



МИНОБРНАУКИ РОССИИ  
Федеральное государственное бюджетное образовательное учреждение  
высшего образования  
«МИРЭА – Российский технологический университет»  
**РТУ МИРЭА**

---

Институт информационных технологий (ИИТ)  
Кафедра корпоративных информационных систем (КИС)

**ОТЧЁТ ПО УЧЕБНОЙ ПРАКТИКЕ**  
**Ознакомительная практика**

приказ Университета о направлении на практику от 09 февраля 2022 г. № 1103-С

Отчет представлен к  
рассмотрению:

Студент группы ИКБО-08-21

«08» июня 2022

Полонкоев Б.А.

(подпись и расшифровка подписи)

Отчет утвержден.

Допущен к защите:

Руководитель практики  
от кафедры

«08» июня 2022

Габриелян Г.А.

(подпись и расшифровка подписи)

Москва 2022 г.



МИНОБРНАУКИ РОССИИ  
Федеральное государственное бюджетное образовательное учреждение  
высшего образования  
«МИРЭА – Российский технологический университет»  
**РТУ МИРЭА**

Институт информационных технологий (ИИТ)  
Кафедра корпоративных информационных систем (КИС)

**ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ НА УЧЕБНУЮ ПРАКТИКУ**

**Ознакомительная практика**

**Студенту 1 курса учебной группы ИКБО-08-21**

**Полонкоеву Берсу Адамовичу**

Место и время практики: РТУ МИРЭА кафедра КИС, с 09 февраля 2022 г. по 31 мая 2022 г.

Должность на практике: студент

**1. СОДЕРЖАНИЕ ПРАКТИКИ:**

1.1. Изучить: методические материалы по курсу, программную документацию языка программирования и стандартных библиотек, API применяемых сервисов.

1.2. Практически выполнить:

- анализ выданных в курсе практических задач;
- поиск, интерпретацию, анализ и ранжирование информации из изученных источников и баз данных, необходимых для решения практических задач;
- решение задач по темам: работа с коллекциями, работа со строками, работа с файлами, основы ООП, стандартные библиотеки, графический интерфейс, использование сторонних API;
- представление результатов выполнения практических задач в требуемом формате (условие, алгоритм, решение задачи, тестирование).

1.3. Ознакомиться: со средствами социального взаимодействия и командной работы в профессиональной среде, с учебно-методическим пособием по ознакомительной практике.

**2. ДОПОЛНИТЕЛЬНОЕ ЗАДАНИЕ:** нет

**3. ОРГАНИЗАЦИОННО-МЕТОДИЧЕСКИЕ УКАЗАНИЯ:** Положение о практической подготовке обучающихся, осваивающих основные профессиональные образовательные программы ВО; учебно-методическое пособие по ознакомительной практике.

Руководитель практики от  
кафедры

  
Подпись

(Габриелян Г.А.)

«09» февраля 2022 г.

Задание получил

  
Подпись

(Полонкоев Б.А.)

«09» февраля 2022 г.

**СОГЛАСОВАНО:**

Заведующий кафедрой:

  
Подпись

(Андрианова Е.Г.)

«09» февраля 2022 г.

**Проведенные инструктажи:**

Охрана труда:

Инструктирующий

  
Подпись

«09» февраля 2022 г.

Трохаченкова Н.Н., старший преподаватель кафедры КИС

Инструктируемый

  
Подпись

Полонкоев Б.А.

Техника безопасности:

Инструктирующий

  
Подпись

«09» февраля 2022 г.

Трохаченкова Н.Н., старший преподаватель кафедры КИС

Инструктируемый

  
Подпись

Полонкоев Б.А.

Пожарная безопасность:

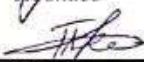
Инструктирующий

  
Подпись

«09» февраля 2022 г.

Трохаченкова Н.Н., старший преподаватель кафедры КИС

Инструктируемый

  
Подпись

Полонкоев Б.А.

С правилами внутреннего распорядка ознакомлен:

  
Подпись

«09» февраля 2022 г.

Полонкоев Б.А.





МИНОБРНАУКИ РОССИИ  
Федеральное государственное бюджетное образовательное учреждение  
высшего образования  
«МИРЭА – Российский технологический университет»  
**РТУ МИРЭА**

**РАБОЧИЙ ГРАФИК ПРОВЕДЕНИЯ  
ОЗНАКОМИТЕЛЬНОЙ ПРАКТИКИ**

Студента Полонкостей Б.А. 1 курса группы ИКБО-08-21 очной формы обучения, обучающегося по направлению подготовки 09.03.04 Программная инженерия

Неделя	Сроки выполнения	Этап	Отметка о выполнении
1	09.02 – 15.02	Подготовительный этап, включающий в себя организационное собрание (Вводная лекция о порядке организации и прохождения производственной практики, инструктаж по технике безопасности, получение задания на практику)	<i>Габриелян</i>
2-3	17.02 – 01.03	Выполнение заданий по теме «Основные конструкции языка, коллекции» (анализ задач; поиск информации для решения; решение задач; представление результатов в требуемом формате)	<i>Габриелян</i>
4-5	02.03 – 14.03	Выполнение заданий по теме «Строки, работа с файлами» (анализ задач; поиск информации для решения; решение задач; представление результатов в требуемом формате)	<i>Габриелян</i>
6-7	16.03 – 28.03	Выполнение заданий по теме «Основы ООП» (анализ задачи; поиск информации для решения; решение задачи; представление результатов в требуемом формате)	<i>Габриелян</i>
8-9	30.03 – 11.04	Выполнение заданий по теме «Стандартные библиотеки языка программирования» (анализ задач; поиск информации для решения; решение задач; представление результатов в требуемом формате)	<i>Габриелян</i>
10-11	13.04 – 25.04	Выполнение заданий по теме «Графический интерфейс и внешние библиотеки» (анализ задачи; поиск информации для решения; решение задачи; представление результатов в требуемом формате)	<i>Габриелян</i>
12-14	27.04 – 16.05	Использование сторонних API для создания приложений, (анализ задачи; поиск информации для решения; решение задачи; представление результатов в требуемом формате)	<i>Габриелян</i>
15-16	18.05 – 31.05	Оформление материалов отчета в полном соответствии с требованиями на оформление письменных учебных работ студентов	<i>Габриелян</i>

Руководитель практики от кафедры

*Габриелян* /Габриелян Г.А. /

Обучающийся

*Полонкостей* /Полонкостей Б.А./

Согласовано:

Заведующий кафедрой

*Андреанова* /Андреанова Е.Г., к.т.н., доцент/

## 1) Тема «Основные конструкции языка, коллекции»

### Задача 1

Условие задачи: Написать функцию `strong_number(number)`, которая определяет является ли число сильным. Число сильное, если сумма факториалов цифр числа равна самому числу.

Решение задачи:

```
def fact(n):
    factorial = 1
    while n > 1:
        factorial *= n
        n -= 1
    return(factorial)

def strong_number(number):
    number = str(number)
    s = 0
    for i in number:
        s += fact(int(i))
    if s == int(number):
        return True
    return False
```

Тестирование:

№	Тест	Ожидаемое значение	Полученное значение
1	1	True	True
2	2	True	True
3	7	False	False
4	93	False	False
5	145	True	True

### Задача 2

Условие задачи: Написать функцию `goldbach(n)` для иллюстрации гипотезы Гольдбаха: каждое четное целое число больше 2 может быть записано как сумма двух простых чисел. Если аргумент нечетный, верните пустой список. Для четных аргументов необходимо вернуть список с двумя простыми числами, сумма которых равна `n`. Два простых числа должны быть самыми дальними (с наибольшей разницей). Первое простое число должно быть наименьшим.

### Решение задачи:

```
def goldbach(n):
    def IsSimple(a):
        if a < 2:
            return False
        for i in range(2, int(a ** 0.5 + 1)):
            if a % i == 0:
                return False
        else:
            return True
    if n % 2 == 0:
        for k in range(2, n):
            if IsSimple(k) and IsSimple(n-k):
                return [k,n-k]
    return []
```

### Тестирование:

№	Тест	Ожидаемое значение	Полученное значение
1	15	[]	[]
2	4	[2, 2]	[2, 2]
3	10	[3, 7]	[3, 7]
4	24	[5,19]	[5,19]
5	100	[3, 97]	[3, 97]
6	1234	[3, 1231]	[3, 1231]

### **Задача 3**

Условие задачи: Написать функцию `strong_enough(earthquake, age)`, которая вычисляет достаточно ли безопасное здание, чтобы выдержать землетрясение. Здание рухнет, если сила землетрясения будет больше, чем сила здания. Earthquake – список, состоящий из списка ударных волн. Вычисление силы землетрясения для  $[[5,3,7], [3,3,1], [4,1,2]] \rightarrow ((5 + 3 + 7) * (3 + 3 + 1) * (4 + 1 + 2)) = 735$ . Прочность нового здания 1000, при этом это значение уменьшается на 1% каждый год

### Решение задачи:

```
def strong_enough(earthquake, age):
    s = sum(earthquake[0]) * sum(earthquake[1]) *
    sum(earthquake[2])
    a = True
    n = 1000
    for i in range(age):
        n *= 0.99
        if s >= n:
            a = False
            break

    return a
```

### Тестирование:

№	Тест	Ожидаемое значение	Полученное значение
1	[[2,3,1],[3,1,1],[1,1,2]], 2	True	True
2	[[5,8,7],[3,3,1],[4,1,2]], 2	True	True
3	[[5,8,7],[3,3,1],[4,1,2]], 3	False	False

#### Задача 4

Условие задачи: Написать функцию `palindrome`, которая для заданного числа `num` возвращает список всех числовых палиндромов, содержащихся в каждом номере. Массив должен быть отсортирован в порядке возрастания, а любые дубликаты должны быть удалены.

Пример:

`palindrome(34322122) => [22, 212, 343, 22122]`

Решение задачи:

```
def palindrome(a):
    a = str(a)
    pall = list()
    for i in range(len(a)-1):
        for n in range(len(a)):
            number = a[i:n+1]
            nlen = len(number)
            if nlen % 2 == 0 and nlen > 1 and int(number) != 0 and \
                (number[:int(nlen/2)] == number[int(nlen/2):][::-1]):
                pall.append(int(number))
            elif nlen > 1 and int(number) != 0 and \
                (number[:int(nlen/2)] == number[int(nlen/2)+1:][::-1]):
                pall.append(int(number))
    return sorted(set(pall))
```

Тестирование:

№	Тест	Ожидаемое значение	Полученное значение
1	1551	[55, 1551]	[55, 1551]
2	221122	[11, 22, 2112, 221122]	[11, 22, 2112, 221122]
3	10015885	[88, 1001, 5885]	[88, 1001, 5885]
4	13598	[]	[]



## Задача 5

Условие задачи: Создать список (каталог мобильных приложений), состоящий из словарей (приложение). Словари должны содержать как минимум 5 полей (например, номер, название, рейтинг...). В список добавить хотя бы 10 словарей.

Конструкция вида:

```
apps = [{"id" : 123456, "title" : "Google Play", "rating" : 4.9,...} , {...}, {...}, ...]
```

Реализовать функции:

- вывода информации о всех приложениях;
- вывода информации о приложении по введенному с клавиатуры номеру;
- вывода количества приложений, с оценкой выше введенного;
- обновлении всей информации о приложении по введенному номеру;
- удалении приложения по номеру.

Решение задачи:

```

apps = [
{"id" : 156433, "author" : "Rovio","name": "Angry Birds", "rating": 5, "price": 0},
{"id" : 254738, "author" : "Apple","name": "Apple Music", "rating": 6.5, "price": 100},
{"id" : 193567, "author" : "Desmos","name": "Calculator" , "rating": 10, "price": 179},
{"id" : 294926, "author" : "Tinkoff","name": "Tinkoff", "rating": 9.6, "price": 349},
{"id" : 254296, "author" : "Facebook","name": "WhatsApp", "rating": 8 , "price": 0 },
{"id" : 907394, "author" : "Yandex","name": "Maps", "rating": 7.5, "price": 15},
{"id" : 154926, "author" : "Yandex","name": "Music", "rating": 6.3, "price": 0},
{"id" : 265836, "author" : "Google","name": "Chrome", "rating": 4.2, "price": 167},
{"id" : 272946, "author" : "MailGroup","name": "Mail", "rating": 3, "price": 200},
{"id" : 295737, "author" : "Lichess.org","name": "Lichess", "rating": 10, "price": 199}]

def infoOutput(apps):
    for perDict in apps:
        for key in perDict:
            print(key,":",perDict[key],";", end = " ")
        print()

def infoOutput_byNum(apps):
    for i in range(len(apps)):
        print(i+1,"\t",(apps[i]["id"]),'\t',apps[i]["name"])

    appNum = int(input("Введите id приложения, информацию о котором нужно вывести.\n"))

    for i in range(len(apps)):
        if apps[i]["id"] == appNum:
            for key in apps[i]: print(key + " -- " + str(apps[i][key]))
            return 0
    print("Нет приложения с подходящим номером")

def quantByRate(apps):
    quant = 0
    minRate = int(input("Введите минимальный рейтинг"))
    for perDict in apps:
        if perDict["rating"] > minRate:
            quant+=1
    print(quant)

def changeInfo(apps):
    for i in range(len(apps)):
        print(i+1,"\t",(apps[i]["id"]),'\t',apps[i]["name"])

    appNum = int(input("Введите id приложения, информацию о котором нужно поменять.\n"))

    for i in range(len(apps)):
        if apps[i]["id"] == appNum:
            for key in apps[i]:
                print("Старый",key,"--",apps[i][key])
                apps[i][key] = input("Введите новое значение для " + str(key) + "\t")
            print(apps[i])
            return 0
    print("Нет приложения с таким id")

```

```

def delByNum(apps):
    for i in range(len(apps)):
        print(i+1, "\t", apps[i]["id"], "\t", apps[i]["name"])

    appNum = int(input("Введите id приложения, которое нужно удалить "))
    for i in range(len(apps)):
        if apps[i]["id"] == appNum:
            print("Удаление программы", apps[i]["name"])
            apps.pop(i)
            for i in range(len(apps)):
                print(i+1, "\t", (apps[i]["name"]))
            return 0
    print("Нет приложения с таким id")
def numChoice():
    num = int(input("\nВведите номер функции: "))
    if num == 0: return 0
    elif num == 1: infoOutput(apps)
    elif num == 2: infoOutput_byNum(apps)
    elif num == 3: quantByRate(apps)
    elif num == 4: changeInfo(apps)
    elif num == 5: delByNum(apps)
    else: print("Неверное значение номера. Попробуйте еще раз \n\n")
    numChoice()

print(" 0. Завершение работы программы \n 1. Вывод информации о всех приложениях \n 2.
Вывод информации о приложении по введенному с клавиатуры номеру \n 3. Вывод количества
приложений, с оценкой выше введенного \n 4. Обновление всей информации о приложении по
введенному номеру \n 5. Удаление приложения по номеру\n")

numChoice()

```

### Тестирование:

#### Тестирование функции infoOutput() – Вывод информации о приложениях:

```

0. Завершение работы программы
1. Вывод информации о всех приложениях
2. Вывод информации о приложении по введенному с клавиатуры номеру
3. Вывод количества приложений, с оценкой выше введенного
4. Обновление всей информации о приложении по введенному номеру
5. Удаление приложения по номеру

Введите номер функции: 1
id : 156433 ; author : Rovio ; name : Angry Birds ; rating : 5 ; price : 0 ;
id : 254738 ; author : Apple ; name : Apple Music ; rating : 6.5 ; price : 100 ;
id : 193567 ; author : Desmos ; name : Calculator ; rating : 10 ; price : 179 ;
id : 294926 ; author : Tinkoff ; name : Tinkoff ; rating : 9.6 ; price : 349 ;
id : 254296 ; author : Facebook ; name : WhatsApp ; rating : 8 ; price : 0 ;
id : 907394 ; author : Yandex ; name : Maps ; rating : 7.5 ; price : 15 ;
id : 154926 ; author : Yandex ; name : Music ; rating : 6.3 ; price : 0 ;
id : 265836 ; author : Google ; name : Chrome ; rating : 4.2 ; price : 167 ;
id : 272946 ; author : MailGroup ; name : Mail ; rating : 3 ; price : 200 ;
id : 295737 ; author : Lichess.org ; name : Lichess ; rating : 10 ; price : 199 ;

Введите номер функции: █

```

Рисунок 1 - Тестирование функции вывода информации о приложениях

Тестирование функции `infoOutput_byNum()` - вывода информации о приложении по

номеру:

```
Введите номер функции: 2
1      156433      Angry Birds
2      254738      Apple Music
3      193567      Calculator
4      294926      Tinkoff
5      254296      WhatsApp
6      907394      Maps
7      154926      Music
8      265836      Chrome
9      272946      Mail
10     295737      Lichess
Введите id приложения, информацию о котором нужно вывести.
254296
id -- 254296
author -- Facebook
name -- WhatsApp
rating -- 8
price -- 0
Введите номер функции: █
```

Рисунок 2 - Тестирование функции вывода информации о приложении по номеру

Тестирование функции `quantByRate()` - вывода количества приложений, с рейтингом выше заданного:

```
Введите номер функции: 3
Введите минимальный рейтинг6
7
```

Рисунок 3 - Тестирование функции вывода количества приложений, с рейтингом выше заданного

Тестирование функции changeInfo() - обновлении всей информации о приложении по введенному ID:

```
Введите номер функции: 4
1      156433      Angry Birds
2      254738      Apple Music
3      193567      Calculator
4      294926      Tinkoff
5      254296      WhatsApp
6      907394      Maps
7      154926      Music
8      265836      Chrome
9      272946      Mail
10     295737      Lichess
Введите id приложения, информацию о котором нужно поменять.
254296
Старый id -- 254296
Введите новое значение для id      4567893
Старый author -- Facebook
Введите новое значение для author      Meta
Старый name -- WhatsApp
Введите новое значение для name Instagram
Старый rating -- 8
Введите новое значение для rating      9.5
Старый price -- 0
Введите новое значение для price      0
{'id': '4567893', 'author': 'Meta', 'name': 'Instagram', 'rating': '9.5', 'price': '0'}
Введите номер функции: █
```

Рисунок 4 - Тестирование функции обновления всей информации о приложении по введенному ID



```

Введите номер функции: 1
id : 156433 ; author : Rovio ; name : Angry Birds ; rating : 5 ; price : 0 ;
id : 254738 ; author : Apple ; name : Apple Music ; rating : 6.5 ; price : 100 ;
id : 193567 ; author : Desmos ; name : Calculator ; rating : 10 ; price : 179 ;
id : 294926 ; author : Tinkoff ; name : Tinkoff ; rating : 9.6 ; price : 349 ;
id : 4567893 ; author : Meta ; name : Instagram ; rating : 9.5 ; price : 0 ;
id : 907394 ; author : Yandex ; name : Maps ; rating : 7.5 ; price : 15 ;
id : 154926 ; author : Yandex ; name : Music ; rating : 6.3 ; price : 0 ;
id : 265836 ; author : Google ; name : Chrome ; rating : 4.2 ; price : 167 ;
id : 272946 ; author : MailGroup ; name : Mail ; rating : 3 ; price : 200 ;
id : 295737 ; author : Lichess.org ; name : Lichess ; rating : 10 ; price : 199 ;

Введите номер функции: █

```

Рисунок 5 – Список приложений после изменения информации о приложении

Тестирование функции delByNum() - удаления приложения по ID:

```

Введите номер функции: 5
1      156433      Angry Birds
2      254738      Apple Music
3      193567      Calculator
4      294926      Tinkoff
5      4567893     Instagram
6      907394      Maps
7      154926      Music
8      265836      Chrome
9      272946      Mail
10     295737      Lichess
Введите id приложения, которое нужно удалить 295737
Удаление программы Lichess
1      Angry Birds
2      Apple Music
3      Calculator
4      Tinkoff
5      Instagram
6      Maps
7      Music
8      Chrome
9      Mail

Введите номер функции: █

```

Рисунок 6 - Тестирование функции удаления приложения по ID

## 2) Тема «Строки, работа с файлами»

### Задача 1

Условие задачи: Написать функцию `break_camel_case`, которая разбивает слова написанные CamelCase, используя в качестве разделителя пробел

Решение задачи:

```
def break_camel_case(s):
    s1 = s[0]
    for i in range(1, len(s)):
        if s[i].istitle() and s[i-1] != " ":
            s1 += " " + s[i]
        else:
            s1 += s[i]
    return s1
```

Тестирование:

№	Тест	Ожидаемое значение	Полученное значение
1	"BreakCamelCase"	"Break Camel Case"	"Break Camel Case"
2	"helloWorld"	"hello World"	"hello World"
3	"helloWorld BreakCamelCase"	"hello World Break Camel Case"	"hello World Break Camel Case"

### Задача 2

Условие задачи: Написать функцию `sum_of_fractions`, которая получает вещественное число и возвращает строку - сумму слагаемых числа в виде дробей. Между слагаемыми поставить символ `+`, все отделить пробелами

Решение задачи:

```

def sum_of_fractions(num):
    a=""
    ss=""

    num=str(num)
    for i in range(0,len(num)):
        if num[i]==".":
            c=i
            if num[:c]!="0":
                a=a+num[:c]+" + "
            for i1 in range(c+1,len(num)):
                if num[i1]=="0":
                    continue
                b=10**(i1-c)
                if (len(num)-1)!=i1:
                    ss=num[i1]+"/"+str(b)+" + "
                else:
                    ss = num[i1] + "/" + str(b)
                a=a+ss
            break
    return a

```

#### Тестирование:

№	Тест	Ожидаемое значение	Полученное значение
1	1.24	'1 + 2/10 + 4/100'	'1 + 2/10 + 4/100'
2	7.304	'7 + 3/10 + 4/1000'	'7 + 3/10 + 4/1000'
3	0.04	'4/100'	'4/100'

#### **Задача 4**

Условие задачи: Создать txt-файл, вставить туда любую англоязычную статью из Википедии.

Реализовать одну функцию, которая выполняет следующие операции:

- прочитать файл построчно;

- непустые строки добавить в список;
- удалить из каждой строки все цифры, знаки препинания, скобки, кавычки и т.д. (остаются латинские буквы и пробелы);
- объединить все строки из списка в одну, используя метод join и пробел, как разделитель;
- создать словарь вида {“слово”: количество, “слово”: количество, ... } для подсчета количества разных слов,
  - где ключом будет уникальное слово, а значением - количество;
- вывести в порядке убывания 10 наиболее популярных слов, используя форматирование (вывод примерно следующего вида: “ 1 place --- sun --- 15 times \n ”);
- заменить все эти слова в строке на слово “PYTHON”;
- создать новый txt-файл;
- записать строку в файл, разбивая на строки, при этом на каждой строке записывать не более 100 символов при этом не делить слова.

#### Решение задачи:

```
def wiki_function():
    file=open('C:\\Users\\polon\\Desktop\\pract\\B\\a.txt')
    k=0
    s=[]
    for row in file:
        if row!="\n":
            s.append(row)
    k=0
    for row in s:
        for i in row:
            if i.isalpha()==False and i!=" ":
                row=row.replace(i,"")
        s[k]=row
        k+=1
    s_str = " ".join(s)
    s.clear()
    s=s_str.lower().split()
    a={}
    for i in range(0,len(s)):
        a.setdefault(s[i],0)
        if a[s[i]]==0:
            for i1 in range(0,len(s)):
                if s[i]==s[i1]:
                    a[s[i]]+=1
    sor=list(a.values())
    sor.sort()
    k=0
    top=""
```

```

for i in range(len(sor)-1,-1,-1):
    if sor[i]==sor[i-1]:
        continue
    for key in a:
        if a[key]==sor[i]:
            k+=1
            keyb=key[0].upper()+key[1:]
            for word in range(0,len(s)):
                if s[word]==key or s[word]==keyb:
                    s[word]="PYTHON"
            top+='{0} place --- {1} --- {2}
times\n'.format(k,key,sor[i])
            if (k == 10):
                break
        if (k==10):
            break
print("\n\n\n\n10 популярных слов:\n\n\n", top)
print("\n\n\n-----")

s_str=" ".join(s)
my_file=open("C:\\Users\\polon\\Desktop\\pract\\B\\b.txt","w")
ss=""
k=0
for i in range(0,len(s_str)):

    if i-k==99:
        if s_str[i]==" " or s_str[i+1]==" " or len(s_str)==i-1:

            my_file.write(s_str[k:i+1]+"\n")
            if s_str[i]==" ":
                k=i+1
            elif s_str[i+1]==" ":
                k=i+2

        else:
            for i1 in range(i,k,-1):
                if s_str[i1]==" ":

                    my_file.write(s_str[k:i1+1] +"\n")
                    k = i1+1
                    break
            elif len(s_str)-1==i:
                my_file.write(s_str[k:i + 1])
return 1

wiki_function()

```

### Тестирование:

#### Статья в википедии:

The arts are a very wide range of human practices of creative expression, storytelling and cultural participation. They encompass multiple diverse and plural



modes of thinking, doing and being, in an extremely broad range of media. Both highly dynamic and a characteristically constant feature of human life, they have developed into innovative, stylized and sometimes intricate forms. This is often achieved through sustained and deliberate study, training and/or theorizing within a particular tradition, across generations and even between civilizations. The arts are a vehicle through which human beings cultivate distinct social, cultural and individual identities, while transmitting values, impressions, judgments, ideas, visions, spiritual meanings, patterns of life and experiences across time and space.

Prominent examples of the arts include architecture, visual arts (including ceramics, drawing, filmmaking, painting, photography, and sculpting), literary arts (including fiction, drama, poetry, and prose), performing arts (including dance, music, and theatre), textiles and fashion, folk art and handicraft, oral storytelling, conceptual and installation art, criticism, and culinary arts (including cooking, chocolate making and winemaking). They can employ skill and imagination to produce objects, performances, convey insights and experiences, and construct new environments and spaces.

The arts can refer to common, popular or everyday practices as well as more sophisticated and systematic, or institutionalized ones. They can be discrete and self-contained, or combine and interweave with other art forms, such as the combination of artwork with the written word in comics. They can also develop or contribute to some particular aspect of a more complex art form, as in cinematography.

By definition, the arts themselves are open to being continually re-defined. The practice of modern art, for example, is a testament to the shifting boundaries, improvisation and experimentation, reflexive nature, and self-criticism or questioning that art and its conditions of production, reception, and possibility can undergo.

As both a means of developing capacities of attention and sensitivity, and as ends in themselves, the arts can simultaneously be a form of response to the world, and a way that our responses, and what we deem worthwhile goals or pursuits, are transformed. From prehistoric cave paintings, to ancient and contemporary forms of ritual, to modern-day films, art has served to register, embody and preserve our ever shifting relationships to each other and to the world.

Вывод программы:

```
10 популярных слов:
1 place --- and --- 36 times
2 place --- of --- 15 times
3 place --- the --- 12 times
4 place --- to --- 11 times
5 place --- arts --- 10 times
6 place --- a --- 9 times
7 place --- art --- 7 times
8 place --- can --- 6 times
9 place --- or --- 6 times
10 place --- as --- 6 times
```

Рисунок 7 - Вывод программы

PYTHON PYTHON are PYTHON very wide range PYTHON human practices PYTHON creative expression storytelling PYTHON cultural participation they encompass multiple diverse PYTHON plural modes PYTHON thinking doing PYTHON being in an extremely broad range PYTHON media both highly dynamic PYTHON PYTHON characteristically constant feature PYTHON human life they have developed into innovative stylized PYTHON sometimes intricate forms this is often achieved through sustained PYTHON deliberate study training andor theorizing within PYTHON particular tradition across generations PYTHON even between civilizations PYTHON PYTHON are PYTHON vehicle through which human beings cultivate distinct social cultural PYTHON individual identities while transmitting values impressions judgments ideas visions spiritual meanings patterns PYTHON life PYTHON experiences across time PYTHON space prominent examples PYTHON PYTHON PYTHON include architecture visual PYTHON including ceramics drawing filmmaking painting photography PYTHON sculpting literary PYTHON including fiction drama poetry PYTHON prose performing PYTHON including dance music PYTHON theatre textiles PYTHON fashion folk PYTHON PYTHON handicraft oral storytelling conceptual PYTHON installation PYTHON criticism PYTHON culinary PYTHON including cooking chocolate making PYTHON winemaking they PYTHON employ skill PYTHON imagination PYTHON produce objects performances convey insights PYTHON experiences PYTHON construct new environments PYTHON spaces PYTHON PYTHON PYTHON refer PYTHON common popular PYTHON everyday practices PYTHON well PYTHON more sophisticated PYTHON systematic PYTHON institutionalized ones they PYTHON be discrete PYTHON selfcontained PYTHON combine PYTHON interweave with other PYTHON forms such PYTHON PYTHON combination PYTHON artwork with PYTHON written word in comics they PYTHON also develop PYTHON contribute PYTHON some particular aspect PYTHON PYTHON more complex PYTHON form PYTHON in cinematography by definition PYTHON PYTHON themselves are open PYTHON being continually redefined PYTHON practice PYTHON modern PYTHON for example is PYTHON testament PYTHON PYTHON shifting boundaries improvisation PYTHON experimentation reflexive nature PYTHON selfcriticism PYTHON questioning that PYTHON PYTHON its conditions PYTHON production reception PYTHON possibility PYTHON undergo PYTHON both PYTHON means PYTHON developing capacities PYTHON attention PYTHON sensitivity PYTHON PYTHON ends in themselves PYTHON PYTHON PYTHON simultaneously be PYTHON form PYTHON response PYTHON PYTHON world PYTHON PYTHON way that our responses PYTHON what we deem worthwhile goals PYTHON pursuits are transformed from prehistoric cave paintings PYTHON ancient PYTHON contemporary forms PYTHON ritual PYTHON modernday films PYTHON has served PYTHON register embody PYTHON preserve our ever shifting relationships PYTHON each other PYTHON PYTHON PYTHON world

Рисунок 8 - Содержимое файла после работы программы

### 3) Тема «Основы объектно-ориентированного программирования»

Класс Person. Класс Person с полями имя, фамилия, возраст. `__init__` - конструктор класса.

```
def __init__(self, first_name , last_name, age):
    self.first_name = first_name
    self.last_name = last_name
    self.age = age
    def show_info(self):
        a = "ФИ: " + self.last_name + ' ' + self.first_name + ";\nВозраст: "
+ str(self.age) + ";"
        print(a)
```

Класс Reader. Класс Reader наследуется от класса Person с полями имя, фамилия, возраст. `__init__` - конструктор класса. Новые поля: номер читательского билета (ticket\_num), читательский билет (ticket)

`__init__` - конструктор, с вызовом родительского конструктора.

`add_book(self,book_num,date)` – метод добавления книги в читательский билет.

`get_date_by_num(self, book_num)` – метод получения даты получения книги по ее номеру

`show_ticket(self)` – метод для форматированной печати всего читательского билета

`__str__(self)` – переопределение для преобразования в строку для печати основной информации

```

class Reader(Person):

    def __init__(self, first_name = "Арсений", last_name = "Гаврилов", age =
45, ticket_num = 173, ticket = {'12': '12.12.12', '123123': '13.13.13', \
    '213434': '21.21.21', '213423': '34.34.34', '131231': '54.54.54'}):
        super().__init__(first_name, last_name, age)
        self.ticket_num = ticket_num
        self.ticket = ticket

    def add_book(self, book_num, date):

        self.ticket.update({book_num:date})
        return self

    def get_date_by_num(self, book_num):
        return(f"Дата получения книги № {book_num}:
{self.ticket.get(str(book_num))}")

    def show_ticket(self):
        print("Номер книги: \t\t Дата получения:")
        items = self.ticket.items()
        for per_item in items:
            print(per_item[0], '\t\t\t', per_item[1])

    def __str__(self):
        a = f"ФИ {self.last_name} {self.first_name};\nВозраст:
{self.age};\nНомер читательского билета: {self.ticket_num};\n"
        return(a)

```

Класс Librarian. Класс Librarian - производный от Person. Новые поля: номер удостоверения (ident\_num) , должность (post) , график работы (словарь вида день недели: часы работы) (schedule)

`__init__` - конструктор, с вызовом родительского конструктора.

`change_post(self, new_post)` – функция изменения должности.

`set_schedule(self)` – добавления, удаления и изменения графика работы.

`__str__(self)` – переопределение для преобразования в строку для печати основной информации



```

class Librarian(Person):

    def __init__(self, first_name = "Андрей", last_name = "Лавров", age =
27, ident_num = 123, post = "Директор", schedule = {"Пн" : "9-21", "Вт": "",
"Ср": "9-21", "Чт": "", "Пт": "9-21", "Сб": "", "Вс": "9-19"}):
        super().__init__(first_name, last_name, age)
        self.ident_num = ident_num
        self.post = post
        self.schedule = schedule

    def change_post(self, new_post):
        self.post = new_post
        return self

    def set_schedule(self):

        a = list(map(str, input("Введите сокращения дней недели, расписание
которых хотите поменять. Пн, Вт, Ср и т.д.\n\n").title().split()))

        print("Введите часы работы в формате (час начала)-(час окончания) 9-
21, для удаления оставьте поле пустым.\n\n")
        for i in a:
            self.schedule[i] = input(str(i)+" : ")

        for i in self.schedule:
            print(i, self.schedule[i])

    def __str__(self):
        a = f"ФИ: {self.last_name} {self.first_name};\nВозраст:
{self.age};\nНомер удостоверения: {self.ident_num};\nДолжность:
{self.post};\n"
        return(a)

```

Класс Library. Поля: название библиотеки (name), адрес (address), список читателей (список экземпляров класса Reader) (readers\_list), список библиотекарей (список экземпляров класса Librarian) (librarians\_list)

def \_\_init\_\_ - конструктор.

def \_\_add\_\_(self, other) - Переопределение операции сложения для добавления читателя. Other – читатель.

def \_\_sub\_\_(self, s) – Переопределение операции вычитания для удаления элемента меню. Other – читатель.

def \_\_str\_\_(self) - Переопределение метода преобразования в строку для печати меню.

def \_\_getitem\_\_(self, k) - Переопределение метода получения по индексу для получения читателя по индексу. k – индекс.

def \_\_setitem\_\_(self, k, value): Переопределение метода изменения по индексу для изменения читателя по индексу. k – индекс, value – то на что изменяем.

def \_\_delitem\_\_(self, key) - Переопределение метода удаления по индексу для удаления читателя по индексу. k – индекс.

info\_to\_txt(self) – функция создания txt-файла и записи всей информации в него

```

from Librarian import Librarian
from Reader import Reader

class Library:
    def __init__(self, name = "Boston", adress = "25 avenue", readers_list =
[Reader(),Reader()], librarians_list = [Librarian(), Librarian()]):
        self.name = name
        self.adress = adress
        self.readers_list = readers_list
        self.librarians_list = librarians_list

    def __str__(self):
        a = f"Название библиотеки: {self.name}\nАдрес библиотеки:
{self.adress} \n\nЧитатели"
        for per_read in self.readers_list:
            a += f"{str(per_read)}\n\n"
        a += f"\n\nБиблиотекари:\n"
        for per_lib in self.librarians_list:
            a += f"{str(per_lib)}\n\n"
        return self

    def __getitem__(self, key):
        if key == 0:
            for per_lib in range(len(self.librarians_list)):
                print(per_lib+1, str(per_lib))
        else:
            print(str(self.readers_list[key-1]))
        return self

    def __delitem__(self, key):
        if not isinstance(key, int):
            raise TypeError("Индекс должен быть целым числом")
        if key == 0:
            self.librarians_list.clear()
        else:
            self.readers_list.pop(key-1)
        return self

    def __setitem__(self, key, value):
        if not isinstance(key, int) or key < 0:
            raise TypeError("Индекс должен быть целым неотрицательным
числом")
        if key == 0:
            self.librarians_list = value
        else:
            self.readers_list[key-1].first_name = value[0]
            self.readers_list[key-1].last_name = value[1]
            self.readers_list[key-1].age = value[2]
            self.readers_list[key-1].ticket_num = value[3]
            self.readers_list[key-1].ticket = value[4]
        return self

    def __add__(self, other):
        if not isinstance(other, Reader):
            raise ArithmeticError("Правый операнд должен быть класса
Reader")
        self.readers_list.append(other)

```

```

def __sub__(self, other):
    if not isinstance(other, Reader):
        raise ArithmeticError("Правый операнд должен быть класса
Reader")
    a = self.readers_list.index(other)
    self.readers_list.pop(a)
    return self

def info_to_txt(self):
    file = open('C\\LibraryTXT.txt', 'w', encoding = 'UTF-8')

    file.write("Название библиотеки: " + self.name + "\nАдрес
библиотеки: " + self.adress + "\n\nЧитатели:\n\n")

    for per_read in self.readers_list:
        file.write(per_read.show_info())
        file.write("Читательский билет: ")
        for tick in per_read.ticket:
            file.write("№" + tick + ': ' + per_read.ticket[tick] + '; ')
        file.write('\n\n')

    file.write("\n\nБиблиотекари:\n\n")

    for per_lib in self.librarians_list:
        file.write(per_lib.show_info())
        file.write("График работы: ")
        for sch in per_lib.schedule:
            file.write(sch + ': ' + per_lib.schedule[sch]+'; ')
        file.write('\n\n')

```

#### 4) Тема «Стандартные библиотеки языка программирования»

Модуль `math` - содержит наиболее применяемые математические функции и константы. Используемые методы:

- `math.log(X, [base])` - логарифм `X` по основанию `base`.
- `math.pow(X, Y)` – возведение числа `X` в степень `Y`.
- `math.sqrt(X)` - квадратный корень из `X`
- `math.fabs(X)` - модуль `X`
- `math.pi` -  $\pi = 3,1415926\dots$
- `math.e` -  $e = 2,718281\dots$
- `math.radians(X)` - конвертирует градусы в радианы.
- `math.cos(X)` - косинус `X` (`X` указывается в радианах).
- `math.sin(X)` - синус `X` (`X` указывается в радианах).

Модуль `re` для регулярных выражений в Python. Используемые методы:

- `re.sub(pattern, repl, string, max=0)` - Этот метод заменяет все вхождения `pattern` в `string` на `repl`, если не указано на `max`. Он возвращает измененную строку.
- `re.search(pattern, string, flags=0)`. Функция `re.search` возвращает объект `match` если совпадение найдено, и `None`, когда нет совпадений.

Модуль `os` - функции для работы с операционной системой, не зависящие от используемой операционной системы. Используемые функции:



- `os.mkdir()` – функция для создания папки. В скобках указывается название папки
- `os.replace()` - используется для перемещения файлов или каталогов. Первым аргументом указывается текущий путь к файлу/папке, вторым куда вы хотите переместить файл/папку.
- `Os.listdir()` - возвращает список, содержащий имена файлов и директорий в каталоге. В скобках указывается путь к папке.

Модуль `time` – модуль для работы со временем. Используемые функции:

- `time.time()` - время, выраженное в секундах с начала эпохи.

## Задача 2

Условие задачи: Реализовать две функции, вычисляющие математические формулы. Параметры формул являются аргументами функций.

$$T = \frac{\arctg \sqrt{|x|} + x^2}{\ln 2x + e^{|-x-5|}} + 3a - 0,2$$


---

$$k = \sin^2 6a + 8 \operatorname{tg} b^3 - \frac{5}{abx} \cdot 3,82^a$$

Рисунок 9 - Формулы

## Решение задачи:

```
from math import *
def first_form(a,x):
    return (atan(sqrt(abs(x)))+pow(x,2))/(log(2*x) + pow(e, abs(-x-5)))+3*a - 0.2

def second_form(a,b,x):
    return pow(sin(6*a),2) + 8 * tan(pow(b,3)) - 5/(a*b*x) * pow(3.82,a)
```

Тестирование для первой функции:

№	Тест	Ожидаемое значение	Полученное значение
1	2, 2	5.804512958860187	5.804512958860187
2	6, 5	17.801187095961996	17.801187095961996

Тестирование для второй функции:

№	Тест	Ожидаемое значение	Полученное значение
1	1, 10, 3	11.203999599588835	11.203999599588835
2	2, 14, 2	42.65507716735134	42.65507716735134

**Задача 3**

Условие задачи: Показать выполненное тестирование.

– Задача. Никнейм

Email состоит из трех частей: никнейм, доменное имя, суффикс доменной зоны. Извлеките никнейм пользователя, имя домена и суффикс из данных email адресов.

Решение задачи:

```
import re
f = open('C:\\Users\\polon\\Desktop\\pract\\D\\email_adresses.txt')
regArr = []
for s in f:
    a = list(re.split(r'([a-z0-9_\\.-]+)@([a-z0-9_\\.-]+)\\.([a-z\\.]{2,6})', s))
    a.remove('')
    if a.count('\\n') > 0: a.remove('\\n')
    print(a)
    regArr.append(a)
print(regArr)
```

Тестирование:

№	Тест	Ожидаемое значение	Полученное значение
1	galga@ing.ru ant.lyadov@gmail.com ignat123@mail.ru on.emirea@mirea.ru	['galga', 'ing', 'ru'] ['ant.lyadov', 'gmail', 'com'] ['ignat123', 'mail', 'ru'] ['on.emirea', 'mirea', 'ru', '']	['galga', 'ing', 'ru'] ['ant.lyadov', 'gmail', 'com'] ['ignat123', 'mail', 'ru'] ['on.emirea', 'mirea', 'ru', '']

## Задача 4

### Условие задачи:

- Собрать в папке файлы «task\_\*\*\*\*.py» – все ранее решенные задачи из тем А, В.
- Написать функцию, которая создаст папку «Ознакомительная папка» с двумя подпапками («тема А», «тема В»), переместит все файлы в правильные подпапки.
- Написать функцию, которая получает адрес ранее созданной папки «Ознакомительная папка» и выполнит обход всех подпапок и:
  - чтение всех «task\_\*\*\*\*.py» файлов, нахождение в тексте названия функции и параметров
  - программный запуск и выполнение данных файлов, подсчет времени выполнения

### Решение задачи:

```
from os import walk, makedirs, chdir, getcwd, scandir, replace, system, startfile
from time import *
import subprocess
import sys
from re import *
from time import *

chdir('C:\\Users\\polon\\Desktop\\pract\\D')
path = getcwd()
def moveTasks(path):
    makedirs(path + '\\Ознакомительная практика\\Тема А', True)
    makedirs(path + '\\Ознакомительная практика\\Тема В', True)
    for item in scandir(path + '\\allTasksD'):
        if search(r'task_{1})(\\d)+\\.py', item.name) != None and item.is_file:
            replace(path + f'\\allTasksD\\{item.name}', path + f'\\Ознакомительная практика\\Тема {item.name[5]}\\{item.name}')

def startTasks(path):
    pract = walk(f'{path}\\Ознакомительная практика')
    for mainfolder in pract:
        if len(mainfolder[-2]) == 2:
            for folder in mainfolder[-2]:
                print(f'folder {folder}')
                theme_fold = walk(f'{path}\\Ознакомительная практика\\{folder}')
                for i in theme_fold:
```

```

        print(f'\n>>> script {task}')
        f =
open('C:\\Users\\polon\\Desktop\\pract\\D\\Ознакомительная
практика\\'+folder+'\\'+task, 'r')
        s = f.readline
        for s in f:
            if s[:3] == 'def':
                func = s[4:-2]
                break
        f.close()
        print(f'>>> >>> function {func}')
        start = time()
        with
open('C:\\Users\\polon\\Desktop\\pract\\D\\Ознакомительная
практика\\'+folder+'\\'+task, 'r', encoding='UTF8') as m:
            result = subprocess.run([sys.executable, "-c",
m.read()], capture_output=True, encoding='ISO-8859-1')
            print(f'>>> >>> output {result.stdout}', end='')
            print(f'>>> >>> time: {time() - start}')

```

## 5) Тема «Графический интерфейс и внешние библиотеки»

Tkinter - это графическая библиотека, позволяющая создавать программы с оконным интерфейсом. Используемые функции:

- Tk является базовым классом любого Tkinter приложения. При создании объекта этого класса запускается интерпретатор tcl/tk и создаётся базовое окно приложения.
- Title() – в скобках указывается название приложения.
- ttk.Notebook — еще один новый виджет из модуля ttk. Он позволяет добавлять разные виды отображения приложения в одном окне, предлагая после этого выбрать желаемый с помощью клика по соответствующей вкладке.
- Tkinter.Frame() используется для группировки виджетов в окне
- Виджет Combobox предназначен для отображения списка значений, их выбора или изменения пользователем.
- Label - это виджет, предназначенный для отображения какой-либо надписи без возможности редактирования пользователем.
- grid() – используется для указания расположения
- Entry() - это виджет, позволяющий пользователю ввести одну строку текста.
- Text() - это виджет, который позволяет пользователю ввести любое количество текста.
- Виджет Button - самая обыкновенная кнопка, которая используется в тысячах программ
- Combobox.get() – получение значения combobox.
- text.get() – получение значение введенное в поле Text()

Модуль urllib.request определяет функции и классы, которые помогают открывать URL-адреса (в основном HTTP).

Модуль xml - содержит встроенные XML инструменты для парсинга, к которым вы можете получить доступ.

Модуль Matplotlib - это основная библиотека для построения графиков.

Модуль datetime - модуль для работы с датой и временем в python.

### Решение задачи:

Создание оформления и запись в словарь валют, полученных с сайта ЦБ:

```
from tkinter import *
from tkinter.ttk import *
from datetime import *
from matplotlib import *
import matplotlib
import matplotlib.pyplot as plt
import graph_defs as gr
from getCurrency import *
from dateutil.relativedelta import relativedelta
import locale

locale.setlocale(locale.LC_TIME, "ru_RU")
window = Tk()

# window.maxsize(1000,500)
# window.minsize(500,100)
window.title("Конвертер валют")
tab_control = Notebook(window, width=700, height=300)
tab1 = Frame(tab_control)
tab2 = Frame(tab_control)
tab_control.add(tab1, text="Калькулятор валют")
tab_control.add(tab2, text="Динамика курса")

#Parsing xml
m = str(datetime.today())[10].replace('-', '/')
d = m[-2:]+m[4:8]+m[:4]
url = 'http://www.cbr.ru/scripts/XML_daily.asp?date_req='+d
curr_list = []
curr_dict = get_currencies_dictionary(get_data(url))
curr_dict['Рубль'] = 1
print(curr_dict)
for key in curr_dict: curr_list.append(key)

first_curr = Combobox(tab1, width=30, values=curr_list)

first_curr.grid(column=0, row=0, padx=10, pady=10) #First combobox
first_curr.set(first_curr['value'][0])

second_curr = Combobox(tab1, values=curr_list, width=30)
second_curr.grid(column=0, row=1, padx=10, pady=10) #Second combobox
second_curr.set(second_curr['value'][0])

quant_entry = Entry(tab1)
quant_entry.grid(column=1, row=0, padx=10, pady=10) #Entry fill

fin_curr = Label(tab1,width=17)
fin_curr.grid(column=1,row=1, padx=10, pady=10) #Currency fin
```

```

#Button
def cha():
    fin_curr.config(text=curr_dict[first_curr.get()] /
curr_dict[second_curr.get()] * float(quant_entry.get().replace(',','.')))
    btn_curr = Button(tab1, text="Конвертировать", command=cha)
    btn_curr.grid(column=2,row=0)

#PART II - Currency dynamic

Label(tab2, text="Валюта").grid(column=0,row=0,padx=10)          #Valute Label
Label(tab2, text="Период").grid(column=1,row=0,padx=10)          #Period Label
Label(tab2, text="Выбор периода").grid(column=2,row=0,padx=10)  #Period
choice Label

#Curr combobox
curr_combo = Combobox(tab2, width=30, values=curr_list)
curr_combo.grid(column=0, row=1)
curr_combo.set(curr_combo['value'][0])

## WEEK COMBOBOX
m = datetime.today()-timedelta(days=datetime.today().weekday()) #Week`s
Monday
s = m+timedelta(days=6)                                          #Week`s
Sunday
w_args = []
for i in range(4):
    m1=str(m-timedelta(days=i*7))[:10].replace('-','.')
    s1=str(s-timedelta(days=i*7))[:10].replace('-','.')
    m1=m1[-2:]+m1[4:8]+m1[:4]
    s1=s1[-2:]+s1[4:8]+s1[:4]
    w_args.append(m1+'-'+s1)
week_combo = Combobox(tab2, values=w_args)
week_combo.set(week_combo['value'][0])

#####

## MONTH COMBOBOX
m_args = []
for i in range(4):m_args.append((datetime.today()-
relativedelta(months=i)).strftime('%B %Y'))
global month_combo
month_combo = Combobox(tab2, values=m_args)
month_combo.set(month_combo['value'][0])

#####

## QUARTER COMBOBOX
quarter_list=[]
for i in range(4):
    m = int(str(datetime.today() - relativedelta(months=3*i))[5:7])%12//3
    if m == 0:
        quarter_list.append("Зима "+(datetime.today() -
relativedelta(months=3*i)).strftime("%Y"))

```

```

        elif m == 2:
            quarter_list.append("Лето "+(datetime.today() -
            relativedelta(months=3*i)).strftime("%Y"))
        elif m == 3:
            quarter_list.append("Осень "+(datetime.today() -
            relativedelta(months=3*i)).strftime("%Y"))

quarter_combo = Combobox(tab2, values=quarter_list)
quarter_combo.set(quarter_combo['value'][0])

## YEAR COMBOBOX

year_list = []
for i in range(4):
    year_list.append(int((datetime.today()).strftime("%Y"))-i)
year_combo = Combobox(tab2, values=year_list)
year_combo.set(year_combo['value'][0])

#Radiobuttons
period = StringVar()
period.set("week")

def perCombo(value):
    week_combo.grid_forget()
    month_combo.grid_forget()
    quarter_combo.grid_forget()
    year_combo.grid_forget()
    if value == 1:
        week_combo.grid(column=2,row=1,padx=15,pady=5,sticky="WE")
    elif value == 2:
        month_combo.grid(column=2,row=2,padx=15,pady=5,sticky="WE")
    elif value == 3:
        quarter_combo.grid(column=2,row=3,padx=15,pady=5,sticky="WE")
    else:
        year_combo.grid(column=2,row=4,padx=15,pady=5,sticky="WE")

week_radio =
Radiobutton(tab2,text="Неделя",variable=period,value="week",command= lambda:
perCombo(1))
week_radio.grid(column=1,row=1,padx=15,pady=5)

month_radio =
Radiobutton(tab2,text="Месяц",variable=period,value="month",command= lambda:
perCombo(2))
month_radio.grid(column=1,row=2,padx=15,pady=5)

quarter_radio =
Radiobutton(tab2,text="Квартал",variable=period,value="quarter",command=
lambda: perCombo(3))
quarter_radio.grid(column=1,row=3,padx=15,pady=5)

year_radio =
Radiobutton(tab2,text="Год",variable=period,value="year",command= lambda:
perCombo(4))
year_radio.grid(column=1,row=4,padx=15,pady=5)

```



```

#Buid a graphic button
def build():
    matplotlib.use('TkAgg')
    fig = plt.figure()
    canvas = matplotlib.backends.backend_tkagg.FigureCanvasTkAgg(fig,
master=tab2)
    plot_widget = canvas.get_tk_widget()
    fig.clear()

    curr = curr_combo.get()
    if period.get() == "week":
        date_str = str(week_combo.get())
        gr.week_graph(date_str, curr)
    elif period.get() == "month":
        date_str = str(month_combo.get())
        gr.month_graph(date_str, curr)
    elif period.get() == "quarter":
        date_str = str(quarter_combo.get())
        gr.quarter_graph(date_str, curr)
    elif period.get() == "year":
        date_str = str(year_combo.get())
        gr.year_graph(date_str, curr)

    plt.grid()
    plot_widget.grid(column=3, row = 5)

graph_btn = Button(tab2,text="Построить график",command=build)
graph_btn.grid(column=0,row=4,sticky="WE")

tab_control.pack(expand=1, fill='both')
window.mainloop()

```

### Функции, получающие курс валют с сайта ЦБ:

```

import urllib.request
import xml.dom.minidom as minidom

def get_data(xml_url):
    try:
        web_file = urllib.request.urlopen(xml_url)
        return web_file.read()
    except:
        pass

def get_currencies_dictionary(xml_content):

    dom = minidom.parseString(xml_content)
    dom.normalize()

    elements = dom.getElementsByTagName("Valute")
    currency_dict = {}

    for node in elements:
        for child in node.childNodes:
            if child.nodeType == 1:

```

```

        if child.nodeType == 1:
            if child.tagName == 'Nominal':
                quant = float(child.firstChild.data.replace(',', ' '))
            if child.tagName == 'Value':
                if child.firstChild.nodeType == 3:
                    value = float(child.firstChild.data.replace(',', ' '))
            if child.tagName == 'Name':
                if child.firstChild.nodeType == 3:
                    char_code = child.firstChild.data
                currency_dict[char_code] = value
        return currency_dict

def get_ex_currency(curr_name, xml_content):
    dom = minidom.parseString(xml_content)
    dom.normalize()

    t = False

    elements = dom.getElementsByTagName("Valute")
    for node in elements:
        for child in node.childNodes:
            if child.nodeType == 1:
                if child.tagName == 'Nominal':
                    quant = float(child.firstChild.data.replace(',', ' '))
                if child.tagName == 'Name':
                    if child.firstChild.nodeType == 3:
                        if child.firstChild.data == curr_name:
                            t = True
                if child.tagName == 'Value' and t == True:
                    if child.firstChild.nodeType == 3:
                        return float(child.firstChild.data.replace(',', ' '))
    return None

```

### Функции, строящие графики:

```

import locale
locale.setlocale(locale.LC_TIME, "ru_RU")
import matplotlib.pyplot as plt

def week_graph(date_str, curr_name):

    nowDate = date(int(date_str[6:10]), int(date_str[3:5]),
int(date_str[:2]))
    x_list = []
    y_list = []
    url = 'http://www.cbr.ru/scripts/XML_daily.asp?date_req='
    for i in range(7):
        x_list.append((nowDate + timedelta(days=i)).strftime("%d.%m"))
        y_list.append(get_ex_currency(curr_name, get_data(url + (nowDate +
timedelta(days=i)).strftime("%d/%m/%Y"))))
    # print(x_list)
    # print(y_list)
    # print(url + (nowDate + timedelta(days=i)).strftime("%d/%m/%Y"))

```

```

def month_graph(date_str, curr_name):

    month_number = {#Словарь месяцев
        "Январь" : 1,
        "Февраль" : 2,
        "Март" : 3,
        "Апрель" : 4,
        "Май" : 5,
        "Июнь" : 6,
        "Июль" : 7,
        "Август" : 8,
        "Сентябрь" : 9,
        "Октябрь" : 10,
        "Ноябрь" : 11,
        "Декабрь" : 12}

    s = date_str.split(' ')
    month = month_number[s[0]]
    year = int(s[1])
    # print(month)
    # print(year)

    start_date = date(year,month,1)

    # print(start_date)
    url = 'http://www.cbr.ru/scripts/XML_daily.asp?date_req='
    x_list = []
    y_list = []

    while int(str(start_date)[5:7]) < (month+1)%12 and start_date <=
datetime.today().date():
        # print(start_date.strftime("%d/%m/%Y"))
        x_list.append(start_date.strftime("%d %b"))
        y_list.append(get_ex_currency(curr_name, get_data(url +
start_date.strftime("%d/%m/%Y"))))
        start_date += timedelta(days = 4)
    # print(x_list)
    # print(y_list)
    plt.plot(x_list,y_list)
def quarter_graph(date_str, curr_name):
    month_number = {
        "Осень" : 9,
        "Зима" : 12,
        "Весна" : 3,
        "Лето" : 6
    }

    x_list = []
    y_list = []
    url = 'http://www.cbr.ru/scripts/XML_daily.asp?date_req='

    s = date_str.split(' ')
    month = month_number[s[0]]
    year = int(s[1])
    # print(month)
    # print(year)

    start date = date(year,month,1)

```

```

for i in range(3):
    x_list.append(start_date.strftime("%d %b"))
    x_list.append((start_date + timedelta(days = 14)).strftime("%d %b"))

    y_list.append(get_ex_currency(curr_name, get_data(url +
start_date.strftime("%d/%m/%Y"))))
    y_list.append(get_ex_currency(curr_name, get_data(url + (start_date
+ timedelta(days = 14)).strftime("%d/%m/%Y"))))
    start_date+= relativedelta(months=1)
    # print(x_list)
    # print(y_list)
    plt.plot(x_list,y_list)

def year_graph(date_str, curr_name):
    start_date = date(int(date_str),1,1)

    x_list = []
    y_list = []
    url = 'http://www.cbr.ru/scripts/XML_daily.asp?date_req='

    while start_date.year <= int(date_str) and start_date <=
datetime.today().date():
        x_list.append(start_date.strftime("%b"))
        y_list.append(get_ex_currency(curr_name, get_data(url +
start_date.strftime("%d/%m/%Y"))))
        start_date+= relativedelta(months=1)
    plt.plot(x_list,y_list)

```

### Тестирование:

Рисунок 10 - Тестирование калькулятора валют

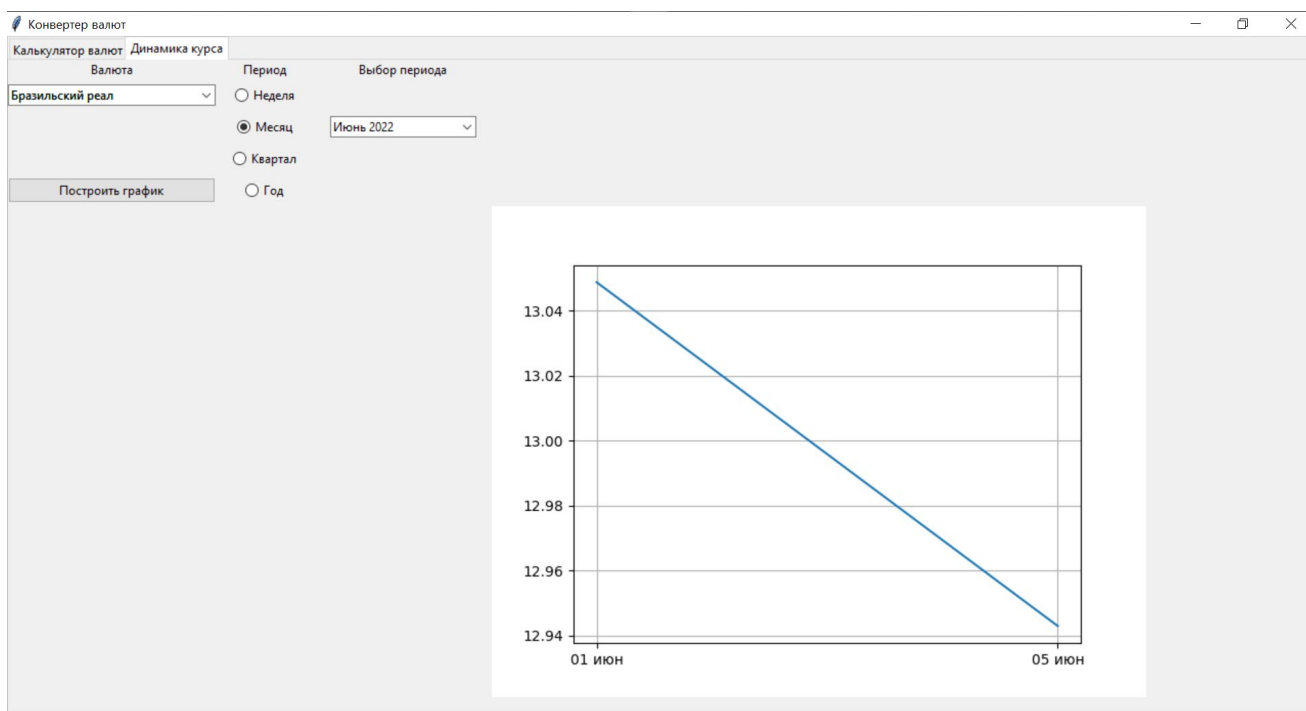


Рисунок 11 - Тестирование функции построения графиков

## 6) Использование сторонних API для создания приложений

Используемые модули:

- Модуль `openpyxl` - библиотека для чтения/записи форматов Office Open XML (файлов Excel 2010).
- Модуль `json` позволяет кодировать и декодировать данные в удобном формате.
- Модуль `Re` для регулярных выражений в Python.
- Модуль `requests`, используется для отправки всех видов HTTP-запросов
- Модуль `Beautiful Soup` для извлечения данных из файлов HTML и XML.
- Модуль `Matplotlib` - это основная библиотека для построения графиков.
- Модуль `datetime` - модуль для работы с датой и временем в python.
- Модуль `PIL` для работы с изображениями.
- Модуль `math` - содержит наиболее применяемые математические функции и константы.
- Модуль `vk_api` для создания скриптов для социальной сети Вконтакте

Ключевые участки кода:

Получение файлов с расписанием:

```
import openpyxl
import requests
import datetime

from bs4 import BeautifulSoup
from datetime import datetime as dt
import locale

locale.setlocale(locale.LC_TIME, "ru_RU")

def get_schedule_xlsx():
    #Getting links
    soup =
BeautifulSoup((requests.get('https://www.mirea.ru/schedule/')).text,
"html.parser")
    link_dom = soup.find(string = "Институт информационных технологий")\
```

```

        .find_parent("div")\
        .find_parent("div")\
        .find_all("a", class_='uk-link-toggle')
links = []
for link in link_dom:
    links.append(link['href'])
#Getting xlsx with links and writing into files
for x in links:
    if x.find('ИИТ_1') > 0:
        f = open("F\\first_course.xlsx", "wb") # открываем файл для
записи, в режиме wb
        resp = requests.get(x) # запрос по ссылке
        f.write(resp.content)
    elif x.find('ИИТ_2') > 0:
        f = open("F\\second_course.xlsx", "wb") # открываем файл для
записи, в режиме wb
        resp = requests.get(x) # запрос по ссылке
        f.write(resp.content)
    elif x.find('ИИТ_3') > 0:
        f = open("F\\third_course.xlsx", "wb") # открываем файл для
записи, в режиме wb
        resp = requests.get(x) # запрос по ссылке
        f.write(resp.content)

```

### Запись расписания в словарь:

```
import openpyxl

import re,json
book=openpyxl.load_workbook("file.xlsx")

sheet=book.active
num_cols=sheet.max_column
num_rows=sheet.max_row
print(num_rows,num_cols)
reg=r"([А-Я])([А-Я])Б0-([0-9])([0-9])-(19|20|21)"

raspisanie1={}
raspisanie2={}
raspisanie_by_day={}
subject_and_info=[]
info=[]
s=""
regf=r"(([А-Яа-яЁЁ]{3,20})(-?)([А-Яа-я]+)?( +)?([А-ЯЁ][., ] ?[А-Я][., ]?)?)?"

for i in range(1,num_cols):
    if re.search(reg,str(sheet.cell(row=2, column=i).value))!=None:
        group=re.search(reg,str(sheet.cell(row=2, column=i).value)).group()
        for i1 in range(4,76,2):
            if sheet.cell(row=i1, column=i).value!=None:
                s+=sheet.cell(row=i1, column=i).value
                if sheet.cell(row=i1, column=i+1).value != None:
                    s+=", " + str(sheet.cell(row=i1, column=i + 1).value)
                    if sheet.cell(row=i1, column=i + 2).value != None:
                        s += ", " + str(sheet.cell(row=i1, column=i + 2).value)
                        if sheet.cell(row=i1, column=i + 3).value != None:
                            s += ", " + str(sheet.cell(row=i1, column=i + 3).value)
                subject_and_info.append(s)
                s=""
            else:
                subject_and_info.append("--")

            if (i1-2)%12==0:
                raspisanie_by_day[str(sheet.cell(row=i1-10,
column=1).value)]=subject_and_info

                subject_and_info=[]

        raspisanie1[str(sheet.cell(row=2, column=i).value)]=raspisanie_by_day
        raspisanie_by_day={}


```



```

for i1 in range(5, 76, 2):
    if sheet.cell(row=i1, column=i).value != None:
        s += sheet.cell(row=i1, column=i).value
        if sheet.cell(row=i1, column=i + 1).value != None:
            s += ", " + str(sheet.cell(row=i1, column=i + 1).value)
            if sheet.cell(row=i1, column=i + 2).value != None:
                s += ", " + str(sheet.cell(row=i1, column=i + 2).value)
                if sheet.cell(row=i1, column=i + 3).value != None:
                    s += ", " + str(sheet.cell(row=i1, column=i + 3).value)
            subject_and_info.append(s)
        s = ""
    else:
        subject_and_info.append("--")
if (i1 - 3) % 12 == 0:
    raspisanie_by_day[str(sheet.cell(row=i1 - 11, column=1).value)] =
subject_and_info

    subject_and_info = []
    raspisanie2[str(sheet.cell(row=2, column=i).value)] = raspisanie_by_day
    raspisanie_by_day = {}

```

### Запись расписания преподавателей в словарь:

```

raspisanie_for_teacher1={}
raspisanie_for_teacher2={}

a=[]
for i in range(4,76,12):
    for i1 in range(6):
        a.append(" ")

def week():
    week_info = {}
    a = []
    for i in range(4, 76, 12):
        for i1 in range(6):
            a.append(" ")

            week_info[sheet.cell(row=i, column=1).value] = a
            a=[]
    return week_info

for i in range(1, num_cols):
    if sheet.cell(row=3, column=i).value != None and ("ФИО" in str(sheet.cell(row=3,
column=i).value)):
        for i1 in range(4, 76):

            tag = re.finditer(regf, str(sheet.cell(row=i1, column=i).value))
            for name in tag:

                if sheet.cell(row=i1, column=i) != None:

                    surname = name.group().replace(", ", ".")

```

```

        surname = surname.replace(".", ".")
        surname = surname.replace(' ', ' ')
        # print('surname', surname)
        if surname[len(surname) - 1] != '.' and
surname[len(surname) - 1] == '.':
            surname += '.'
            if surname[len(surname) - 1] == '-' or (len(surname) ==
2 and surname[len(surname) - 1] == ' '):
                continue
            raspisanie_for_teacher1.setdefault(surname, week())
            # print("Словарь до добавления", raspisanie_for_teacher1)
            raspisanie_for_teacher2.setdefault(surname, week())

        if i1 % 2 == 0:
            # print(i1, surname)
            # print(int(4 + 12 * int(int(i1 - 4) / int(12))),

```

Запись словарей с расписанием в json файл:

```

my_file=open("raspisanie1.json", "w")

my_file.write(json.dumps(raspisanie1))
my_file2=open("raspisanie2.json", "w")
my_file2.write(json.dumps(raspisanie2))
my_file=open("raspisanie_for_teacher1.json", "w")

my_file.write(json.dumps(raspisanie_for_teacher1))
my_file=open("raspisanie_for_teacher2.json", "w")

my_file.write(json.dumps(raspisanie_for_teacher2))

```

Получение погоды на сегодня:

```

respons=
requests.get("https://api.openweathermap.org/data/2.5/forecast?q=moscow&appid=93e8395
6c91b2e00ebbabe0672230693&units=metric")
    respons = respons.json()
    responses = requests.get(

"http://api.openweathermap.org/data/2.5/weather?q=moscow&appid=93e83956c91b2e00ebbabe
0672230693&units=metric")
        responses = responses.json()
        print(event.text.lower())
        vk.messages.send(

            user_id=event.user_id,
            random_id=get_random_id(),

            message="Ожидайте...")
        if event.text.lower()=="сейчас":

            upload=VkUpload(vk_session)
            attachemens=[]

image=requests.get("http://openweathermap.org/img/wn/{ }@2x.png".format(responses[ 'wea
ther' ][0][ 'icon' ]), stream=True)

```

```

print("http://openweathermap.org/img/wn/{0}@2x.png".format(responses['weather'][0]['icon']))
        photo=upload.photo_messages(photos=image.raw)[0]

attachemens.append("photo{0}_{1}".format(photo["owner_id"],photo['id']))
        derictionC = deriction_def(responses['wind']['deg'])
        print(derictionC)
        traslator = Translator(from_lang='en', to_lang="ru")
        print(attachemens)
        sspeed = sspeed_def(responses['wind']['speed'])

        ss = translate(responses['weather'][0]['description'])
        s = "{0} , температура: {1}-{2} °C\nОщущается: {8}\nДавление: {3} мм
пт. ст., влажность: {4}%\nВетер: {5}, {6} м/с, {7}". \
        format(ss,
                round(responses["main"]["temp_min"]),
round(responses["main"] \
["temp_max"]),
                round(responses['main']['pressure'] / 1.33),
responses['main'] \
                ['humidity'], sspeed, responses['wind']['speed'],
derictionC,
                responses['main']['feels_like'])

        vk.messages.send(

                user_id=event.user_id,
                random_id=get_random_id(),
                attachment=attachemens[0],
                message="Погода сейчас:\n"+s)

```

### Получение статистики коронавируса в определенном городе:

```

def corona_reg(region_row):
    link =
BeautifulSoup((requests.get('https://coronavirusstat.ru//')).text,
"html.parser").find_all("span", class_='small')
    for i in link:
        if region_row.lower() in i.text.lower():
            region_corr = i.text
            reg_link = i.find_parent().find_parent().find_parent()

            ill_block = reg_link.find('div', class_='p-1 col-7 row m-0')

            all = ill_block.find_all('div', class_='p-1 col-4 col-sm-2')
            good = ill_block.find_all('div', class_='p-1 col-4 col-sm-3')
            dead = ill_block.find_all('div', class_='p-1 col-3 col-sm-2 d-none d-sm-
block')
            abs = []
            for i in all:
                abs.append(i.text.replace('\n',' ').replace('\t',' ').split())
            for j in good:
                abs.append(j.text.replace('\n',' ').replace('\t',' ').split())
            for k in dead:
                abs.append(k.text.replace('\n',' ').replace('\t',' ').split())
            mes = f"Регион: {region_corr}\n"

```

```

for per in abs:
    s = f"{per[0]}: {per[1]} "
    if len(per) > 2:
        s+=f"Сегодня({per[2]})"
    mes+=s+'\n'
return mes

```

Получение статистики коронавируса в Москве и построение графика:

```

def corona_russia():
    link =
BeautifulSoup((requests.get('https://coronavirusstat.ru/country/russia/')).t
ext, "html.parser").find("tbody").find_all("tr")
    a = []
    x_list=[]
    cured = []
    active = []
    dead = []
    for i in link:
        a.append(i.text.split())
    #print(a[0])
    k = a[0]
    s = 'По состоянию на ' + k[0] + '\n'
    s += 'Случаев: ' + k[-3] + ' (' + k[-2] + ' за сегодня)\n'
    s += 'Активных: ' + k[1] + ' (' + k[2] + ' за сегодня)\n'
    s += 'Вылечено: ' + k[4] + ' (' + k[5] + ' за сегодня)\n'
    s += 'Умерло: ' + k[7] + ' (' + k[8] + ' за сегодня)'
    a = a[1:11]
    for i in a:
        active.append(int(i[1]))
        cured.append(int(i[4]))
        dead.append(int(i[7]))
        x_list.append(i[0][:5])
    barWidth = 1
    plt.title("Корона в России")
    plt.bar(x_list,cured, color='orange',
width=barWidth,label='Выздоровевшие')
    plt.bar(x_list,dead, color='purple', width=barWidth,label='Умершие')
    plt.bar(x_list,active, color='green', width=barWidth,label='Заболевшие')
    y_list = []
    for i in range(0,max(cured)+2500000,2500000):y_list.append(i)
    #print(y_list)
    plt.xlabel('', fontsize=15)
    plt.ylabel('Кол-во', fontweight='bold', fontsize=15)
    plt.yticks(y_list)
    plt.legend()
    plt.savefig("graph.png")
    return s

```

## Тестирование:

Получение статистики коронавируса в Москве:



**Берс** 22:55

корона

1 июня



**page** 22:55

По состоянию на 01.06.2022

Случаев: 18335514 (+4151 за сегодня)

Активных: 211330 (-1560 за сегодня)

Вылечено: 17744984 (+5628 за сегодня)

Умерло: 379200(+83 за сегодня)

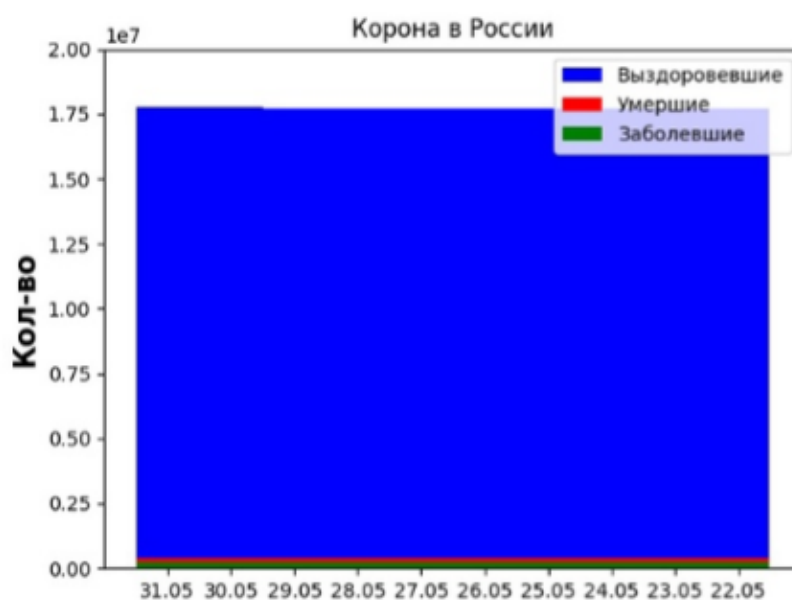


Рисунок 12 - статистика коронавируса в Москве

Получение статистики коронавируса в определенном регионе:



**Берс** 22:55  
корона ханты



**page** 22:55  
Регион: Ханты-Мансийский автономный округ - Югра  
Случаев: 49825 Сегодня(+39)  
Вылечено: 47360 Сегодня(+35)  
Активных: 1758 Сегодня(+3)  
Умерло: 707 Сегодня(+1)

Рисунок 13 - статистика коронавируса в ХМАО

Получение погоды в Москве на данный момент:



**Берс** 13:58  
погода



**page** 13:58  
Показать погоду в Москве



**Берс** 13:58  
Погода сейчас



**page** 13:58  
Ожидайте...

Погода сейчас:  
Облачно , температура: 19-24 °C  
Ощущается: 23.36  
Давление: 767 мм рт. ст., влажность: 68%  
Ветер: Умеренный, 5.43 м/с, юго-восточный



Рисунок 14 - погода в Москве

## Получение расписания преподавателя:



**Берс** 19:10

Найти габриелян



**page** 19:10

Показать расписание



**Берс** 19:10

На эту неделю



**page** 19:10

ПОНЕДЕЛЬНИК

1)

2)

3)

4) Информационные системы управления корпоративным контентом, пр, А-423, ИКБО-17-20

5)

6)

ВТОРНИК

1)

2)

3) Программирование на языке Питон, пр, И-202-6, ИНБО-03-20

4) Программирование на языке Питон, пр, И-202-6, ИНБО-03-20

Рисунок 15 - расписание преподавателя

## Получение расписания:



**Берс** 22:57  
икбо-03-20



**page** 22:57  
Отлично! Группа ИКБО-03-20 выбрана



**Берс** 22:57  
бот



**page** 22:57  
Показать расписание



**Берс** 22:57  
На сегодня



**page** 22:57  
Расписание на среда 01 июнь  
1 Анализ и концептуальное моделирование систем лк  
Геращенко Л.А. В-86\* К-5  
2 Дизайн мобильных приложений лк Болбаков Р.Г. В-86\* К-3  
3 Технология разработки программных приложений лк  
Петренко А.А. В-86\* К-3  
4 -  
5 -  
6 1 н Разработка мобильных приложений лк Чернов Е.А. Д

Рисунок 16 - Расписание



## 7) Заключение

В курсе ознакомительной практики на ЯП Python мы выполнили большое кол-во интересных и применимых на практике заданий. Начиная от начального этапа ООП с созданием библиотеки, до создания бота ВК и конвертера валют.

В течение курса мы коснулись таких библиотек как:

- Tkinter – кросс-платформенная графическая библиотека Python, позволяющая создавать как небольшие программки, так и крупные проекты.
- Matplotlib – библиотека, ориентированная на работу с данными. Пригодится для Data Science. Нам она помогла в создании большого кол-ва графиков.
- Re – очень практичная и легкая в понимании библиотека предназначенная для работы с регулярными выражениями.
- BeautifulSoup (bs4) – библиотека, позволившая нам работать с html страницами в удобном формате.
- VkApi – главная библиотека для работы с ВК.
- Request, openpyxl, pandas и т.д.