Name"])

        return f' {code:8s}  |  {item["Name"]:{name\_size}s}  |  {item["Quantity"]:8d}  |  {item["Price"]:8.2f}€\n'

    def calcutate\_totals(self):

        self.total = 0

        self.total\_impostos = 0

        for code, item in self.\_dados\_orcamento.items():

            self.total += item['Quantity'] \* item['Price']

        self.total\_impostos = self.total \* 0.23

    def \_update\_max\_name\_size(self):

        if len(self.\_dados\_orcamento) == 0:

            self.\_max\_name\_size = 10

            return

        self.\_max\_name\_size = max([len(item["Name"]) for code, item in self.\_dados\_orcamento.items()])

    def \_\_str\_\_(self):

        string = f' {"Code":8s}  |  {"Name":{self.\_max\_name\_size}s}  |  {"Quantity":8s}  |  {"Price":6s}\n'

        string += '-'\*(43+self.\_max\_name\_size)+'\n'

        for key, value in self.\_dados\_orcamento.items():

            string += self.\_elem\_to\_str(key, value)

        string += f'\n{"IVA":6}= {self.total\_impostos:.2f}€\n'

        string += f'{"Total":6s}= {self.total:.2f}€\n'

        return string