# 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 спсика ударных волн. Вычисление силы землетрясения для [[5,3,7], [3,3,1], [4,1,2]]-> ((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 введённого;

– 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 подхдящим 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 введённого \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:

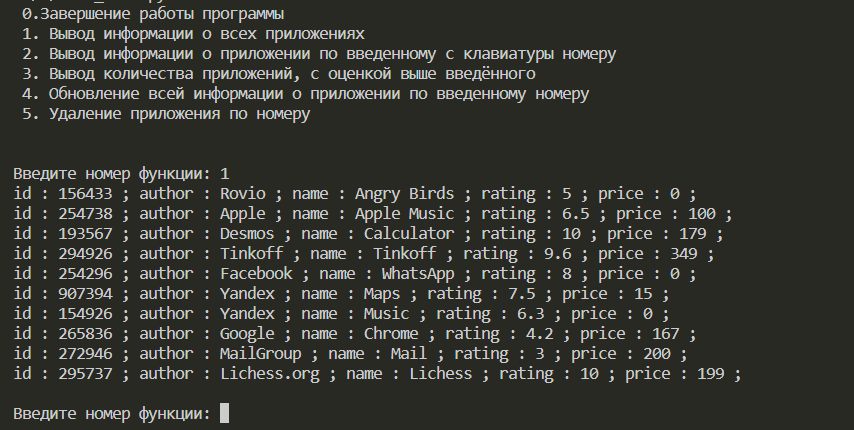


Рисунок 1 - Тестирование функции вывода информации о приложениях

Тестирование функции infoOutput\_byNum() - вывода информации о приложении по

номеру:

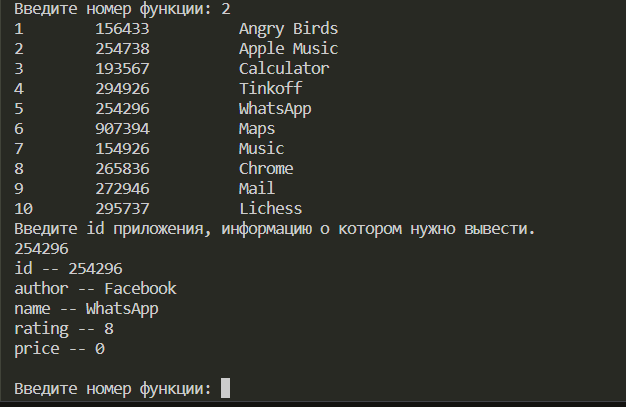


Рисунок 2 - Тестирование функции вывода информации о приложении по номеру

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

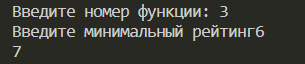


Рисунок 3 - Тестирование функции вывода количества приложений, с рейтингом выше заданного

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

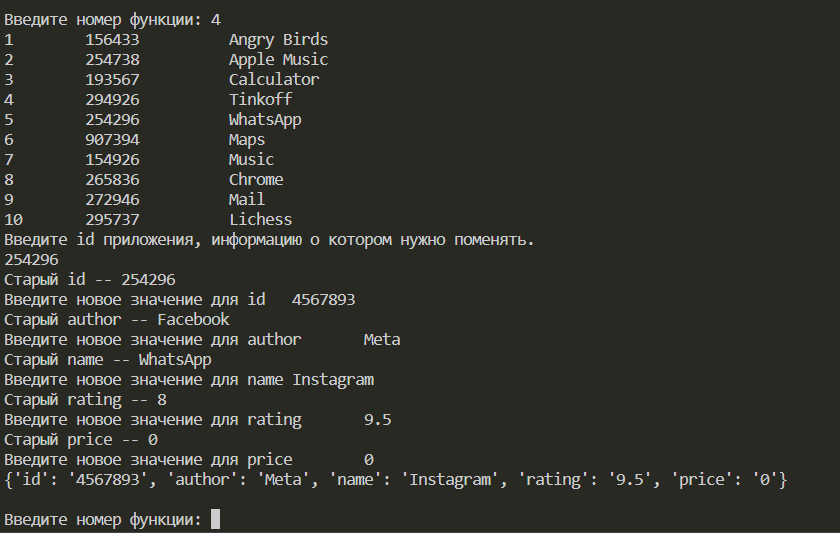


Рисунок 4 - Тестирование функции обновления всей информации о приложении по введенному ID

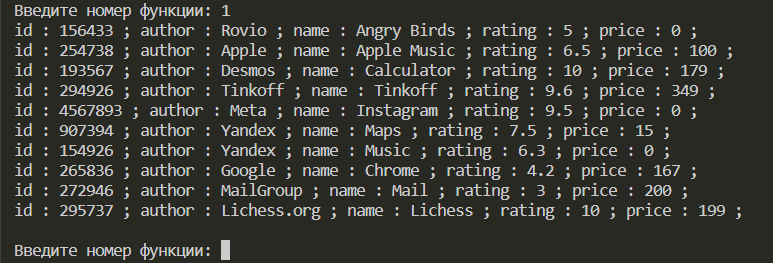


Рисунок 5 – Список приложений после изменения информации о приложении

Тестирование функции delByNum() - удаления приложения по ID:

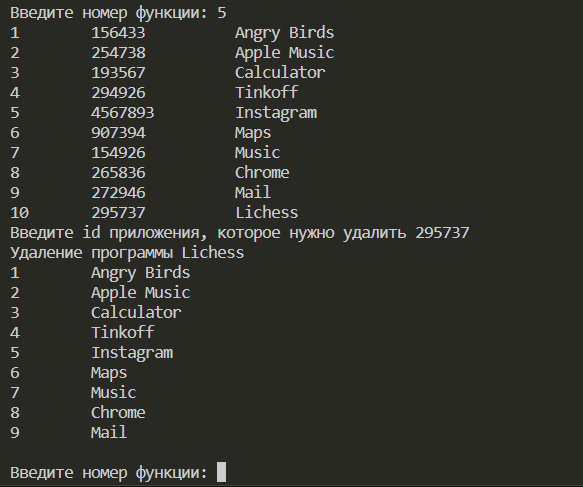


Рисунок 6 - Тестирование функции удаления приложения по ID

# 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:

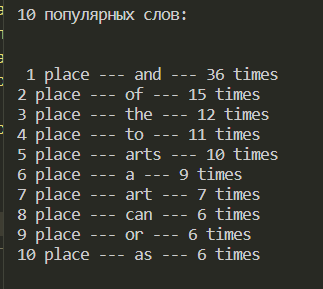


Рисунок 7 - Вывод программы

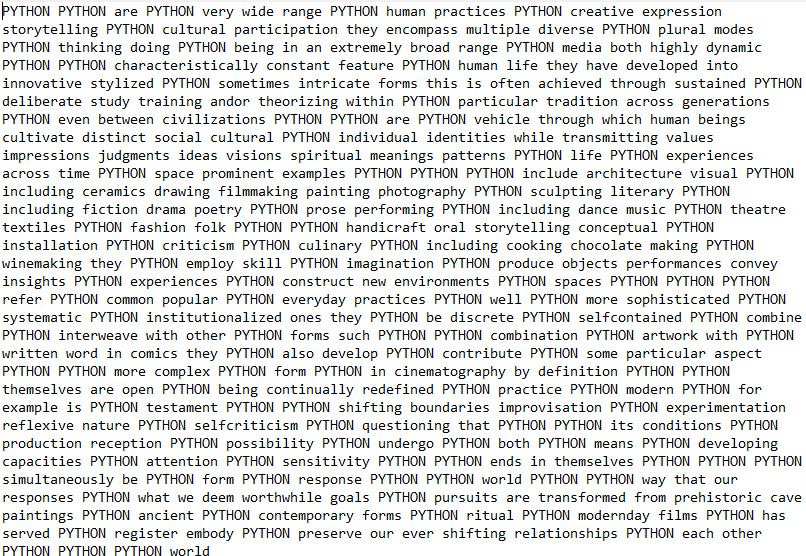


Рисунок 8 - Содержимое файла после работы программы

# 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\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)

return self

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\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')

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\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')

# 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...
  + math.e - e = 2,718281...
  + 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.

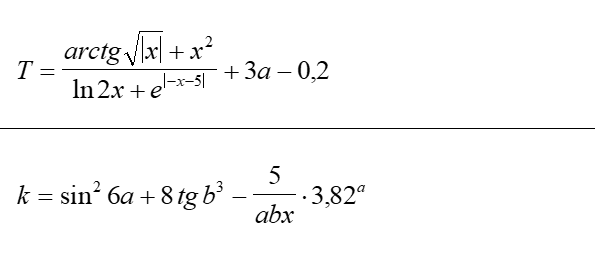


Рисунок 9 - Формулы

Решение задачи:

from math import \*

def fist\_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 A, B.
  + Написать функцию, которая создаст папку «Ознакомительная папка» с двумя подпапками («тема A», «тема B»), 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 A', True)

makedirs(path + '\\Ознакомительная практика\\Тема B', True)

for item in scandir(path + '\\allTasksD'):

if search(r'task\_([A-B]{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:

tasks = i[-1]

for task in tasks:

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}')

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}')

# 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])

#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 == 1:

quarter\_list.append("Весна "+(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])

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()

#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(',', '.')) / quant

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(',', '.')) / quant

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"))

plt.plot(x\_list,y\_list)

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 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:

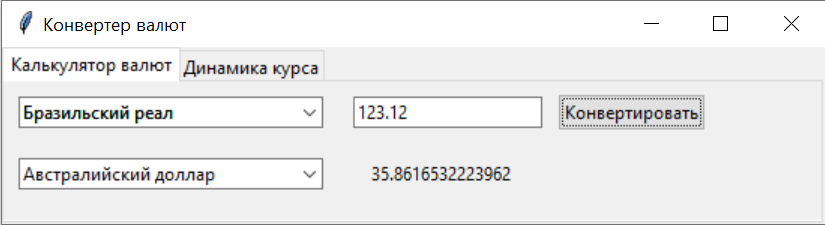


Рисунок 10 - Тестирование калькулятора валют

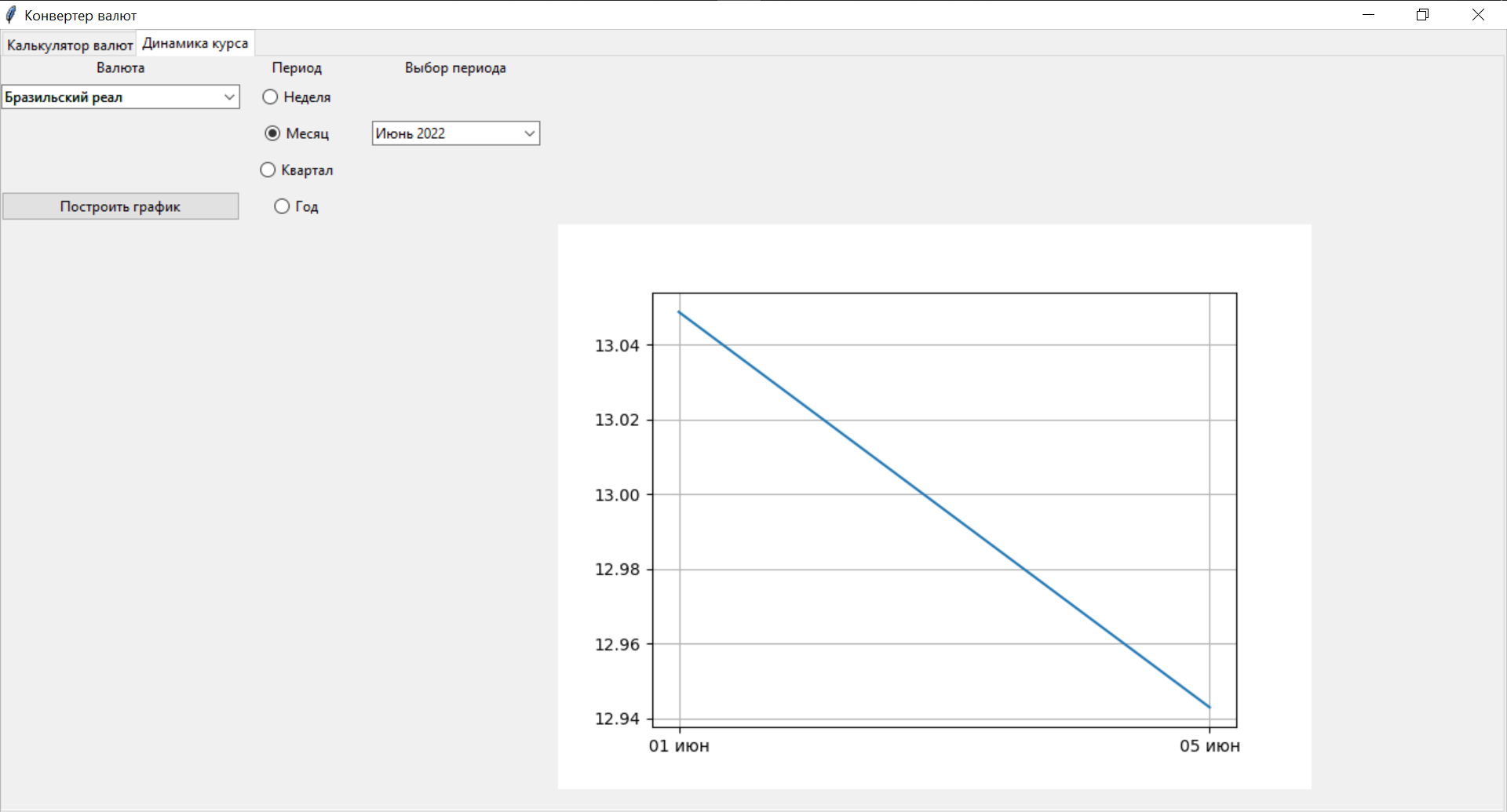


Рисунок 11 - Тестирование функции построения графиков

# 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)

.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])БО-([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))),

# sheet.cell(row=int(4 + 12 \* int(int(i1 - 4) / int(12))), column=1).value)

# print(((i1 - 4) % 12)/2)

# print("----------")

# if raspisanie\_for\_teacher1.get(surname)[

#

# sheet.cell(row=int(4 + 12 \* int(int(i1 - 4) / int(12))), column=1).value][

# int(((i1 - 4) % 12) / 2)] \

# == " " :

if str(sheet.cell(row=i1, column=i).value).find(surname[:3]) == 0:

# print("Обращение по ключу",raspisanie\_for\_teacher1.setdefault(surname))

raspisanie\_for\_teacher1.get(surname)[

sheet.cell(row=int(4 + 12 \* int(int(i1 - 4) / int(12))), column=1).value][

int(((i1 - 4) % 12) / 2)] \

+= str(sheet.cell(row=i1, column=i - 2).value)[

:str(sheet.cell(row=i1, column=i - 2).value).find('\n')] \

+ ", " + str(

sheet.cell(row=i1, column=i - 1).value) \

+ ", " + str(sheet.cell(row=i1, column=i + 1).value) + ", " + str(

sheet.cell(row=2, column=i - 2).value)+'\n'

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](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.](http://openweathermap.org/img/wn/)png".format(responses['wea ther'][0]['icon']), stream=True)

[print("http://openweathermap.org/img](http://openweathermap.org/img/wn/)/wn/{}@2x.png".format(responses['weather'][0]['ic on']))

photo=upload.photo\_messages(photos=image.raw)[0]

attachemens.append("photo{}\_{}".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} °С\nОщущается: {8}\nДавление: {3} мм рт. 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

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 посторенние графика:

def corona\_russia():

link = BeautifulSoup((requests.get('https://coronavirusstat.ru/country/russia/')).text, "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:

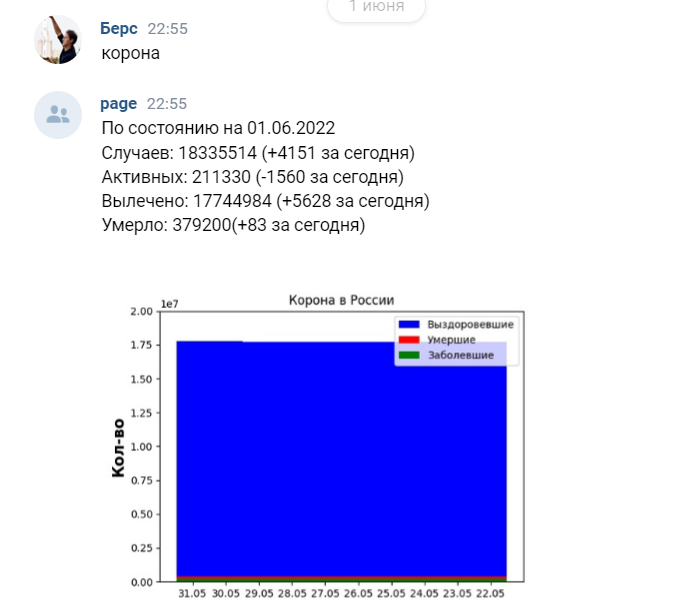


Рисунок 12 - статистика коронавируса в Москве

Получение статистики коронавируса в определенном регионе:

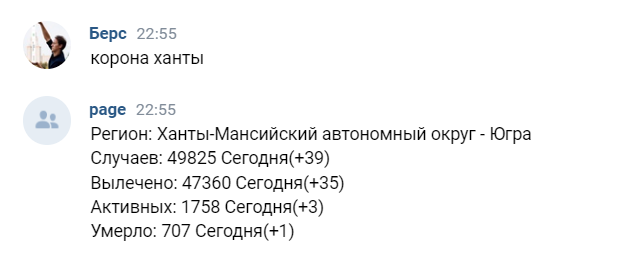


Рисунок 13 - статистика коронавируса в ХМАО

Получение погоды в Москве на данный момент:

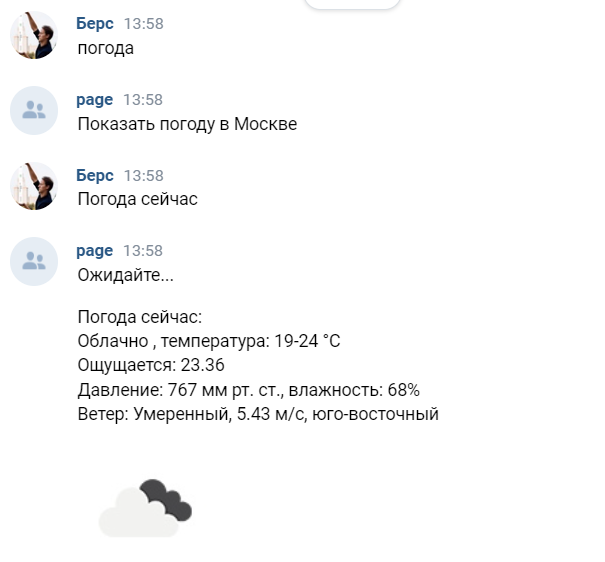


Рисунок 14 - погода в Москве

Получение расписания преподавателя:

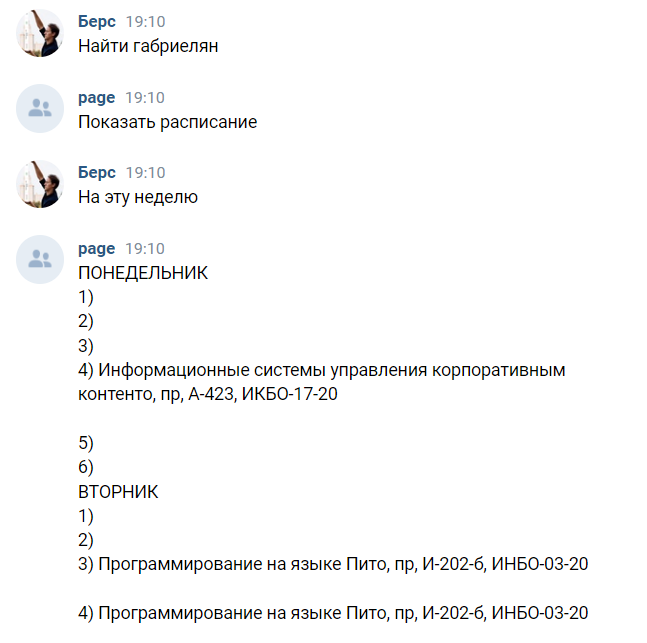


Рисунок 15 - расписание преподавателя

Получение расписания:

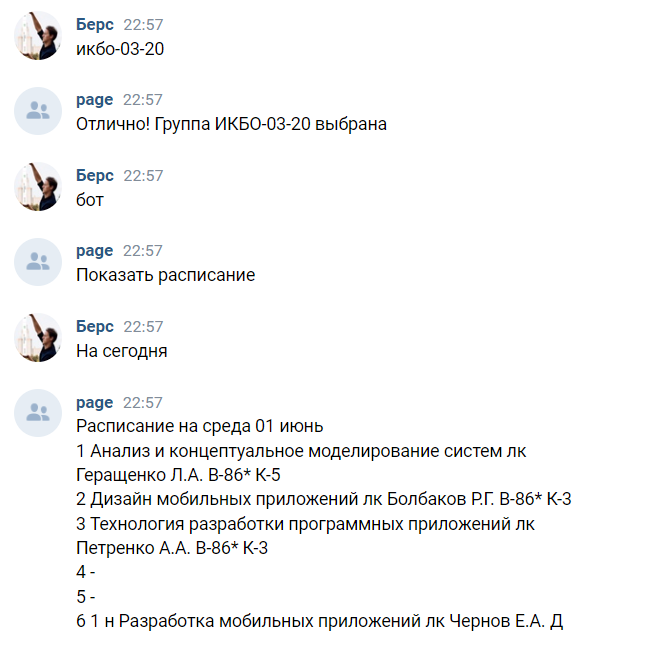


Рисунок 16 - Расписание

# 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, openyxls, pandas и т.д.