**Project 1\_2B: Perform Operations Using Modules & Tools**

Project: You will create three lists that each contain ten values. One will contain product names, another product costs and the third a three-digit employee code. Your task will be to merge all three lists into a single navigable array.

# Создаем списки:

cost = [100.0, 200.0, 300.0, 400.0, 500.0, 600.0, 700.0, 800.0, 900.0, 1000.0]

names = ['product1', 'product2', 'product3', 'product4', 'product5', 'product6', 'product7', 'product8', 'product9', 'product10']

codes = [111, 222, 333, 444, 555, 666, 777, 888, 999, 112]

# Собираем общий массив:

newArray = list(zip(cost, names, codes))

# Печать результатов:

print('newArray:', newArray)}

**Project 1\_3B: Control Flow with Decisions and Loops**

**Задание 1. Числа**

Create a list of randomly generated numbers between 1 and 100. Based on the first list, generate a second that has the word “High” or “Low” depending on whether the corresponding number in the first list is greater than or less than 50.

# Импорт функций:

from random import randint

# Создаем пустой список:

list1 = []

y = 1

# Создаем пустой список:

m = int(input('сколько элементом нужно в спике:'))

# Заполняем список:

while y <= m:

list1.append(randint(1, 100))

y += 1

# Печатаем список:

print(list1)

# Создаем пустые списки:

list2 = []

list3 = []

# Заполняем списки:

for x in list1:

if (x < 50):

list2.append(x)

else:

list3.append(x)

# Печатаем список:

print("list2(x < 50):", list2)

print("list3(x => 50):", list3)

**Задание 2. Имена**

Generate a list of 100 names. Generate two new lists. One with the names where the first character begins with a letter between “a” and “m” with the other lists containing the other names.

# Создаем список с именами:

#names = ['Liam', 'Olivia', 'William', 'Isabella', 'Oliver', 'Charlotte', 'Elijah', 'Amelia', 'Mason', 'Evelyn',

# 'Alexander', 'Emily', 'Jacob', 'Mila', 'Daniel', 'Avery', 'Jackson', 'Camila', 'Aiden', 'Scarlett',

# 'Samuel', 'Madison', 'Joseph', 'Grace', 'Owen', 'Penelope', 'John', 'Riley', 'Luke', 'Nora',

# 'Dylan', 'Eleanor', 'Levi', 'Lillian', 'Gabriel', 'Aubrey', 'Mateo', 'Stella', 'Jaxon', 'Zoe',

# 'Joshua', 'Hazel', 'Andrew', 'Aurora', 'Caleb', 'Audrey', 'Asher', 'Bella', 'Thomas', 'Skylar',

# 'Isaiah', 'Paisley', 'Josiah', 'Anna', 'Christian', 'Nova', 'Connor', 'Emilia', 'Ezra', 'Samantha',

# 'Landon', 'Willow', 'Jonathan', 'Naomi', 'Jeremiah', 'Elena', 'Elias', 'Ariana', 'Cameron', 'Gabriella',

# 'Robert', 'Madelyn', 'Maverick', 'Ruby', 'Dominic', 'Serenity', 'Greyson', 'Adeline', 'Ian', 'Gianna',

# 'Santiago', 'Isla', 'Cooper', 'Quinn', 'Roman', 'Ivy', 'Ezekiel', 'Piper', 'Jose', 'Alexa',

# 'Jameson', 'Emery', 'Bryson', 'Delilah', 'Everett', 'Vivian', 'Kayden', 'Sophie', 'Sawyer', 'Madeline']

# Создаем пустые списки:

#names2 = []

#names3 = []

# Цикл:

#for x in names:

# s = x[0]

# if (ord(s)<=ord('M')):

# names2.append(x)

# else:

# names3.append(x)

# печать результатов:

#print("names2(x <= M):", names2)

#print("neame3(x > M):", names3)

**Project 1\_4B: Perform Input and Output Operations**

**Project:** Load the …\data\orderdata\_sample.csv file.

Use the “Quantity”, “Price” and “Freight” columns to generate a new “Total” column (e.g. Quantity \* Price + Freight).

Save all the data to a new file

# ### Работа с форматом CSV

# In[62]:

# Read the CSV file using the csv module

import csv

informationHeaders = []

information = []

with open('D:\data\orderdata\_sample.csv') as csvfile:

count = 0

inputcsv = csv.reader(csvfile, delimiter=',')

for i in inputcsv:

if (count == 0):

curLin = i

newColumn = (i[3] +" \* "+ i[4] +" + "+ i[5])

curLin.append(newColumn)

informationHeaders = curLin

information.append(curLin)

#print(informationHeaders)

else:

newValue = round((float(i[3])+float(i[4])+float(i[5])),2)

curLin = i

curLin.append(newValue)

#print(curLin)

information.append(curLin)

count = count+1

# In[ ]:

# Запись в файл:

with open('D:\data\orderdata\_sampleWrite.csv', mode='w', newline='') as csvfile:

fieldnames = informationHeaders

writer = csv.DictWriter(csvfile, fieldnames=fieldnames)

for lin in information:

writer.writerow({'OrderID':lin[0], 'CustomerID':lin[1], 'ManagerID':lin[2], 'Quantity':lin[3], 'Price':lin[4], 'Freight':lin[5], 'OrderDate':lin[6], 'ShippedDate':lin[7], 'Quantity \* Price + Freight':lin[8]})

# Конец

**Project 1\_5: Document and Structure Code**

***Упражнение 1***

Дан список температурных изменений в течение дня (целые числа). Известно, что измеряющее устройство иногда сбоит и записывает отсутствие температуры (значение None). Выведите среднюю температуру за наблюдаемый промежуток времени, предварительно очистив список от неопределенных значений. Гарантируется, что хотя бы одно определенное значение в списке есть.

def makeBuitiful(list1):

listOfAllResults = []

list2 = []

sum = 0

for val in list1:

if val != None:

list2.append(val)

sum = sum + val

mathAverage = sum / len(list1)

listOfAllResults.append(list2)

listOfAllResults.append(sum)

listOfAllResults.append(mathAverage)

return listOfAllResults

# Исходный список:

list1 = [0, 23, 234, 89, None, 0, 35, 9]

# Печать исходного списка:

print("list1:", list1)

# Применение функции:

listOfAllResults = makeBuitiful(list1)

# Применение функции:

list2 = listOfAllResults[0]

sum = listOfAllResults[1]

mathAverage = listOfAllResults[2]

# Печать результатов:

print("list2:", list2)

print("sum(list2):", sum)

print("mathAverage(list2):", mathAverage)

***Упражнение 2***

Напишите функцию, которая принимает неограниченное количество числовых аргументов и возвращает кортеж из двух списков: отрицательных значений (отсортирован по убыванию); неотрицательных значений (отсортирован по возрастанию).

def makeTuple(list0):

list1 = []

list2 = []

for val in list0:

if val < 0:

list1.append(val)

else:

list2.append(val)

list1.sort(reverse = True)

list2.sort()

tupleOfResults = (list1, list2)

return tupleOfResults

# Импорт функций:

from random import randint

# Создаем пустой список:

list0 = []

y = 1

# Создаем пустой список:

m = int(input('сколько элементом нужно в спике:'))

# Заполняем список:

while y <= m:

list0.append(randint(-100, 100))

y += 1

# Печатаем список:

print("Исходный список:")

print(list0)

# Применение функции:

tuple1 = makeTuple(list0)

# Печать результатов:

print("Кортеж:")

print(tuple1)

**Задание 3**

Составьте две функции для возведения числа в степень: один из вариантов реализуйте в рекурсивном стиле

def exptRec(b, n):

if n==0:

return 1

return b\*exptRec(b, n-1)

def exptLin(b, n):

result = b\*\*n

return result

# Проверка функций:

result1 = exptRec(2, 10)

result2 = exptLin(2, 10)

# Печать результатов:

print("2^10(exptRec):", result1)

print("2^10(exptLin):", result2)

**Project 1\_6B: Perform Troubleshooting and Error Handling**

Создайте сценарий, который использует список имен файлов CSV в качестве источника для копирования файлов в плоский файл. Текущая дата и время должны быть добавлены к имени файла в качестве префикса перед копированием. Каждая операция копирования должна быть записана в файл журнала на локальном компьютере, после чего локальный файл будет удален. Исключения для файлов, которые не были найдены, также должны быть записаны в журнал

import pandas as pd

import csv

from datetime import datetime

import dateutil.parser

def copy\_csv(fullFileInputName,fullFileOutputName):

print(fullFileInputName," ",fullFileOutputName)

df = pd.read\_csv(fullFileInputName)

#print(df)

df.to\_csv(fullFileOutputName)

dirfile = "D:\\data\\" # можете заменить на свой каталог

expansion = ".csv"

namefileList = []

resultfileList = []

informationHeaders = ["i", "fullFileInputName", "FileOutputName", "time", "status"]

information = []

namefileList.append("input")

namefileList.append("ground")

count = 1

for i in namefileList:

dataTime = datetime.strftime(datetime.now(), "%Y-%m-%d\_%H-%M-%S")

fullFileInputName = dirfile + i + expansion

fullFileOutputName = dirfile + i + "Copy" + dataTime + expansion

## Try to delete the file

try:

copy\_csv(fullFileInputName, fullFileOutputName)

curLin = [count, fullFileInputName, fullFileOutputName, dataTime, "success"]

information.append(curLin)

except OSError as e:

curLin = [count, fullFileInputName, fullFileOutputName, dataTime, "fail"]

information.append(curLin)

#print("Error: {} - {}!".format(e.filename, e.strerror))

count = count + 1

#Конец цикла:

# Запись в логов в файл:

with open('D:\data\logs\_Bubyrev.csv', mode='w', newline='') as csvfile:

fieldnames = informationHeaders

writer = csv.DictWriter(csvfile, fieldnames=fieldnames)

for lin in information:

writer.writerow({'i': str(lin[0]), 'fullFileInputName': str(lin[1]), 'FileOutputName': str(lin[2]), 'time': str(lin[3]), 'status': str(lin[4])})

print(str(lin[0]).ljust(20), str(lin[1]).ljust(20), str(lin[2]).ljust(30), str(lin[3]).ljust(25), str(lin[4]).ljust(20))

# Конец

print("Программа завершена!")

**Итоговая работа № 1**

def returnList(address):

information = []

with open(address) as csvfile:

count = 0

inputcsv = csv.reader(csvfile, delimiter=',')

for i in inputcsv:

if (count == 0):

curLin = i

information.append(curLin)

# print(informationHeaders)

else:

curLin = i

# print(curLin)

information.append(curLin)

count = count + 1

# In[ ]:

return information

# Конец функции

def reformat(s):

ss = s.strip().split('\n')

tb = [i.strip().split('\t') for i in ss]

length = max(max(len(w) for w in words) for words in tb) + 4

return '\n'.join(''.join(f'{w:{length}}' for w in words) for words in tb)

# Импорт функций:

import csv

import os

import operator

# Начало программы:

print("Hello!")

print()

print(" 1. Add")

print(" 2. Show all")

print(" 3. Show for date")

print(" 4.Show by category")

print(" 5.Show by min->max")

print(" 6. Delete")

print(" 0. Exit")

print()

# Исходная информация:

address = 'D:\data\orderdata\_Bubyrev.csv'

informationHeaders = ['Category', 'Product', 'Cost', 'Date']

# Цикл:

while True:

try:

x = int(input("What do you like to do?"))

if x == 1:

print("Action: 1. Add")

Category = (input("Category:"))

Product = (input("Product:"))

Cost = (input("Cost:"))

Date = (input("Date(Example: 2018-02-02):"))

# Запись в файл:

information = returnList(address)

# Запись в файл:

with open(address, mode='w', newline='') as csvfile:

fieldnames = informationHeaders

writer = csv.DictWriter(csvfile, fieldnames=fieldnames)

for i in information:

writer.writerow({'Category': i[0], 'Product': i[1], 'Cost': i[2], 'Date': i[3]})

writer.writerow({'Category': Category, 'Product': Product, 'Cost': Cost, 'Date': Date})

# Конец

print("Thanks! Have a good day!")

elif x == 2:

print("Action: 2. Show all")

print(informationHeaders[0].ljust(15), informationHeaders[1].ljust(15), informationHeaders[2].ljust(15), informationHeaders[3].ljust(15))

information = returnList(address)

for i in information:

print(i[0].ljust(15), i[1].ljust(15), i[2].ljust(15), i[3].ljust(15))

elif x == 3:

print("Action: 3. Show for date")

reqDate = (input("Chose date:"))

information = returnList(address)

for i in information:

if (i[3] == reqDate):

print(i)

elif x == 4:

print("Action: 4.Show by category")

reqCategory = (input("Chose category:"))

information = returnList(address)

for i in information:

if (i[0] == reqCategory):

print(i)

elif x == 5:

print("Action: 5.Show by min->max")

information = returnList(address)

sort = sorted(information, key=operator.itemgetter(2))

print(informationHeaders[0].ljust(15), informationHeaders[1].ljust(15), informationHeaders[2].ljust(15), informationHeaders[3].ljust(15))

for i in information:

print(i[0].ljust(15), i[1].ljust(15), i[2].ljust(15), i[3].ljust(15))

elif x == 6:

print("Action: 6. Delete")

## Try to delete the file

try:

os.remove(address)

print("The file: {} is deleted!".format(address))

except OSError as e:

print("Error: {} - {}!".format(e.filename, e.strerror))

print("Information delete!")

elif x == 0:

print("Action: 0. Exit")

break

else:

print("Oops! That was no valid number. Try again...")

except ValueError:

print("Oops! That was no valid number. Try again...")

# Конец программы:

}

**pythonProject.TestWork.02.01**

Реализованы классы с насследованием, композицией и переопределением операторов.

Замысел зачетного задания состоит в следующем: когда мы измеряем некую природную величину (например мы измеряем силу источника света, размер готовой детали, расход воды в водопроводе) то многократно измеряя одну и ту же величину(многократные измерения нужны для повышения точности) одним и тем же прибором будут получаться немного разные результаты (пусть и отличающиеся на доли процента). Тем не менее в наш ряд измерений могут закрасться неверные результаты, получившиеся от ошибок исследователя/или чего-то другого. Для выбраковки из ряда измерений неверной величины(ошибки) применяются различные коэффициенты. В своей работе я реализовал работу коэффициентов Шарли и Романовского. Ход работы: исследователь получает ряд измерений, с помощью метрологических коэффициентов из ряда измерений исключаются грубые ошибки (если такие присутствуют), из ряда измерений с выброшенными ошибками мы находим среднее арифметическое, которое и будет являться максимально приближенным к истине значением измеряемой величины.

**pythonProject.TestWork.02.02**

Добавлен абстрактный класс с абстрактными методами, добавлен класс миксин. Изменения прописаны в самом начале и самом конце файла.

**pythonProject.TestWork.02.03**

Добавлен декоратор(выводит в консоль название запущенной функции) и метакласс (содержит информацию о пользователе и передает её в класс измерений). Изменения указаны вначале и в конце фаула.

**pythonProject.TestWork.02.05**

Добавлена многопоточная и асинхронная работа с расчётом нескольких серий измерений. Изменения даны в самом конце файла.

**pythonProject.TestWork.02.06**

Добавлен графический интерфейс к программе. Изменения указаны в конце документа.