Министерство науки и высшего образования Российской Федерации

федеральное государственное бюджетное образовательное учреждение
высшего образования
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
МОРДОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
им. Н.П. ОГАРЁВА»
(ФГБОУ ВО «МГУ им. Н.П. Огарёва»)

Факультет математики и информационных технологий

Кафедра фундаментальной информатики

ЗАДАНИЕ
на учебную практику

Технологическая (проектно-технологическая) практика

Студенту Новикову Максиму Евгеньевичу 3 курса 302 группы

Направление подготовки 02.03.02 Фундаментальная информатика и информационные технологии

Место прохождения практики ФГБОУ ВО «Национальный исследовательский Мордовский государственный университет им. Н.П. Огарёва», факультет математики и информационных технологий, кафедра фундаментальной информатики, лаб. № 203

Срок прохождения практики 02.09.2024 г. по 31.12.2024 г. (объем 2 нед.)

Срок представления отчёта студентом на защиту 30.12.2024

1. Цели и задачи практики

1.1. Цель проведения учебной практики состоит в систематизации знаний и формировании навыков практического применения ЭВМ для решения различных прикладных задач.

1.2. Задачи практики:

Задачи практики: закрепить умение реализовывать при решении различных практических задач алгоритмы и структуры данных, методы программирования и компьютерные технологии: - закрепить и углубить теоретические знания по использованию различных языков и методов программирования и инструментальных систем разработки приложений; - выработать устойчивые навыки решения практических задач на ЭВМ с

Основная часть

Лабораторная работа №1.

Задание:

1) Пользователь вводит 12 целых чисел из отрезка [1, 1000]. Вывести на экран наибольшее и наименьшее тех из них, которые являются полными кубами.
2) Есть два списка (двух групп студентов):
   a. Заменить три случайно выбранных элемента первой группы на три случайно выбранных элемента второй группы.
   b. Удалить из каждой группы предпоследний элемент. Списки предварительно отсортировать в лексикографическом порядке.
3) «Карманные деньги». У студента есть n рублей (вводится с клавиатуры). Кусок пиццы стоит 80 рублей, хлеб – 45 рублей, лапша быстрого приготовления – 15 рублей, молоко – 85 рублей, шоколадный батончик – 55 рублей, чипсы – 67 рублей, шаурма – 190 рублей. Вывести на экран продукты, которые он может купить (пицца и хлеб, пицца и лапша, пицца+хлеб+молоко, пицца+пицца+молоко и т.д.) и сколько денег у него останется при каждом варианте покупки.
4) Пользователь вводит время, вывести на экран, какая это по счету пара (в 08.00 начинается первая) и сколько времени прошло с начала пары. Вывести на экран, какой угол при этом составляет часовая и минутная стрелки часов.
5) Таисия 2 января 2024 года приобрела облигацию компании «AI2024» номинальной стоимостью 1 000 рублей за 800 рублей. По облигации предусмотрена купонная выплата (из расчета 9% годовых) 12 раз в год первого числа каждого месяца, в рублях по курсу юаня. 31 декабря 2025 г. Таисия продала ценную бумагу за 1100 рублей. Рассчитать доходность к погашению.

Решение.

1)

```
fullcube = [1,8,27,64,125,216,343,512,729,1000]
tempo = []
lis = []
'''
for i in range(12):
  hello = str(i+1) + '-th:'
  lis.append(int(input(hello)))
  for cube in fullcube:
    if(cube==lis[i]):
      tempo.append(lis[i])
'''
#input search cube
for i in range(12):
  hello = str(i) + '-ое число:'
  prove = int(input(hello))
  while (not 0<=prove<=1000):
    print('!!!!!!Число за пределами [1;1000]!!!!!!')
    prove = int(input(hello))
  lis.append(prove)
  for cube in fullcube:
    if(cube==lis[i]):
      tempo.append(lis[i])
```

```
#max min out
if(len(tempo)!=0):
   max = min = tempo[0]
   for i in range(len(tempo)-1):
      if (max < tempo[i+1]):
         max = tempo[i+1]
      if (min > tempo[i+1]):
         min = tempo[i+1]

   print('max = ',max)
   print('min= ',min)
else:
   print('netu')
```

Результат:

```
0-oe число:10
1-oe число:2
2-oe число:8
3-oe число:9
4-oe число:123
5-oe число:27
6-oe число:512
7-oe число:216
8-oe число:55
9-oe число:44
10-oe число:8
11-oe число:64
max =  512
min=  8
```

2)

```
list1 = ['Иванов', 'Алексеев', 'Николаев', 'Александров']
list2 = ['Ивановский', 'Алексеевский', 'Николаевский', 'Александровский']
list3 = ['Иванович', 'Алексеевич', 'Николаевич', 'Александрович']

import random
def printAll():
  print('--------------------')
  print("list 1:", list1)
  print("list 2:", list2)
  print("list 3:", list3)
  print('--------------------')

printAll()

def replace_elements(list1, list2, list3):
    for _ in range(2):
        pos1 = random.randint(0, len(list1) - 1)
        swap = list1[pos1]
        pos2 = random.randint(0, len(list1) - 1)
        list1[pos1] = list2[pos2]
        list2[pos2] = swap
        print('Swapped '+ str(pos1) + ' and ' + str(pos2))
    #print('Swapped 1 and 2')
    printAll()
```

```
    for _ in range(3):
        pos2 = random.randint(0, len(list2) - 1)
        swap = list2[pos2]
        pos3 = random.randint(0, len(list3) - 1)
        list2[pos2] = list3[pos3]
        list3[pos3] = swap
        print('Swapped '+ str(pos2) + ' and ' + str(pos3))
    printAll()
def remove_third_element(list):
    del list[2]

replace_elements(list1, list2, list3)

print('----------------sorted----------------')

list1.sort()
list2.sort()
list3.sort()
printAll()

print('----------------after remove----------------')


remove_third_element(list1)
remove_third_element(list2)
remove_third_element(list3)

printAll()
```

Результат:

```
--------------------
list 1: ['Иванов', 'Алексеев', 'Николаев', 'Александров']
list 2: ['Ивановский', 'Алексеевский', 'Николаевский', 'Александровский']
list 3: ['Иванович', 'Алексеевич', 'Николаевич', 'Александрович']
--------------------
Swapped 1 and 3
Swapped 1 and 2
--------------------
list 1: ['Иванов', 'Николаевский', 'Николаев', 'Александров']
list 2: ['Ивановский', 'Алексеевский', 'Александровский', 'Алексеев']
list 3: ['Иванович', 'Алексеевич', 'Николаевич', 'Александрович']
--------------------
Swapped 0 and 0
Swapped 2 and 0
Swapped 0 and 3
--------------------
list 1: ['Иванов', 'Николаевский', 'Николаев', 'Александров']
list 2: ['Александрович', 'Алексеевский', 'Ивановский', 'Алексеев']
list 3: ['Александровский', 'Алексеевич', 'Николаевич', 'Иванович']
--------------------
----------------sorted----------------
--------------------
list 1: ['Александров', 'Иванов', 'Николаев', 'Николаевский']
list 2: ['Александрович', 'Алексеев', 'Алексеевский', 'Ивановский']
list 3: ['Александровский', 'Алексеевич', 'Иванович', 'Николаевич']
--------------------
----------------after remove----------------
--------------------
list 1: ['Александров', 'Иванов', 'Николаевский']
list 2: ['Александрович', 'Алексеев', 'Ивановский']
list 3: ['Александровский', 'Алексеевич', 'Николаевич']
--------------------
```

3)

```
items = {
    "Пицца": 80,
    "Хлеб": 45,
    "Лапша бп": 15,
    "Молоко": 85,
    "Шоколадный батончик": 55,
    "Чипсы": 67,
    "Шаурма": 190
}

n = int(input("Введите деньги: "))

def generate_combinations(items):
    item_list = list(items.keys())
    total_items = len(item_list)
    combinations = []

    for i in range(1, 1 << total_items):
        combo = []
        cost = 0
        for j in range(total_items):
            if (i & (1 << j)) > 0:
                combo.append(item_list[j])
                cost += items[item_list[j]]
        combinations.append((combo, cost))

    return combinations
```

```
combinations = generate_combinations(items)

for combo, total_cost in combinations:
    if total_cost <= n:
        remaining_money = n - total_cost
        items_names = ', '.join(combo)
        print(f"Можно купить: {items_names}. Стоимость: {total_cost} рублей. Осталось: {remaining_money} рублей.")
```

Результат:

```
Введите деньги: 50
Можно купить: Хлеб. Стоимость: 45 рублей. Осталось: 5 рублей.
Можно купить: Лапша бп. Стоимость: 15 рублей. Осталось: 35 рублей.
{'Пицца': 80,
 'Хлеб': 45,
 'Лапша бп': 15,
 'Молоко': 85,
 'Шоколадный батончик': 55,
 'Чипсы': 67,
 'Шаурма': 190}
```

4)

```
from __future__ import division
time = input('Введите время Hh:Mm:')

def AddTime(start_hours, start_minutes):
  start_hours +=1
  start_minutes +=30
  if(start_minutes>=60):
    start_hours += 1
    start_minutes -=60
  return(start_hours, start_minutes)

def LowerTime(start_hours, start_minutes, rem):
  start_hours -=1
  start_minutes -=(30 + rem)
  if(start_minutes < 0):
    start_hours -= 1
    start_minutes +=60
  return(start_hours, start_minutes)


def AddMin(start_hours, start_minutes, add):
  start_minutes += add
  if(start_minutes>=60):
      start_hours += 1
      start_minutes -=60
  return(start_hours, start_minutes)

def DivTime(hours, minutes, start_hours, start_minutes):
  hours -= start_hours
  minutes -= start_minutes
  if(minutes < 0):
    hours -= 1
    minutes +=60
  return(hours, minutes)
```

```
def Compare(hours, minutes, start_hours, start_minutes):
  if (hours<start_hours):
    return(False)
  elif(hours == start_hours):
    if(minutes < start_minutes):
      return(False)
    else:
      return(True)
  else:
    return(True)

hours = int(time.split(':')[0])
minutes = int(time.split(':')[1])


hourAngle = (hours%12 * 30) + minutes * 0.5
minuteAngle = minutes * 6
hourAngle = abs(hourAngle - minuteAngle)
if(hourAngle > 180):
  hourAngle = 360 - hourAngle

print(f'Угол между часовой и минутной стрелкой равно: {hourAngle}°')

while(minutes>60 and hours>24):
  print('Записана неверное время')
  time = int(input('Введите время HH:MM:'))

pairs = True

if (hours<8):
  print('Пары ещё не начались')
  pairs = False
elif (hours>=21):
  if(hours==21 and minutes>20):
    print('Пары закончились')
    pairs = False
  else:
    print('Сейчас 8-ая пара')
```

```
    print(f'Прошло {hours-20}:{abs(10+minutes)} с начала пары')
    pairs = False


start_hours = 8
start_minutes = 0
not_find = True;
notBreak = True;
i=0

if (pairs):
  while(not_find and (i<8)):
    (start_hours, start_minutes) = AddTime(start_hours, start_minutes)
    if (i == 0 or i ==2):
      add = 15
    elif (i == 1):
      add = 20
    elif (i == 3 or i == 4):
      add = 10
    else:
      add = 5
    notBreak = not_find = Compare(hours, minutes, start_hours, start_minutes)
    (start_hours, start_minutes) = AddMin(start_hours, start_minutes, add)
    not_find = Compare(hours, minutes, start_hours, start_minutes)
    i+= 1

  (start_hours, start_minutes) = LowerTime(start_hours, start_minutes, add)
  (hours, minutes) = DivTime(hours, minutes, start_hours, start_minutes)
  if(notBreak):
    print(f'Сейчас перерыв после {i}-ой пары')
    print(f'Прошло {hours}:{minutes} с начала пары')
  else:
    print(f'Сейчас идёт {i}-ая пара')
    print(f'Прошло {hours}:{minutes} с начала пары')
```

Результат:

```
Введите время Hh:Mm:11:46
Угол между часовой и минутной стрелкой равно: 77.0°
Сейчас идёт 3-ая пара
Прошло 0:11 с начала пары
```

5)

```
face_value_yuan = 80
purchase_price_yuan = 800 / 12.58
coupon_rate = 0.09
coupon_frequency = 12

years = 2
n_payments = coupon_frequency * years

C_yuan = coupon_rate * face_value_yuan

YTM_yuan = ((C_yuan + ((face_value_yuan - purchase_price_yuan) / n_payments)) /
              ((face_value_yuan + purchase_price_yuan) / 2))

YTM_percentage = YTM_yuan * (1 / years) * 100
print(f"Доходность к погашению:: {YTM_percentage:.2f}%")
```

```
Доходность к погашению:: 5.49%
```

Лабораторная работа №2.

Задание: 1)Построить круговую и столбцовую диаграммы распределения подписчиков групп vk г. N (где N = последняя_цифра_номера_зачетки div 10) по полу, возрасту, вузу. (Использовать VK API)

N:

1 – Нижний Новгород
2 – Чебоксары
3 – Казань
4 – Пенза
5 – Ульяновск
6 – Самара
7 – Рязань
8 – Саратов
9 – Уфа
0 – Челябинск

Решение.

```
import vk_api
import requests
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
from datetime import datetime


ACCESS_TOKEN = 'aboba'
GROUP_ID = 'ufa'
API_VERSION = '5.131'
```

```
def get_group_members(group_id, access_token):
    url = 'https://api.vk.com/method/groups.getMembers'
    data = {
        'group_id': group_id,
        'access_token': access_token,
        'v': API_VERSION,
        'fields': 'sex, education, bdate, city',
    }

    response = requests.post(url, data)
    response.raise_for_status()
    data = response.json()
    return data['response']

members = get_group_members(GROUP_ID, ACCESS_TOKEN)
```

```
df = pd.DataFrame(members['items'])
df = df[df['is_closed'] == False]
df = df.loc[:, ['bdate','city','sex','university_name']]

#df.university_name.fillna(value = 'Не указано', inplace=True)
df.dropna(inplace=True)
```

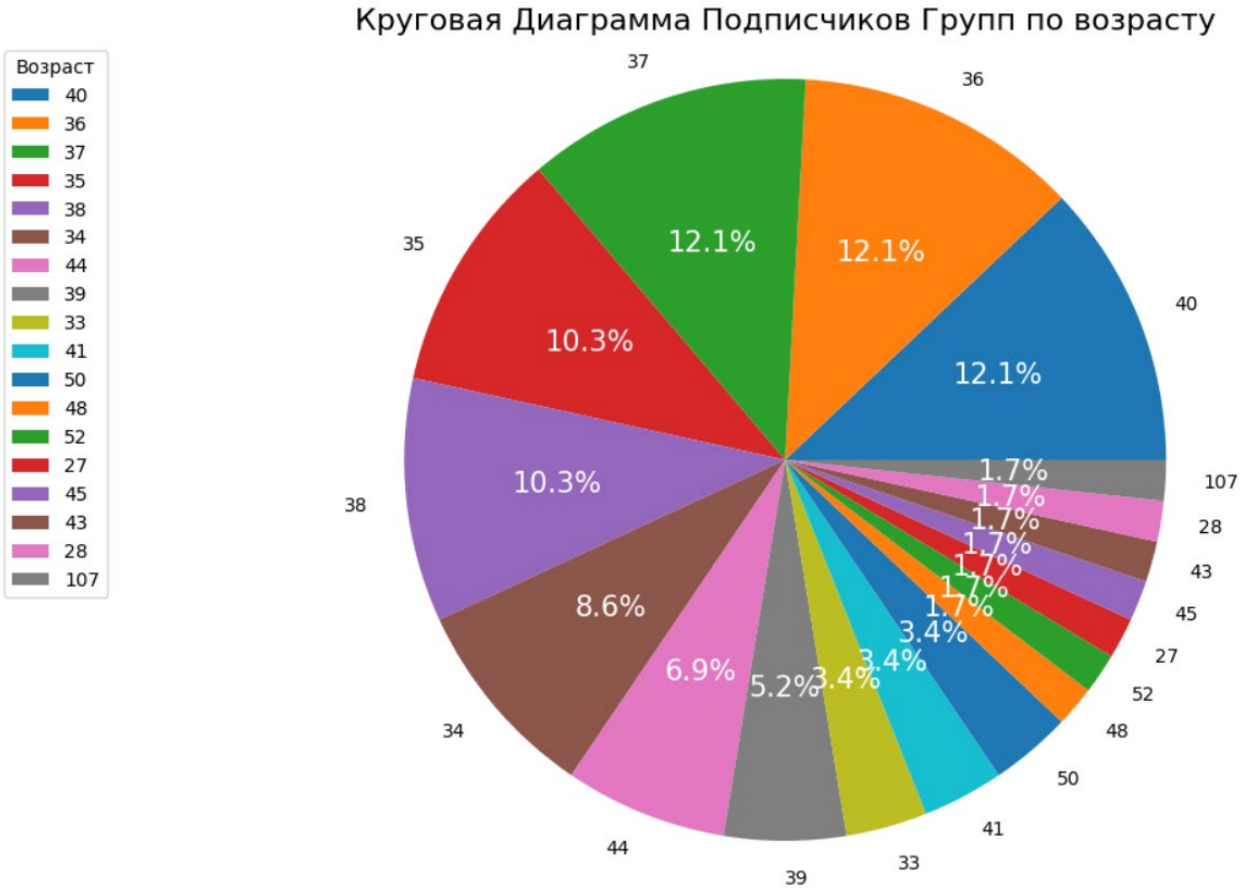
```
df['city_title'] = df['city'].apply(lambda x: x['title'])
df = df.drop(columns=['city'])
df = df[df['bdate'].str.contains(r'\d{4}')]
df['bdate'] = pd.to_datetime(df['bdate'], format='%d.%m.%Y')
```

```
#sns.heatmap(df[cols].isnull(), cmap=sns.color_palette(colours))
current_date = datetime.now()
```

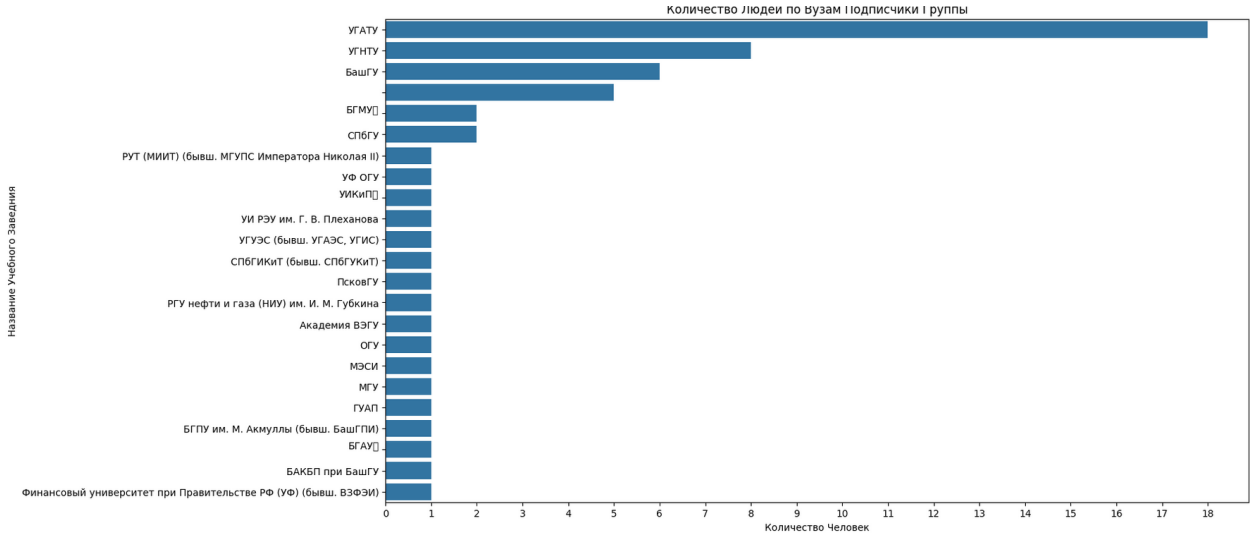
```
df['age'] = current_date.year - df['bdate'].dt.year

df['age'] -= ((current_date.month < df['bdate'].dt.month) |
              ((current_date.month == df['bdate'].dt.month) &
               (current_date.day < df['bdate'].dt.day))).astype(int)

gender_map = {1: 'Female', 2: 'Male'}
df['sex'] = df['sex'].map(gender_map)
df['count'] = 1
df = df.drop(columns=['bdate'])
df.reset_index(drop=True)
```

```
cols = df.columns[:30]
colours = ['#000', '#ff0']
sns.heatmap(df[cols].isnull(), cmap=sns.color_palette(colours))
```

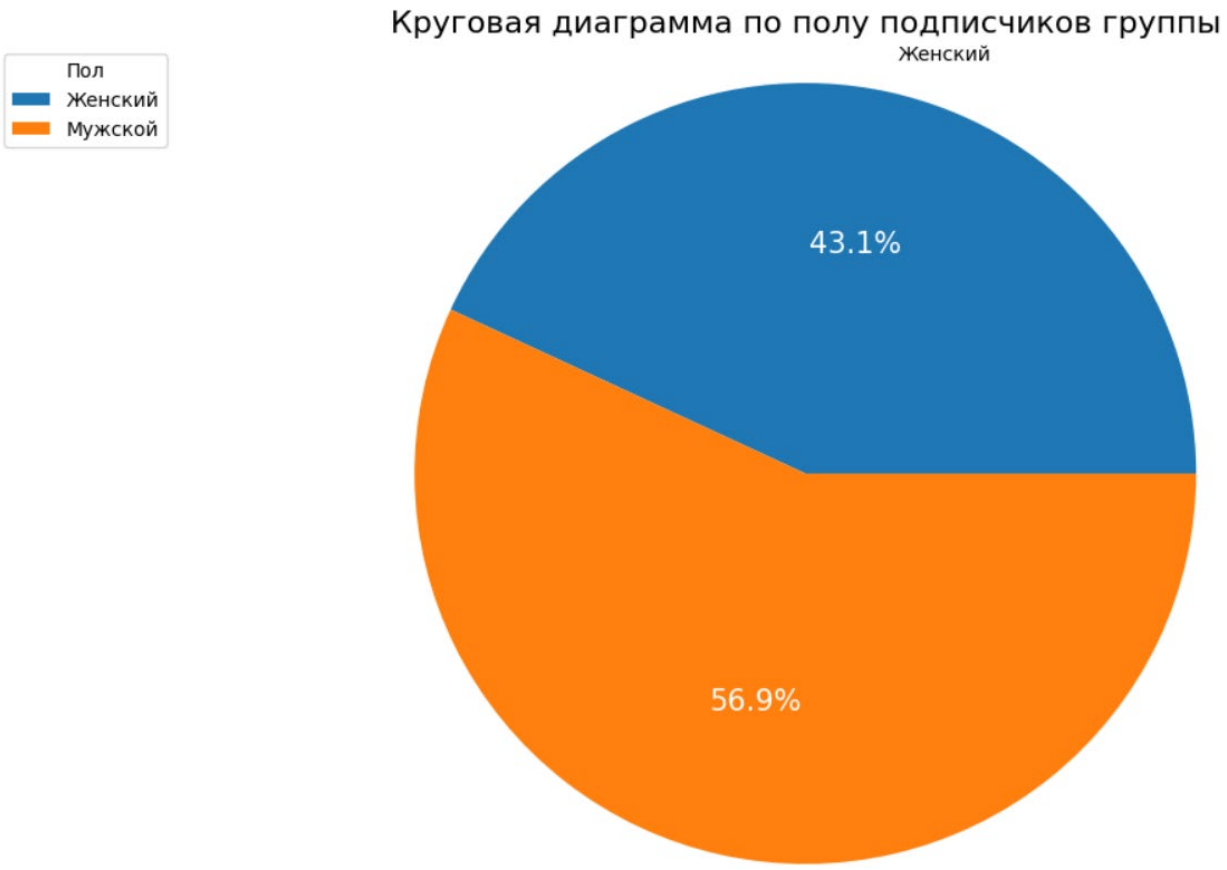
```
DrawPieByAge = df.loc[:,['age','count']]
DrawPieByAge = DrawPieByAge.groupby(['age']).sum()
DrawPieByAge = DrawPieByAge.sort_values(by='count', ascending=False)
sizes = DrawPieByAge['count']
labels = DrawPieByAge.index

plt.figure(figsize=(15, 8))
patches, texts, autotexts = plt.pie(
    sizes,
    labels=labels,
    startangle=0,
    autopct='%1.1f%%',
)

for text in autotexts:
    text.set_color('white')
    text.set_size(15)

plt.legend(patches, labels, title="Возраст", loc="upper left")
plt.title("Круговая Диаграмма Подписчиков Групп по возрасту", fontsize=16)
plt.axis('equal')
plt.show()
```



```
DrawBarByUniversity = df.loc[:, ['university_name','count']]
DrawBarByUniversity = DrawBarByUniversity.groupby(['university_name']).sum()
DrawBarByUniversity = DrawBarByUniversity.sort_values(by='count', ascending=False)
#DrawBarByUniversity
plt.figure(figsize=(16,9))
ax = sns.barplot(data = DrawBarByUniversity, x = 'count', y= 'university_name')
ax.set_xticks(np.arange(0, 19, 1))
ax.set(xlabel='Количество Человек', ylabel='Название Учебного Заведния', title='Количество Людей по Вузам Подписчики Группы')
```




```
DrawPieBySex = df.loc[:, ['sex','count']]
DrawPieBySex = DrawPieBySex.groupby(by='sex').sum()

sizes = DrawPieBySex['count']
labels = ['Женский','Мужской']

plt.figure(figsize=(15, 8))
patches, texts, autotexts = plt.pie(
    sizes,
    labels=labels,
    startangle=0,
    autopct='%1.1f%%',
)

for text in autotexts:
    text.set_color('white')
    text.set_size(15)

plt.legend(patches, labels, title="Пол", loc="upper left")
plt.title("Круговая диаграмма по полу подписчиков группы", fontsize=16)
plt.axis('equal')
plt.show()
```



2) вывести диаграммы, в которых высота столбиков пропорциональна количеству комментариев, которые оставили а) люди определенных возрастов б) выпускники/студенты определенных вузов.

```
def post_request(url, data):
    response = requests.post(url, data)
    return response.json()
def get_wall_posts(owner_id, access_token):
    url = 'https://api.vk.com/method/wall.get'
    data = {
        'owner_id': owner_id,
        'count': 100,
        'access_token': access_token,
        'v': '5.131',
        'fields': 'id'
    }
    return post_request(url, data)

def get_comments(owner_id, post_id, access_token):
    url = 'https://api.vk.com/method/wall.getComments'
    data = {
        'owner_id': owner_id,
        'post_id': post_id,
        'access_token': access_token,
        'v': '5.131',
        'count': 200,
        'extended': 1,
        'fields': 'bdate, education',
    }
    return post_request(url, data)['response']['profiles']
```

```
def collect_data(owner_id, access_token):
    profiles = []

    posts_data = get_wall_posts(owner_id, access_token)

    if 'response' in posts_data:
        posts = posts_data['response']['items']

        for post in posts:
            post_id = post['id']

            profiles.append(get_comments(owner_id, post_id, access_token))


    return profiles


owner_id = -62298989

users = collect_data(owner_id, ACCESS_TOKEN)
```

```
PersonsDataset = pd.DataFrame.from_dict(users[0])
for person in users[1:]:
    dftemp = pd.DataFrame.from_dict(person)
    PersonsDataset = pd.concat([PersonsDataset, dftemp], ignore_index=True)
PersonsDataset = PersonsDataset[PersonsDataset['is_closed'] == False]
PersonsDataset = PersonsDataset[['bdate', 'university_name']]
```

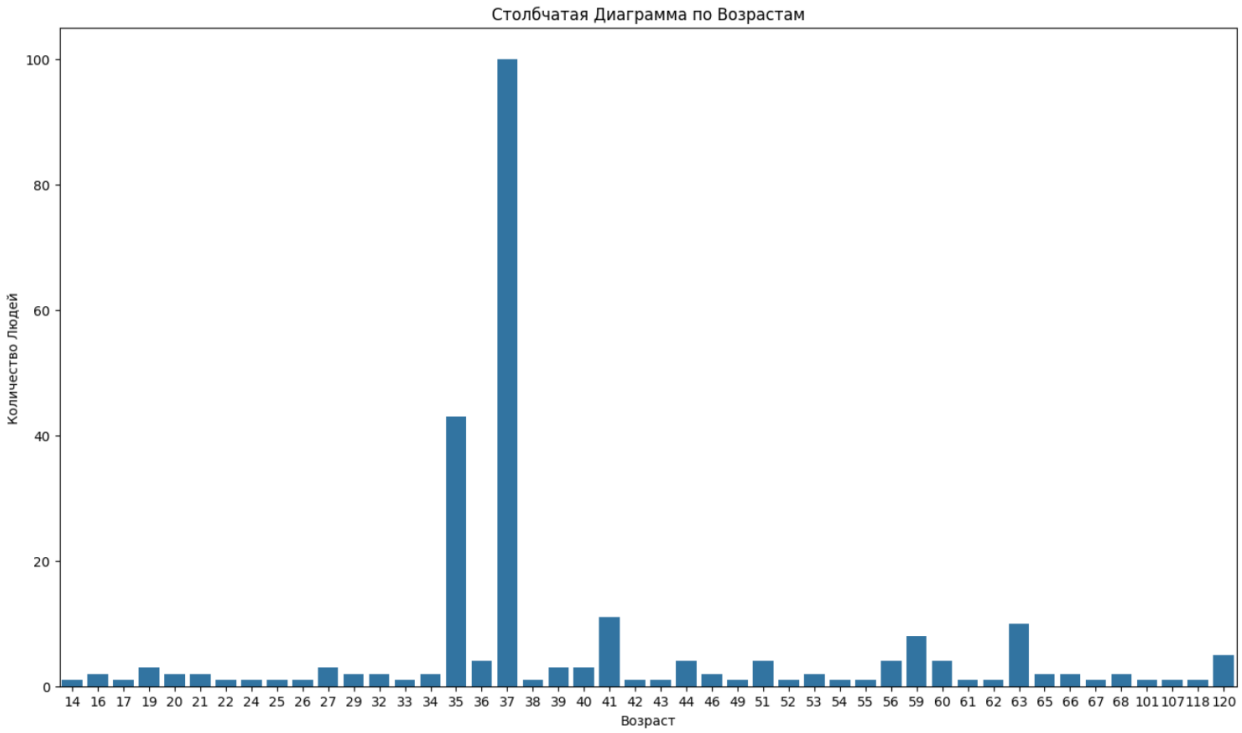
```
UniversityNames = PersonsDataset.university_name.dropna()
BDate = PersonsDataset.bdate.dropna()
```

```
BDate = BDate[BDate.str.contains(r'\d{4}')]
BDate = pd.to_datetime(BDate, format='%d.%m.%Y')
current_date = datetime.now()
BDateAge = current_date.year - BDate.dt.year

BDateAge -= ((current_date.month < BDate.dt.month) |
             ((current_date.month == BDate.dt.month) &
              (current_date.day < BDate.dt.day))).astype(int)
```

+ Code +

```
BDateAge = BDateAge.to_frame(name='Age')
BDateAge['count'] = 1
BDateAge = BDateAge.groupby(['Age']).sum()
```

```
plt.figure(figsize = (16,9))
ax = sns.barplot(data=BDateAge,x='Age', y = 'count')
ax.set(xlabel='Возраст', ylabel='Количество Людей', title='Столбчатая Диаграмма по Возрастам')
```

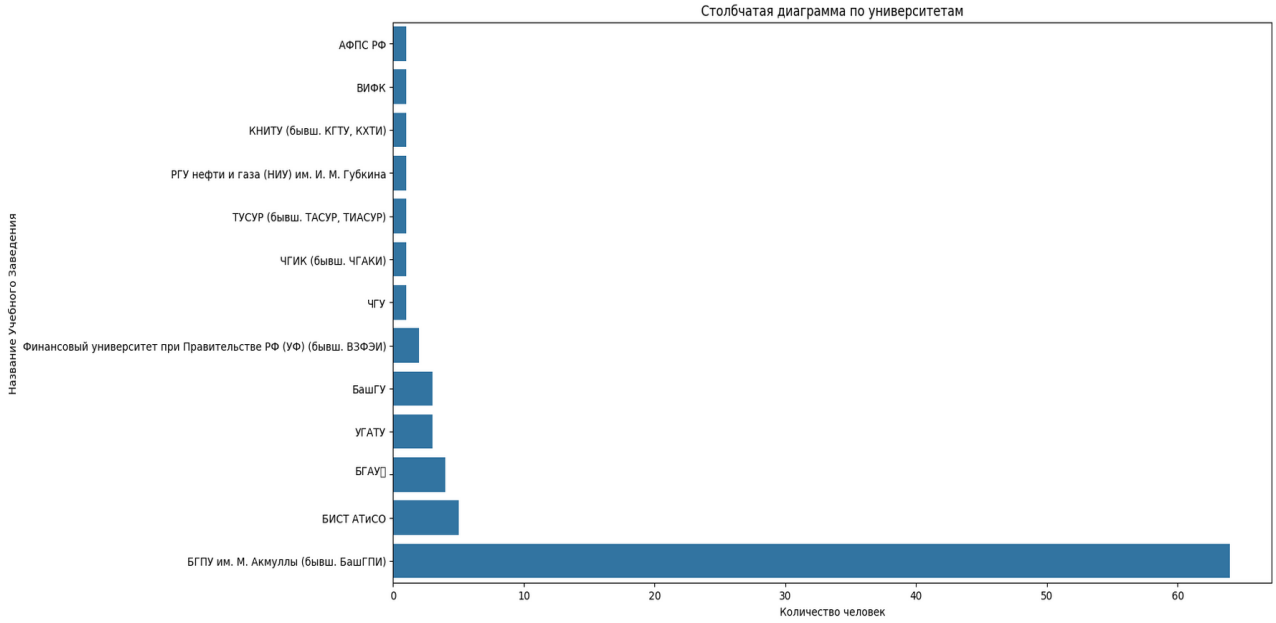



```
UniversityNames = UniversityNames.to_frame(name='name')
UniversityNames['count'] = 1
UniversityNames = UniversityNames.groupby(['name']).sum()
UniversityNames.sort_values(['count'],inplace=True,asc=False)
```

```
UniversityNames = UniversityNames[UniversityNames.index !='']
```

+ Code + Text

```
plt.figure(figsize = (16,9))
ax = sns.barplot(data=UniversityNames,x='count', y = 'name')
ax.set(xlabel='Количество человек', ylabel='Название Учебного Заведения', title='Столбчатая диаграмма по университетам')
```



Лабораторная работа №3

Задание: Средствами Python выяснить, в каких российских городах с населением свыше n (n=50000/100000) человек 1) в настоящее время температура воздуха ниже 0/идет дождь/идет снег/облачно. 2) три дня подряд, включая сегодняшний, температура воздуха ниже 0/идет дождь/идет снег/облачно.

Использовать открытые погодные библиотеки (например, Openweathermap). Уделить внимание скорости выполнения запроса.

```
import requests
import pandas as pd
from datetime import datetime, timedelta
from google.colab import drive
drive.mount('/content/gdrive', force_remount=False) #ones


API_KEY = 'kek1a'

def get_weather_data(lat, lon):
    url = f"https://api.openweathermap.org/data/2.5/weather?lat={lat}&lon={lon}&appid={API_KEY}&units=metric"
    response = requests.get(url)
    return response.json()

def get_forecast(lat, lon):
    url = f"https://api.openweathermap.org/data/2.5/forecast?lat={lat}&lon={lon}&cnt=17&appid={API_KEY}&units=metric"
    response = requests.get(url)
    return response.json()
```

```
def checkWeather(temperature, state):
  return temperature < 0 or state == 'Rain' or state == 'Snow' or state == 'Clouds'
```

```
namedataset = 'ru'

df = pd.read_csv(f'gdrive/My Drive/{namedataset}.csv',sep=',')
```

```
n = 100000

df = df[df['population'] > n]
```

```
df = df.loc[:, ['city','lat','lng']]
```

```
dfw = pd.DataFrame(columns=['Temperature', 'Weather']) #cloudly, rainy, snowy
```

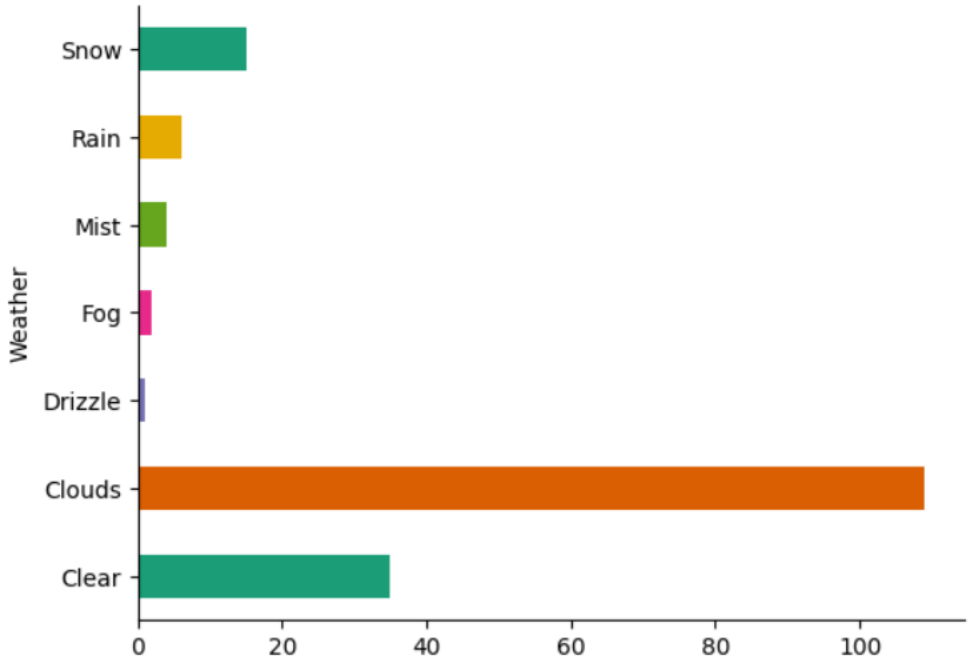
```
import time

results = pd.DataFrame(columns=['City','Temperature', 'Weather']) #W
allWeather = pd.DataFrame(columns=['City','Temperature', 'Weather'])

for index, city in df.iterrows():

  if (index == 59 or index == 119): { time.sleep(61) }

  weather_data = get_weather_data(city.lat, city.lng)
  temp = weather_data['main']['temp']
  state = weather_data['weather'][0]['main']

  if (checkWeather(temp, state)):
    results.loc[len(results)] = [city.city , temp, state]
  allWeather.loc[len(allWeather)] = [city.city , temp, state]


results
```

	City	Temperature	Weather
0	Moscow	-2.46	Clouds
1	Saint Petersburg	4.46	Clouds
2	Novosibirsk	-6.40	Clear
3	Yekaterinburg	-0.19	Clear
4	Kazan	-2.73	Clear
...	...	...	...
158	Berdsk	-9.55	Clear
159	Elista	2.13	Rain
160	Noginsk	-0.14	Clouds
161	Novokuybyshevsk	1.23	Clouds
162	Zheleznogorsk	0.53	Clouds

```
allWeather
```

	City	Temperature	Weather
0	Moscow	-2.46	Clouds
1	Saint Petersburg	4.46	Clouds
2	Novosibirsk	-6.40	Clear
3	Yekaterinburg	-0.19	Clear
4	Kazan	-2.73	Clear
...	...	...	...
167	Elista	2.13	Rain
168	Noginsk	-0.14	Clouds
169	Novokuybyshevsk	1.23	Clouds
170	Zheleznogorsk	0.53	Clouds
171	Zelënodol'sk	0.05	Clear

```
from matplotlib import pyplot as plt
import seaborn as sns
allWeather.groupby('Weather').size().plot(kind='barh', color=sns.palettes.mpl_palette('Dark2'))
plt.gca().spines[['top', 'right',]].set_visible(False)
```




```
time.sleep(61)
Day3 = pd.DataFrame(columns=['City','TempDay1', 'WeatDay1','TempDay2', 'WeatDay2','TempDay3', 'WeatDay3']) #Where clo
allWeather = pd.DataFrame(columns=['City','Temperature', 'Weather'])

for index, city in df.iterrows():
  temps = []
  states = []

  if (index == 59 or index == 119): { time.sleep(61) }


  weather_data = get_forecast(city.lat, city.lng)
  for i in range(0, 17, 8):
    temps.append(weather_data['list'][i]['main']['temp'])
    states.append(weather_data['list'][i]['weather'][0]['main'])

  if (checkWeather(temps[0], states[0]) and checkWeather(temps[1], states[1]) and checkWeather(temps[2], states[2])):
    Day3.loc[len(Day3)] = [city.city, temps[0], states[0], temps[1], states[1], temps[2], states[2]]


Day3
```

	City	TempDay1	WeatDay1	TempDay2	WeatDay2	TempDay3	WeatDay3
0	Moscow	-2.46	Clouds	-0.65	Snow	0.34	Snow
1	Saint Petersburg	4.46	Clouds	3.16	Clouds	4.56	Clouds
2	Novosibirsk	-6.40	Clear	-10.88	Clear	-5.17	Snow
3	Yekaterinburg	-0.19	Clear	-5.15	Clear	-7.22	Clouds
4	Kazan	-2.73	Clouds	0.57	Clouds	-0.98	Clouds
...	...	...	...	...	...	...	...
157	Elista	2.13	Rain	3.82	Rain	3.94	Clouds
158	Noginsk	-1.20	Clouds	-1.16	Snow	0.31	Clouds
159	Novokuybyshevsk	0.28	Clouds	-0.07	Clouds	0.10	Clouds
160	Zheleznogorsk	0.34	Clouds	0.23	Clouds	0.74	Clouds
161	Zelënodol'sk	-0.68	Clouds	0.35	Clouds	-1.93	Clouds

Лабораторная работа №4

Задание:

1) Используя статистические данных о последних n сезонах игры в футбольной Лиге 1 (Испания), найти фамилию судьи, при котором Лион чаще всего проигрывал Марселю.
2) Найти топ-10 команд (по количеству очков за все сезоны).
3) Найти количество голов, которые забили команды в прошлом сезоне (2023-2024) после 90-й минуты.
4) Найти самую молодую команду в сезоне 2023-2024 (среднее арифметическое возрастов всех игроков). Учитывать только игроков, выходивших на поле (идеально, если еще и количество отыгранных ими минут).

1)

```
import numpy as np
import pandas as pd

from google.colab import drive
drive.mount('/content/gdrive') #ones

namedataset = 'ligue1__all_years_dataset(2)'

dfgen = pd.read_csv(f'gdrive/My Drive/{namedataset}.csv',sep=',')

dfgen2 = pd.read_csv(f'gdrive/My Drive/ligue_1_2023_2024_stats.csv',sep=',')
```

```
df = dfgen[["Home","Away","Score", "Referee"]]
df = df[(df['Home'] == "Lyon")&(df['Away'] == "Marseille")|(df['Away'] == "Lyon")&(df['Home'] == "Marseille")]
df
```

	Home	Away	Score	Referee
22	Marseille	Lyon	1–1	Laurent Duhamel
218	Lyon	Marseille	1–0	Pierluigi Collina
565	Marseille	Lyon	1–4	Bertrand Layec

```
df.dropna(inplace=True)
df[['HomeScore', 'AwayScore']] = df['Score'].str.split('–', expand=True)
df['HomeScore'] = pd.to_numeric(df['HomeScore'])
df['AwayScore'] = pd.to_numeric(df['AwayScore'])
df.drop(columns=['Score'], inplace=True)
df = df[((df['Home'] == "Lyon") & (df['HomeScore'] < df['AwayScore'])) | ((df['Away'] == "Lyon") & (df['HomeScore'] > df['AwayScore']))]
df['count'] = 1
df = df.groupby('Referee')[['count']].sum()
df = df[df['count'] == df['count'].max()]
df
```

Referee	count
François Letexier	2
Fredy Fautrel	2

2)

```
df = dfgen[["Home","Away","Score"]]
df[['HomeScore', 'AwayScore']] = df['Score'].str.split('–', expand=True)
df['HomeScore'] = pd.to_numeric(df['HomeScore'])
df['AwayScore'] = pd.to_numeric(df['AwayScore'])
df.dropna(inplace=True)
df
```

	Home	Away	Score	HomeScore	AwayScore
0	Guingamp	Lyon	3–3	3.0	3.0
1	Troyes	Monaco	0–4	0.0	4.0
2	Montpellier	Rennes	1–0	1.0	0.0
3	Bastia	Lens	1–1	1.0	1.0

```
home_scores = df[['Home', 'HomeScore']].rename(columns={'Home': 'Team', 'HomeScore': 'Score'})
away_scores = df[['Away', 'AwayScore']].rename(columns={'Away': 'Team', 'AwayScore': 'Score'})

df = pd.concat([home_scores, away_scores])
df = df.groupby('Team')['Score'].sum()
df = df.sort_values(ascending=False)
df.head(10)
```

Team	Score
Paris S-G	1581.0
Lyon	1447.0
Marseille	1257.0
Monaco	1165.0
Lille	1150.0
Rennes	1088.0
Nice	966.0
Bordeaux	959.0
Montpellier	822.0
Saint-Étienne	817.0

4)

```
dfgen2
dfgen2['Min'] = dfgen2['Min'].str.replace(',', '').astype(int)
```

```
df = dfgen2[dfgen2['Min'] > 30]
df = df.groupby('Team')['Age'].mean()
df = df.sort_values()
df.head(1)
```

Team	Age
Toulouse	22.730769

ЗАКЛЮЧЕНИЕ

Учебная практика была проведена в период с 02.09.2024 г. по 31.12.2024 г. (объем 2 нед.)

В установленный период практики были выполнены лабораторные работы

№1 ”Основы программирования на Python.”, №2 “ Парсинг vk.”, №3 “Получение погодных данных”,№4 “Машинное обучение и нейронные сети на Python”.

Отчёт по практике был подготовлен и сдан в установленные сроки.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Лути М. Изучаем Python, 4-е издание. Пер. с англ. - СПб.: Символ- Плюс, 2011. - 1280 с.
2. Златопольский Д.М. Основы программирования на языке Python. - М.: ДМК Пресс, 2017. - 284 с.
3. Гэддис Т. Начинаем программировать на Python. 4-е изд.: Пер. с англ. - СПб.: БХВ-Петербург, 2019. - 768 с.
4. Федоров, Д. Ю. Программирование на языке высокого уровня Python : учебное пособие для прикладного бакалавриата / Д. Ю. Федоров. - 2-е изд., (Бакалавр. перераб. и доп.- Москва: Издательство Юрайт, 2019. - 161 с. - Прикладной курс). ISBN 978-5-534-10971-9. Текст: электронный // ЭБС Юрайт [сайт]. - URL: https://urait.ru/bcode/437489 (дата обращения: 13.02.2020).
5. Прохоренок Н.А. Самое необходимое. - СПб.: БХВ-Петербург, 2011. - 416 с.
6. Доусон М. Программируем на Python. - СПб.: Питер, 2014. - 416 с.