

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

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**

**Тема практики**: Улучшение навыков программирования посредством решения практических задач при изучение нового языка программирования»

приказ Университета о направлении на практику от «10» февраля 2020 г. № 742-С

|  |  |  |
| --- | --- | --- |
| Отчет представлен к  рассмотрению:  Студент группы ИКБО-15-19 | «23» сентября 2021 г. | Бородин К.В.    \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  *(подпись)* |
| Отчет утвержден.  Допущен к защите: |  |  |
| Руководитель практики от кафедры | «23» сентября 2021 г. | Габриелян Г. А.  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  *(подпись)* |

Москва 2021 г.

|  |
| --- |
| МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ  Федеральное государственное  бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** |

Институт информационных технологий

Кафедра корпоративных информационных систем

**ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ НА УЧЕБНУЮ ПРАКТИКУ**

**Ознакомительная практика**

**студенту 1** **курса учебной группы** ИКБО-15-19 **института** ИТ

Бородина Кирилла Владимировича

**Место и время проведения:** РТУ МИРЭА, кафедра КИС, II семестр (10.02.20 – 30.05.20).

**Должность на практике:** студент

**1.ЦЕЛЕВАЯ УСТАНОВКА:** улучшение навыков программирования посредством решения практических задач при изучение нового языка программирования.

**2.СОДЕРЖАНИЕ ПРАКТИКИ:**

**2.1.** Изучить: программную документацию языка программирования и стандартных библиотек, API применяемых сервисов.

**2.2.** Практически выполнить: задания по темам: основные конструкции языка, работа с коллекциями, работа со строками, работа с файлами; основы объектно-ориентированного программирования, стандартные библиотеки, графический интерфейс; использование API сторонних сервисов.

**2.3.** Ознакомиться: с Положением о практике обучающихся, осваивающих основные профессиональные образовательные программы бакалавриата; с учебно-методическим пособием по ознакомительной практике.

**3.ДОПОЛНИТЕЛЬНОЕ ЗАДАНИЕ:** отсутствует.

**4.ОРГАНИЗАЦИОННО-МЕТОДИЧЕСКИЕ УКАЗАНИЯ:** Положение о практике обучающихся, осваивающих основные профессиональные образовательные программы бакалавриата; учебно-методическое пособие по ознакомительной практике.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Заведующий кафедрой | КИС  *(наименование)* | \_\_\_\_\_\_\_\_  *(подпись)* | (Андрианова Е. Г.)  *(фамилия и инициалы)* | «23» сентября 2021 г. | |
| СОГЛАСОВАНО | |  |  |  | |
| Руководитель практики | ассистент | \_\_\_\_\_\_\_\_  *(подпись)* | (Габриелян Г. А.)  *(фамилия и инициалы)* | «23» сентября 2021 г. | |
| Задание получил | студент | \_\_\_\_\_\_\_\_  *(подпись)* | (Бородин К.В.)  *(фамилия и инициалы)* | «3» сентября 2021 г. | |
| **Изображение выглядит как текст, коллекция картинок  Автоматически созданное описание**  МИНОБРНАУКИ РОССИИ  Федеральное государственное  бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** | | | | |

Институт информационных технологий

Кафедра корпоративных информационных систем

**РАБОЧИЙ ГРАФИК ПРОВЕДЕНИЯ**

**УЧЕБНОЙ ПРАКТИКИ**

студента Бородина К. В. 1 курса группы ИКБО-15-19 очной формы обучения, обучающегося по направлению подготовки «Программная инженерия», профиль «Информационные системы управления ресурсами предприятия».

|  |  |  |  |
| --- | --- | --- | --- |
| **Неделя** | **Сроки выполнения** | **Этап** | **Отметка о выполнении** |
| 1 | 10.02 – 15.02 | Вводная лекция, проведение инструктажа |  |
| 2-3 | 17.02 – 29.02 | Изучение и решение задач по теме «Основные конструкции языка, коллекции» |  |
| 4-5 | 02.03 – 14.03 | Изучение и решение задач по теме «Строки, работа с файлами» |  |
| 6-7 | 16.03 – 28.03 | Изучение и решение задач по теме «Основы объектно-ориентированного программирования» |  |
| 8-9 | 30.03 – 11.04 | Изучение и решение задач по теме «Стандартные библиотеки языка программирования» |  |
| 10-11 | 13.04 – 25.04 | Изучение и решение задач по теме «Графический интерфейс и внешние библиотеки» |  |
| 12-14 | 27.04 – 16.05 | Использование сторонних API для создания приложений, подготовка отчетных материалов |  |
| 15-16 | 18.05 – 30.05 | Подготовка отчетной документации |  |

**СОГЛАСОВАНО:**

Заведующий кафедрой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Андрианова Е. Г.

*(подпись)*

Руководитель практики от кафедры \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Габриелян Г. А.

*(подпись)*

Обучающийся \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Бородин К.В.

*(подпись)*

**Проведенные инструктажи:**

|  |  |  |
| --- | --- | --- |
| Охрана труда: |  | «10» февраля 2020 г. |
| Инструктирующий | *(подпись)* | Трохаченкова Н.Н., старший преподаватель кафедры КИС |
| Инструктируемый | *(подпись)* | Мамыров С.У |
|  |  |  |
| Техника безопасности: |  | «10» февраля 2020 г. |
| Инструктирующий | *(подпись)* | Трохаченкова Н.Н., старший преподаватель кафедры КИС |
| Инструктируемый | *(подпись)* | Бородин К.В. |
|  |  |  |
| Пожарная безопасность: |  | «10» февраля 2020 г. |
| Инструктирующий | *(подпись)* | Трохаченкова Н.Н., старший преподаватель кафедры КИС |
| Инструктируемый | *(подпись)* | Бородин К.В. |
|  | |  |
| С правилами внутреннего распорядка ознакомлен: | | «10» февраля 2020 г. |
|  | *(подпись)* | Бородин К.В. |

**ОТЧЁТ**

**по УЧЕБНОЙ практике**

**студента 1 курса учебной группы ИКБО-15-19 института ИТ Университета**

Бородина Кирилла Владимировича

1. Практику проходил с 10.02.2020 г. по 30.05.2020 г. в ФГБОУ ВО «МИРЭА – Российский технологический университет», на кафедре корпоративных информационных систем, студент

*(место прохождения практики и должность)*

2. Задание на практику выполнил

частично

*(указать: в полном объеме или частично)*

Не выполнены следующие задания:

«Использование сторонних API для создания приложений».

*(указать также причины невыполнения)*

Подробное содержание выполненной на практике работы и достигнутые результаты: успешно изучены и решены задачи по темам: «Основные конструкции языка, коллекции», «Строки, работа с файлами», «Основы объектно-ориентированного программирования», «Стандартные библиотеки языка программирования», «Графический интерфейс и внешние библиотеки», «Использование сторонних API для создания приложений».

Предложения по совершенствованию организации и прохождения практики:

предложений нет

Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (Бородин К.В.)

*(подпись)*

«23» сентября 2021 г.

Заключение руководителя практики

|  |
| --- |
| Приобрел следующие профессиональные навыки: |
|  |
|  |
|  |
|  |
| Проявил себя как: |
|  |
|  |
|  |
|  |

«23» сентября 2021 г.

**Отчет проверил:**

**Руководитель практики от Университета**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (Габриелян Г. А.)

*(подпись)*

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

**Задача 1 (task139)**

Условие задачи: написать функцию special\_number(number), которая определяет является ли число особенным. Назовем число особенным, если сумма цифр числа, возведенных в степень, равную позиции цифры, равна самому числу.

Решение задачи:

def special\_number(number):

num\_string = str(number)

z = 1

sum = 0

for i in num\_string:

sum = sum + pow(int(i), z)

z = z + 1

if sum == number:

return True

else:

return False

Тестирование:

|  |  |  |  |
| --- | --- | --- | --- |
| № | Тест | Ожидаемое значение | Полученное значение |
| 1 | 1 | True | True |
| 2 | 2 | True | True |
| 3 | 89 | True | True |
| 4 | 77 | False | False |
| 5 | 518 | True | True |

**Задача 2 (task209)**

Условие задачи: задан список из четного количества целых положительных чисел. Написать функцию min\_sum, которая возвращает минимальную сумму произведений по два числа.

Решение задачи:

def min\_sum(arr):

arr.sort()

summa = 0

##print (arr)

while len(arr) != 0:

summa += arr.pop(0) \* arr.pop()

##print (summa)

return summa

Тестирование:

|  |  |  |  |
| --- | --- | --- | --- |
| № | Тест | Ожидаемое значение | Полученное значение |
| 1 | [5, 4, 2, 3] | 22 | 22 |
| 2 | [12, 6, 10, 26, 3, 24] | 342 | 342 |
| 3 | [9, 2, 8, 7, 5, 4, 0, 6] | 74 | 74 |

**Задача 3 (task355)**

Условие задачи: в старину почту доставляли гонцы на лошадях. Между городами находились почтовые станции, где можно было поменять лошадь или передать пакет следующему гонцу Задан список расстояний от одной станции к другой. Написать функцию riders, которая определяет сколько гонцов будут участвовать в доставке почты, если каждый путешествует так далеко, как только может, но не более 100 км.

Решение задачи:

def riders(stations):

Horse\_stamina = 100

k = 1

while len(stations)!=0:

if (Horse\_stamina - stations[0]) > 0:

Horse\_stamina -= stations.pop(0)

else:

k=k+1

Horse\_stamina = 100

return k

Тестирование:

|  |  |  |  |
| --- | --- | --- | --- |
| № | Тест | Ожидаемое значение | Полученное значение |
| 1 | [18, 15] | 1 | 1 |
| 2 | [43, 23, 40, 13] | 2 | 2 |
| 3 | [33, 8, 16, 47,30, 30, 46] | 3 | 3 |
| 4 | [6, 24, 6, 8, 28, 8, 23, 47, 17, 29, 37, 18, 40, 49] | 4 | 4 |

**Задача 4 (task474)**

Условие задачи: написать функцию order\_weight, которая сортирует список положительных чисел. Критерий сортировки – возрастание веса числа (сумма всех цифр числа). Если два числа имеют одинаковый вес, сортировать их так, словно они строки.

Решение задачи:

def order\_weight(integers):

#массив для веса чисел

weight\_integers = []

#вспомогательная переме

sum = 0

#считаем сумму числа и добавляем в массив

for i in integers:

while i > 0:

sum += i % 10

i = int(i/10)

weight\_integers.append(sum)

sum = 0

# определяем длину массива

Len = len(weight\_integers)

#Внешний цикл, количество проходов N-1

for i in range(Len):

# Внутренний цикл, N-i-1 проходов

for j in range(0, Len-i-1):

#если элементы меньше то меняем их местами

if weight\_integers[j] > weight\_integers[j+1]:

weight\_integers[j], weight\_integers[j+1] = weight\_integers[j+1], weight\_integers[j]

integers[j], integers[j+1] = integers[j+1], integers[j]

#если элементы равны, сравниваем числа как строки

elif weight\_integers[j] == weight\_integers[j+1]:

if str(integers[j]) > str(integers[j+1]):

weight\_integers[j], weight\_integers[j+1] = weight\_integers[j+1], weight\_integers[j]

integers[j], integers[j+1] = integers[j+1], integers[j]

return integers

Тестирование:

|  |  |  |  |
| --- | --- | --- | --- |
| № | Тест | Ожидаемое значение | Полученное значение |
| 1 | [103, 123, 4444, 99, 2000] | [2000, 103, 123, 4444, 99] | [2000, 103, 123, 4444, 99] |
| 2 | [2000, 10003, 1234000, 44444444, 9999, 11, 11, 22, 123] | [11, 11, 2000, 10003, 22, 123, 1234000, 44444444, 9999] | [11, 11, 2000, 10003, 22, 123, 1234000, 44444444, 9999] |

**Задача 5 (task560)**

Условие задачи: создать список (супермаркет), состоящий из словарей (товары). Словари должны содержать как минимум 5 полей (например, номер, наименование, отдел продажи, …). В список добавить хотя бы 10 словарей.

Решение задачи:

market = [

{'id':1, 'product':'Coca-Cola', 'department':'Drinks', 'price':100, 'weight':0.5},

{'id':2, 'product':'Still\_Water', 'department':'Drinks', 'price':30, 'weight':0.5},

{'id':3, 'product':'Bread', 'department':'Bakery', 'price':25, 'weight':0.3},

{'id':4, 'product':'Meat', 'department':'Fresh\_meat', 'price':250, 'weight':1},

{'id':5, 'product':'Tomatoes', 'department':'Grossery', 'price':70, 'weight':1},

{'id':6, 'product':'Cucumbers', 'department':'Grossery', 'price':60, 'weight':1},

{'id':7, 'product':'Potatoes', 'department':'Grossery', 'price':90, 'weight':1},

{'id':8, 'product':'Potato\_Chips', 'department':'Snacks', 'price':100, 'weight':0.2},

{'id':9, 'product':'Crackers', 'department':'Snacks', 'price':140, 'weight':0.5},

{'id':10, 'product':'Milk', 'department':'Grossery', 'price':79, 'weight':0.5}

]

def printInfoAllProduct(market):

for i in market[0]:

print(f"{i:12} |", end=" ")

print()

for product in market:

for j in product:

print(f"{product[j]:12} |", end=" ")

print()

def printProductWithId(market):

id = int(input("Введите id товара - "))

idFind = False

for product in market:

if id == product["id"]:

idFind = True

for j in product:

print(f"{j} : {product[j]} |", end=" ")

print()

if not idFind:

print("товара с таким id не существует, пёс")

def printDepartmentCount(market):

department = input("Введите название отдела - ")

count = 0

for product in market:

if department == product["department"]:

count += 1

if count == 0:

print("Такого отдела не существет")

else:

print(f"Количество товаров в отделе {department} - {count}")

def changeProductWithId(market):

id = int(input("Введите ид товара, которого хотите поменять - "))

for product in market:

if id == product["id"]:

for j in product:

product[j] = input(f"Введите значение для {j} - ")

def delProductWithId(market):

id = int(input("Введите ид товара, которогы хотите удалить - "))

for product in market:

if id == product["id"]:

market.remove(product)

print('''

1 – вывод информации о всех товарах; \n

2 – вывод информации о товаре по введенному с клавиатуры номеру; \n

3 – вывод количества товаров, продающихся в определнном отделе; \n

4 – обновлении всей информации о товаре по введенному номеру; \n

5 – удалении товара по номеру; \n

0 - выход из программы.

''')

while True:

whatMeDo = int(input("Выберите функцию - "))

if whatMeDo == 1:

printInfoAllProduct(market)

elif whatMeDo == 2:

printProductWithId(market)

elif whatMeDo == 3:

printDepartmentCount(market)

elif whatMeDo == 4:

changeProductWithId(market)

elif whatMeDo == 5:

delProductWithId(market)

elif whatMeDo == 0:

exit()

Тестирование:

Выберите функцию - 1

id | product | department | price | weight |

1 | Coca-Cola | Drinks | 100 | 0.5 |

2 | Still\_Water | Drinks | 30 | 0.5 |

3 | Bread | Bakery | 25 | 0.3 |

4 | Meat | Fresh\_meat | 250 | 1 |

5 | Tomatoes | Grossery | 70 | 1 |

6 | Cucumbers | Grossery | 60 | 1 |

7 | Potatoes | Grossery | 90 | 1 |

8 | Potato\_Chips | Snacks | 100 | 0.2 |

9 | Crackers | Snacks | 140 | 0.5 |

10 | Milk | Grossery | 79 | 0.5 |

Выберите функцию - 2

Введите id товара - 7

id : 7 | product : Potatoes | department : Grossery | price : 90 | weight : 1 |

Выберите функцию - 3

Введите название отдела - Snacks

Количество товаров в отделе Snacks – 2

**2) Тема «Строки, работа с файлами»**

**Задача 1 (task2\_118)**

Условие задачи: написать функцию case\_sensitive, которая определяет все ли символы в заданной строке являются прописными и возвращает список из значения истинности и списка заглавных букв

Решение задачи:

def case\_sensitive(s):

upperCase = []

for i in s:

if i.isupper():

upperCase.append(i)

return [ not len(upperCase) > 0, upperCase]

Тестирование:

|  |  |  |  |
| --- | --- | --- | --- |
| № | Тест | Ожидаемое значение | Полученное значение |
| 1 | ‘asd’ | True | True |
| 2 | ‘callS’ | False, [S] | False, [S] |
| 3 | ‘z’ | True | True |
| 4 | ‘ ‘ | True | True |
| 5 | ‘aaTRggjS’ | False, [T], [R], [S] | False, [T], [R], [S] |

**Задача 2 (task2\_218)**

Условие задачи: написать функцию blocks, которая получает строку, состоящую из букв и цифр и возвращает строку в виде блоков, разделенных символом дефис. Элементы блока должны быть отсортированы по принципу, указанному ниже, и каждый блок не может содержать несколько экземпляров одного и того же символа.

Порядок блоков:

Строчные буквы (a – z) в алфавитном порядке

Заглавные буквы (A – Z) в алфавитном порядке

Цифры (0 – 9) в порядке возрастания

Решение задачи:

def blocks(s):

#чанки для сортировки

chunks = ['']

#готовая строка

sortedStrings = ''

addSymbol = True

#разбиваем строку на чанки из маленьких букв, больших букв, цифр

addSymbol = True

for index, elem in enumerate(s):

addSymbol = True

#мелкие буквы

if not elem.isupper() and not elem.isdigit():

for indexChunk, elemChunk in enumerate(chunks):

if not elem in elemChunk and not elemChunk.isupper() and not elemChunk.isdigit():

chunks[indexChunk] += elem

addSymbol = False

break

if addSymbol:

chunks.append(elem)

#большие буквы

if elem.isupper() and not elem.isdigit():

for indexChunk, elemChunk in enumerate(chunks):

if not elem in elemChunk and elemChunk.isupper():

chunks[indexChunk] += elem

addSymbol = False

break

if addSymbol:

chunks.append(elem)

#цифры

if elem.isdigit():

for indexChunk, elemChunk in enumerate(chunks):

if not elem in elemChunk and elemChunk.isdigit():

chunks[indexChunk] += elem

addSymbol = False

break

if addSymbol:

chunks.append(elem)

#сортируем чанки

for i in range(0, len(chunks)):

chunks[i] = "".join(sorted(chunks[i]))

#объединяем чанки в строку

index = 0

while len(chunks) != 0:

for i in range(0,len(chunks)):

if not chunks[i].isupper() and not chunks[i].isdigit():

sortedStrings += chunks[i]

chunks.pop(i)

break

for i in range(0,len(chunks)):

if chunks[i].isupper() and not chunks[i].isdigit():

sortedStrings += chunks[i]

chunks.pop(i)

break

for i in range(0,len(chunks)):

if chunks[i].isdigit():

sortedStrings += chunks[i]

chunks.pop(i)

break

if len(chunks) != 0:

sortedStrings += "-"

return sortedStrings

Тестирование:

|  |  |  |  |
| --- | --- | --- | --- |
| № | Тест | Ожидаемое значение | Полученное значение |
| 1 | "21AxBz" | "xzAB12" | "xzAB12" |
| 2 | "abacad" | "abcd-a-a" | "abcd-a-a" |
| 3 | "heyitssampletestkk" | "aehiklmpsty-ekst-est" | "aehiklmpsty-ekst-est" |

**Задача 3 (task2\_342)**

Условие задачи: написать функцию next\_version, которая будет принимать строку (текущая версия программного обеспечения) и будет возвращать строку, содержащую следующий номер версии.  
Правила: все числа, кроме первого, должны быть меньше 10. Если после увеличения они становятся равными 10 - установите их в 0 и последовательно увеличите следующий номер

Решение задачи:

def next\_version(s):

#разбиваем строку на список

versionList = (s.split('.'))

#переводим список строк в список чисел

versionList = list(map(int, versionList))

#переворачиваем список для прохода с конца

versionList = versionList[::-1]

#проходимся по всему списку

for index, number in enumerate(versionList):

#если число меньше девяти и это не первый(последний) элемент списка - прибавить к нему один и выйти из цикла

if number < 9 and index != (len(versionList) - 1):

versionList[index] += 1

break

#если первый(последний) элемент списка - прибавить к нему один

elif index == (len(versionList) - 1):

versionList[index] += 1

#если элемент равен девяти - обнулить его

else:

versionList[index] = 0

#перводим список чисел в список строк

versionList = list(map(str, versionList))

#переворачиваем для равномерного прогрева шавухи

versionList = versionList[::-1]

#объединяем идигриеты в строку

s = '.'.join(versionList)

return s

Тестирование:

|  |  |  |  |
| --- | --- | --- | --- |
| № | Тест | Ожидаемое значение | Полученное значение |
| 1 | "1.2.3" | "1.2.4" | "1.2.4" |
| 2 | "0.9.9" | "1.0.0" | "1.0.0" |
| 3 | "1" | "2" | "2" |
| 4 | "1.2.3.4.5.6.7.8" | "1.2.3.4.5.6.7.9" | "1.2.3.4.5.6.7.9" |
| 5 | "9.9" | "10.0" | "10.0" |

**Задача 4 (task2\_408)**

* 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 метод format  
  (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  
  и не делить слова.

Решение задачи:

import codecs

import re

def wiki\_function():

wikiText = codecs.open('wiki\_2.4.txt', 'r', 'utf\_8\_sig')

wikiList = []

#непустые строки добавить в список

for line in wikiText:

if line and line != '\r\n':

wikiList.append(line)

#закрываем файл для не надобности

wikiText.close()

#удалить из каждой строки все цифры, знаки препинания, скобки, кавычки и т.д. (остаются латинские буквы и пробелы)

for i in range(0, len(wikiList)):

#могучие РЕГУЛЯРКИ)))))))))

wikiList[i] = re.sub(r'[^a-zA-Z\s]+', '', wikiList[i])

wikiList[i] = (re.sub(r'[\r\n]+', '', wikiList[i])).strip()

wikiList[i] = re.sub(r' +', ' ', wikiList[i])

#объединить все строки из списка в одну, используя метод join и пробел, как разделитель;

wikiString = ' '.join(wikiList)

#создать словарь вида {“слово”: количество, “слово”: количество, … } для подсчета количества разных слов,

#где ключом будет уникальное слово, а значением - количество;

wikiDict = {wikiString.split(' ')[0].lower(): 1}

wikiWordsList = wikiString.split(' ')

for word in wikiWordsList:

addWord = True

for key in wikiDict.copy():

if word.lower() == key:

wikiDict[key] += 1

addWord = False

break

if addWord:

wikiDict[word.lower()] = 1

#вывести в порядке убывания 10 наиболее популярных слов, используя метод format

#(вывод примерно следующего вида: “ 1 place --- sun --- 15 times \n....”);

wikiListReplace = wikiString.split(' ')

place = 1

for key, item in sorted(wikiDict.items(),key=lambda item: item[1], reverse=True)[:10]:

print("{0} place --- {1} --- {2} times \n".format(place, key, item))\

#заменить все эти слова в строке на слово “PYTHON”;

for index, elem in enumerate(wikiListReplace):

if elem == key:

wikiListReplace[index] = "PYTHON"

place += 1

wikiString = ' '.join(wikiListReplace)

#записать строку в файл, разбивая на строки, при этом на каждой строке записывать не более 100 символов

#и не делить слова.

readyTxt = codecs.open('readyTxt\_2.4.txt', 'w', 'utf\_8\_sig')

step = 0

stepEnd = 1

print(wikiString)

while True:

stepEnd += wikiString[step: step + 100].rfind(' ')

if stepEnd == step:

break

readyTxt.write(wikiString[step: stepEnd] + "\n")

print(step, stepEnd, wikiString[step: stepEnd])

step = stepEnd

readyTxt.close()

return 1

# Вызов функции

wiki\_function()

Тестирование:

См.Приложения 1 и 2

**7) Заключение**

В результате прохождения ознакомительной практики освоил:

-основные конструкции языка, коллекции; строки;

-работа с файлами; основы объектно-ориентированного программирования;

-стандартные библиотеки языка программирования;

-графический интерфейс и внешние библиотеки.

Научился работать на языке программирования Python. Выполнял:

- основные конструкции языка, коллекции;

-строки, работа с файлами;

- основы объектно-ориентированного программирования;

Выучил различные библиотеки Python, научился подключать их. Ознакомился:

- основные конструкции языка, коллекции;

-строки, работа с файлами;

- основы объектно-ориентированного программирования;

- стандартные библиотеки языка программирования.