**Московский государственный технический**

**университет им. Н.Э. Баумана.**

Факультет «Информатика и управление»

Кафедра «Системы обработки информации и управления»

Курс «Базовые компоненты интернет-технологий»

Отчет по лабораторной работе №3-4

## «Основные конструкции языка Python»

|  |  |  |
| --- | --- | --- |
| Выполнил: |  | Проверил: |
| студент группы ИУ5-35Б |  |
| Дзитиев Вадим |  |
| Подпись и дата: |  | Подпись и дата: |

Москва, 2022г.

Задание:

1) Необходимо реализовать генератор field. Генератор field последовательно выдает значения ключей словаря

2) Необходимо реализовать генератор gen\_random(количество, минимум, мак-симум), который последовательно выдает заданное количество случайных чисел в заданном диапазоне от минимума до максимума, включая границы диапазона

3) Необходимо реализовать итератор Unique(данные), который принимает на вход массив или генератор и итерируется по элементам, пропуская дубли-каты

4) Дан массив 1, содержащий положительные и отрицательные числа. Необхо-димо одной строкой кода вывести на экран массив 2, которые содержит зна-чения массива 1, отсортированные по модулю в порядке убывания. Сорти-ровку необходимо осуществлять с помощью функции sorted

5) Необходимо реализовать декоратор print\_result, который выводит на экран результат выполнения функции

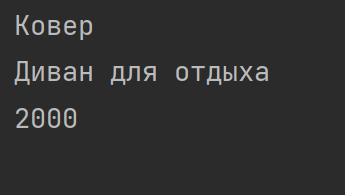
6) Необходимо написать контекстные менеджеры cm\_timer\_1 и cm\_timer\_2, ко-торые считают время работы блока кода и выводят его на экран

7) В предыдущих задачах были написаны все требуемые инструменты для ра-боты с данными. Применить их на реальном примере

Задание 1:

def field(items, \*args):  
 assert len(args) > 0  
 for a in range(len(args)):  
 for i in range(len(items)):  
 try: print(items[i][args[a]])  
 except: print()  
  
  
  
goods = [  
 {'title': 'Ковер', 'price': 2000, 'color': 'green'},  
 {'title': 'Диван для отдыха', 'color': 'black'}  
]  
  
  
field(goods, "title", "price")

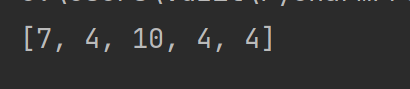
Результат:



Задание 2:

import random  
  
def gen\_random(num\_count, begin, end):  
 a=[]  
 for i in range(num\_count): a.append(random.randint(begin,end))  
 print(a)  
  
  
  
gen\_random(5,1,10)

Результат:



Задание 3:

class Unique(object):  
 def \_\_init\_\_(self, items, \*\*kwargs):  
 self.items = items  
 for k, v in kwargs.items():  
 setattr(self, k, v)  
 if not hasattr(self, "ignore\_case"):  
 self.ignore\_case=False  
  
 def \_\_next\_\_(self):  
 if self.ignore\_case:  
 li=[]  
 for x in self.items: li.append(x.lower())  
 else:  
 li = self.items  
  
 self.items = []  
 for i in li:  
 if i not in self.items:  
 self.items.append(i)  
 return self.items  
  
 def \_\_iter\_\_(self):  
 return self



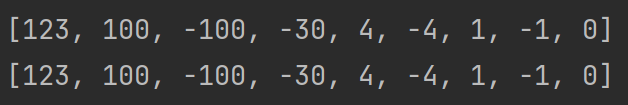
Результат:

****

Задание 4:

data = [4, -30, 100, -100, 123, 1, 0, -1, -4]  
  
if \_\_name\_\_ == '\_\_main\_\_':  
  
 result = sorted(data,key=abs ,reverse=True)  
 print(result)  
  
 result\_with\_lambda = sorted(data,key=lambda x: abs(x),reverse=True)  
 print(result\_with\_lambda)

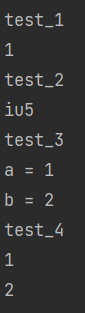
Результат:



Задание 5:

def print\_result(func):  
 def function():  
 print(func.\_\_name\_\_)  
 func\_result = func()  
  
 if isinstance(func\_result, int) or isinstance(func\_result, str):  
 print(func\_result)  
 elif isinstance(func\_result, dict):  
 all\_keys = list(func\_result.keys())  
 all\_values = list(func\_result.values())  
 for elem in range(len(all\_keys)):  
 print(all\_keys[elem], '=', all\_values[elem])  
 elif isinstance(func\_result, list):  
 for elem in func\_result:  
 print(elem)  
  
 return function  
  
  
  
@print\_result  
def test\_1():  
 return 1  
  
  
@print\_result  
def test\_2():  
 return 'iu5'  
  
  
@print\_result  
def test\_3():  
 return {'a': 1, 'b': 2}  
  
  
@print\_result  
def test\_4():  
 return [1, 2]  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 test\_1()  
 test\_2()  
 test\_3()  
 test\_4()

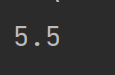
Результат:



Задание 6:

import time  
from contextlib import contextmanager  
  
class cm\_timer\_1:  
 def \_\_init\_\_(self):  
 self.start\_time = time.time()  
 pass  
  
 def \_\_enter\_\_(self):  
 return 0  
  
 def \_\_exit\_\_(self, exp\_type, exp\_value, traceback):  
 if exp\_type is not None:  
 print(exp\_type, exp\_value, traceback)  
 else:  
 print(round(time.time() - self.start\_time, 1))  
  
@contextmanager  
def cm\_timer\_2():  
 start\_time = time.time()  
 yield 0  
 print(round(time.time() - start\_time, 1))  
  
with cm\_timer\_1():  
 time.sleep(5.5)  
  
with cm\_timer\_2():  
 time.sleep(5.5)

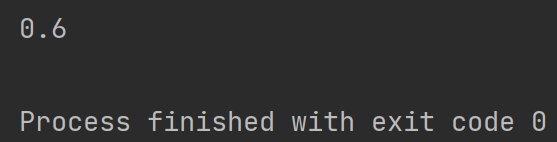
Результат:



Задание 7:

import json,random,sys,time  
from contextlib import contextmanager  
  
path ='data.json'  
with open(path, encoding='utf-8', errors='ignore') as f:  
 data = json.load(f)  
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''  
class Unique(object):  
 def \_\_init\_\_(self, items, \*\*kwargs):  
 self.items = items  
 for k, v in kwargs.items():  
 setattr(self, k, v)  
 if not hasattr(self, "ignore\_case"):  
 self.ignore\_case=False  
  
 def \_\_next\_\_(self):  
 if self.ignore\_case:  
 li=[]  
 for x in self.items: li.append(x.lower())  
 else:  
 li = self.items  
  
 self.items = []  
 for i in li:  
 if i not in self.items:  
 self.items.append(i)  
 return self.items  
  
 def \_\_iter\_\_(self):  
 return self  
  
def gen\_random(num\_count, begin, end):  
 a = []  
 for i in range(num\_count): a.append(random.randint(begin, end))  
 return a  
  
class cm\_timer\_1:  
 def \_\_init\_\_(self):  
 self.start\_time = time.time()  
 pass  
  
 def \_\_enter\_\_(self):  
 return 0  
  
 def \_\_exit\_\_(self, exp\_type, exp\_value, traceback):  
 if exp\_type is not None:  
 print(exp\_type, exp\_value, traceback)  
 else:  
 print(round(time.time() - self.start\_time, 1))  
  
  
def N2(data):  
 res=[]  
 for x in data:  
 if x.startswith("программист"):  
 res.append(x)  
 return res  
  
def N3(data, start=' с опытом Python'):  
 return data+start  
  
def N4(data):  
 a=gen\_random(len(data),100000,200000)  
 l=[]  
 for i in a:  
 l.append("зарплата "+str(i))  
 res=zip(data,l)  
 return res  
  
def field(items, \*args):  
 res=[]  
 assert len(args) > 0  
 for a in range(len(args)):  
 for i in range(len(items)):  
 try: res.append(items[i][args[a]])  
 except: pass  
 return res  
  
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''  
def f1(arg):  
 return next(Unique(field(arg, "job-name"), ignore\_case=True))  
  
def f2(arg):  
 return N2(arg)  
  
def f3(arg):  
 return list(map(N3, arg))  
  
def f4(arg):  
 return N4(arg)  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 with cm\_timer\_1():  
 time.sleep(0.5)  
 f4(f3(f2(f1(data))))

Результат:



Задание 7:

import json,random,sys,time  
from contextlib import contextmanager  
  
path ='data.json'  
with open(path, encoding='utf-8', errors='ignore') as f:  
 data = json.load(f)  
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''  
class Unique(object):  
 def \_\_init\_\_(self, items, \*\*kwargs):  
 self.items = items  
 for k, v in kwargs.items():  
 setattr(self, k, v)  
 if not hasattr(self, "ignore\_case"):  
 self.ignore\_case=False  
  
 def \_\_next\_\_(self):  
 if self.ignore\_case:  
 li=[]  
 for x in self.items: li.append(x.lower())  
 else:  
 li = self.items  
  
 self.items = []  
 for i in li:  
 if i not in self.items:  
 self.items.append(i)  
 return self.items  
  
 def \_\_iter\_\_(self):  
 return self  
  
def gen\_random(num\_count, begin, end):  
 a = []  
 for i in range(num\_count): a.append(random.randint(begin, end))  
 return a  
  
class cm\_timer\_1:  
 def \_\_init\_\_(self):  
 self.start\_time = time.time()  
 pass  
  
 def \_\_enter\_\_(self):  
 return 0  
  
 def \_\_exit\_\_(self, exp\_type, exp\_value, traceback):  
 if exp\_type is not None:  
 print(exp\_type, exp\_value, traceback)  
 else:  
 print(round(time.time() - self.start\_time, 1))  
  
  
def N2(data):  
 res=[]  
 for x in data:  
 if x.startswith("программист"):  
 res.append(x)  
 return res  
  
def N3(data, start=' с опытом Python'):  
 return data+start  
  
def N4(data):  
 a=gen\_random(len(data),100000,200000)  
 l=[]  
 for i in a:  
 l.append("зарплата "+str(i))  
 res=zip(data,l)  
 return res  
  
def field(items, \*args):  
 res=[]  
 assert len(args) > 0  
 for a in range(len(args)):  
 for i in range(len(items)):  
 try: res.append(items[i][args[a]])  
 except: pass  
 return res  
  
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''  
def f1(arg):  
 return next(Unique(field(arg, "job-name"), ignore\_case=True))  
  
def f2(arg):  
 return N2(arg)  
  
def f3(arg):  
 return list(map(N3, arg))  
  
def f4(arg):  
 return N4(arg)  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 with cm\_timer\_1():  
 f4(f3(f2(f1(data))))

Результат:

**Изображение выглядит как текст

Автоматически созданное описание**  
**Изображение выглядит как текст

Автоматически созданное описание**