



**Министерство науки и высшего образования Российской
Федерации**
Федеральное государственное бюджетное образовательное учреждение высшего образования
**«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)**

**Отчет по лабораторной работе №3
Курса «Разработка интернет-приложений»**

Студент: Яровенко М.В.
Группа: ИУ5Ц-72Б

Преподаватель: Гапанюк Ю.Е.

2021 г.

Оглавление

Задание	3
Задача 1 (файл field.py)	3
Задача 2 (файл gen_random.py)	3
Задача 3 (файл unique.py)	3
Задача 4 (файл sort.py)	3
Задача 5 (файл print_result.py)	4
Задача 6 (файл cm_timer.py)	4
Задача 7 (файл process_data.py)	4
Текст программы	5
Задача 1 (файл field.py)	5
Задача 2 (файл gen_random.py)	6
Задача 3 (файл unique.py)	7
Задача 4 (файл sort.py)	9
Задача 5 (файл print_result.py)	10
Задача 6 (файл cm_timer.py)	11
Задача 7 (файл process_data.py)	13
Результат выполнения программы	15
Задача 1 (файл field.py)	15
Задача 2 (файл gen_random.py)	15
Задача 3 (файл unique.py)	16
Задача 4 (файл sort.py)	16
Задача 5 (файл print_result.py)	16
Задача 6 (файл cm_timer.py)	16
Задача 7 (файл process_data.py)	17

Задание

Задача 1 (файл `field.py`)

Необходимо реализовать генератор `field`. Генератор `field` последовательно выдает значения ключей словаря.

- В качестве первого аргумента генератор принимает список словарей, дальше через `*args` генератор принимает неограниченное количество аргументов.
- Если передан один аргумент, генератор последовательно выдает только значения полей, если значение поля равно `None`, то элемент пропускается.
- Если передано несколько аргументов, то последовательно выдаются словари, содержащие данные элементы. Если поле равно `None`, то оно пропускается. Если все поля содержат значения `None`, то пропускается элемент целиком.

Задача 2 (файл `gen_random.py`)

Необходимо реализовать генератор `gen_random(количество, минимум, максимум)`, который последовательно выдает заданное количество случайных чисел в заданном диапазоне от минимума до максимума, включая границы диапазона.

Задача 3 (файл `unique.py`)

- Необходимо реализовать итератор `Unique(данные)`, который принимает на вход массив или генератор и итерируется по элементам, пропуская дубликаты.
- Конструктор итератора также принимает на вход именованный `bool`-параметр `ignore_case`, в зависимости от значения которого будут считаться одинаковыми строки в разном регистре. По умолчанию этот параметр равен `False`.
- При реализации необходимо использовать конструкцию `**kwargs`.
- Итератор должен поддерживать работу как со списками, так и с генераторами.
- Итератор не должен модифицировать возвращаемые значения.

Задача 4 (файл `sort.py`)

Дан массив `1`, содержащий положительные и отрицательные числа. Необходимо одной строкой кода вывести на экран массив `2`, который содержит значения массива `1`, отсортированные по модулю в порядке убывания. Сортировку необходимо осуществлять с помощью функции `sorted`.

Необходимо решить задачу двумя способами:

- С использованием lambda-функции.
- Без использования lambda-функции.

Задача 5 (файл `print_result.py`)

Необходимо реализовать декоратор `print_result`, который выводит на экран результат выполнения функции.

- Декоратор должен принимать на вход функцию, вызывать её, печатать в консоль имя функции и результат выполнения, после чего возвращать результат выполнения.
- Если функция вернула список (`list`), то значения элементов списка должны выводиться в столбик.
- Если функция вернула словарь (`dict`), то ключи и значения должны выводиться в столбик через знак равенства.

Задача 6 (файл `cm_timer.py`)

Необходимо написать контекстные менеджеры `cm_timer_1` и `cm_timer_2`, которые считают время работы блока кода и выводят его на экран.

`cm_timer_1` и `cm_timer_2` реализуют одинаковую функциональность, но должны быть реализованы двумя различными способами (на основе класса и с использованием библиотеки `contextlib`).

Задача 7 (файл `process_data.py`)

В файле `data_light.json` содержится фрагмент списка вакансий.

Структура данных представляет собой список словарей с множеством полей: название работы, место, уровень зарплаты и т.д.

Необходимо реализовать 4 функции - `f1`, `f2`, `f3`, `f4`. Каждая функция вызывается, принимая на вход результат работы предыдущей. За счет декоратора `@print_result` печатается результат, а контекстный менеджер `cm_timer_1` выводит время работы цепочки функций.

Предполагается, что функции `f1`, `f2`, `f3` будут реализованы в одну строку. В реализации функции `f4` может быть до 3 строк.

- Функция `f1` должна вывести отсортированный список профессий без повторений (строки в разном регистре считать равными). Сортировка должна игнорировать регистр. Используйте наработки из предыдущих задач.
- Функция `f2` должна фильтровать входной массив и возвращать только те элементы, которые начинаются со слова “программист”. Для фильтрации используйте функцию `filter`.

- Функция f3 должна модифицировать каждый элемент массива, добавив строку “с опытом Python” (все программисты должны быть знакомы с Python). Пример: Программист С# с опытом Python. Для модификации используйте функцию map.

- Функция f4 должна сгенерировать для каждой специальности зарплату от 100 000 до 200 000 рублей и присоединить её к названию специальности. Пример: Программист С# с опытом Python, зарплата 137287 руб. Используйте zip для обработки пары специальность — зарплата.

Текст программы

Задача 1 (файл field.py)

```
def field(items, *args):
    assert len(args) > 0
    if len(args) == 1:
        sl = ""
        for k in items:
            if k[args[0]] != None:
                sl = str(k[args[0]])
                yield sl
    elif len(args) > 1:
        sl = {}
        for k in items:
            for i in range(len(args)):
                if k[args[i]] != None:
                    sl[args[i]] = k[args[i]]
            yield sl
        sl.clear()
```

```
goods = [
```

```
{'title': 'Диван', 'price': 2000, 'color': 'Черный'},  
{'title': 'Кресло', 'price': 500, 'color': 'Красный'},  
{'title': 'Утюг', 'price': 1000, 'color': None},  
{'title': 'Картина', 'price': 1500, 'color': None},  
{'title': 'Стол', 'price': None, 'color': None}  
]
```

```
print("Генератор field(goods, 'title'):")  
for i in field(goods, 'title':):  
    print(i)
```

```
print("\nГенератор field(goods, 'price'):")  
for i in field(goods, 'price':):  
    print(i)
```

```
print("\nГенератор field(goods, 'title', 'price'):")  
for i in field(goods, 'title', 'price':):  
    print(i)
```

```
print("\nГенератор field(goods, 'title', 'color'):")  
for i in field(goods, 'title', 'color':):  
    print(i)
```

```
input("Press Enter to continue...")
```

Задача 2 (файл gen_random.py)

#Задача 2. Яровенко Максим, ИУ5Ц-72Б

```
from random import randint
```

```

def gen_random(kolvo, min, max):                                #генератор
    for k in range(kolvo):
        i = randint(min, max)
        yield i

data = gen_random(15, 1, 10)
for i in data:                                                  #пример работы генератора
    print(i)

input("Press Enter to continue...")

```

Задача 3 (файл unique.py)

#Задача 3. Яровенко Максим, ИУ5Ц-72Б

```

from random import randint

def gen_random(kolvo, min, max):
    for k in range(kolvo):
        i = randint(min, max)
        yield i

class Unique(object):

    def __init__(self, items, **kwargs):
        self.items = items
        if len(kwargs) != 0:
            if len(kwargs) != 0:
                self.ignore_case = kwargs['ignore_case']
            else:
                self.ignore_case = False

```

```
self.count = 0
```

```
self.prsp = []
```

```
def __next__(self):
```

```
    if type(self.items) != list:
```

```
        sp = []
```

```
        for i in self.items:
```

```
            sp.append(i)
```

```
        self.items = sp
```

```
while True:
```

```
    if self.count < len(self.items):
```

```
        element = self.items[self.count]
```

```
        self.count += 1
```

```
        if self.ignore_case:
```

```
            if element.lower() not in self.prsp:
```

```
                self.prsp.append(element.lower())
```

```
                return element
```

```
        else:
```

```
            if element not in self.prsp:
```

```
                self.prsp.append(element)
```

```
                return element
```

```
    else:
```

```
        raise StopIteration
```

```
def __iter__(self):
```

```
    return self
```



```
data = ['ABc', 'ab', 'bca', 'ab', 'abc', 'Ab', 'c']
```

```
for i in Unique(data, ignore_case=True):  
    print(i)
```

```
data = gen_random(15, 1, 5)
```

```
for i in Unique(data):  
    print(i)
```

```
input("Press Enter to continue...")
```

Задача 4 (файл sort.py)

#Задача 4. Яровенко Максим, ИУ5Ц-72Б

```
def sorter(item):  
    return abs(item)
```

```
data = [4, -30, 100, -100, 123, 1, 0, -1, -4]
```

```
print("Реализация сортировки без lambda-функции:")  
result = sorted(data, reverse = True, key = sorter)  
print(result)
```

```
print("\nРеализация сортировки с lambda-функцией:")  
result_with_lambda = sorted(data, reverse = True, key = lambda item: abs(item))  
print(result_with_lambda)
```

```
input("Press Enter to continue...")
```

Задача 5 (файл print_result.py)

#Задача 5. Яровенко Максим, ИУ5Ц-72Б

```
def print_result(func):
```

```
    def other_func(*args, **kwargs):
```

```
        print(func.__name__)
```

```
        vozvrat = func(*args, **kwargs)
```

```
        if type(vozvrat) == list:
```

```
            for i in vozvrat:
```

```
                print(i)
```

```
        elif type(vozvrat) == dict:
```

```
            for k, v in vozvrat.items():
```

```
                print(k, " = ", v)
```

```
        else:
```

```
            print(vozvrat)
```

```
        return func(*args, **kwargs)
```

```
    return other_func
```

```
@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__':
    print("Демонстрация работы декоратора")
    test_1()
    test_2()
    test_3()
    test_4()
```

```
input("Press Enter to continue...")
```

Задача 6 (файл cm_timer.py)

#Задача 6. Яровенко Максим, ИУ5Ц-72Б

```
import time
from contextlib import contextmanager
```

```
class cm_timer_1:
```

```
    def __enter__(self):
        self.time1 = time.perf_counter()
```

```
return None
```

```
def __exit__(self, exp_type, exp_value, traceback):
```

```
    if exp_type is not None:
```

```
        print(exp_type, exp_value, traceback)
```

```
    else:
```

```
        self.time2 = time.perf_counter()
```

```
        print ("time:", self.time2-self.time1)
```

```
@contextmanager
```

```
def cm_timer_2():
```

```
    time1 = time.perf_counter()
```

```
    yield None
```

```
    time2 = time.perf_counter()
```

```
    print ("time:", time2-time1)
```

```
if __name__ == '__main__':
```

```
    print("Демонстрация работы контекстного менеджера, реализованного на  
основе класса:")
```

```
    with cm_timer_1():
```

```
        time.sleep(3)
```

```
    print("\nДемонстрация работы контекстного менеджера, реализованного с  
использованием библиотеки contextlib:")
```

```
    with cm_timer_2():
```

```
        time.sleep(5)
```

```
    input("Press Enter to continue...")
```

Задача 7 (файл process_data.py)

#Задача 7. Яровенко Максим, ИУ5Ц-72Б

```
import json
```

```
import sys
```

```
import random
```

```
from lab_python_fp.cm_timer import *
```

```
from lab_python_fp.print_result import *
```

```
# Сделаем другие необходимые импорты
```

```
path = "D:\data_light.json"
```

```
# Необходимо в переменную path сохранить путь к файлу, который был  
передан при запуске сценария
```

```
with open(path, encoding='utf-8') as f:
```

```
    data = json.load(f)
```

```
# Далее необходимо реализовать все функции по заданию, заменив `raise  
NotImplemented`
```

```
# Предполагается, что функции f1, f2, f3 будут реализованы в одну строку
```

```
# В реализации функции f4 может быть до 3 строк
```

```
@print_result
```

```
def f1(arg):
```

```
    l = sorted(list(set(i['job-name'].capitalize() for i in arg)))
```

```
return l
```

```
@print_result
```

```
def f2(arg):
```

```
    l = list(filter(lambda x: x.split()[0] == "Программист", arg))
```

```
    return l
```

```
@print_result
```

```
def f3(arg):
```

```
    l = list(map(lambda x: x + " с опытом Python", arg))
```

```
    return l
```

```
@print_result
```

```
def f4(arg):
```

```
    l = ["", заплата ".join(i) for i in list(zip(arg, [str(random.randint(100_000, 200_000)) for i in range(len(arg))]))]
```

```
    return l
```

```
if __name__ == '__main__':
```

```
    with cm_timer_1():
```

```
        f4(f3(f2(f1(data))))
```

```
    input("Press Enter to continue...")
```

Результат выполнения программы

Задача 1 (файл field.py)

```
C:\Program Files\Python38\python.exe
Генератор field(goods, 'title'):
Диван
Кресло
Утюг
Картина
Стол

Генератор field(goods, 'price'):
2000
500
1000
1500

Генератор field(goods, 'title', 'price'):
{'title': 'Диван', 'price': 2000}
{'title': 'Кресло', 'price': 500}
{'title': 'Утюг', 'price': 1000}
{'title': 'Картина', 'price': 1500}
{'title': 'Стол'}

Генератор field(goods, 'title', 'color'):
{'title': 'Диван', 'color': 'Черный'}
{'title': 'Кресло', 'color': 'Красный'}
{'title': 'Утюг'}
{'title': 'Картина'}
{'title': 'Стол'}
Press Enter to continue...
```

Задача 2 (файл gen_random.py)

```
C:\Program Files\Python38\python.exe
9
8
4
9
3
2
10
8
7
10
7
6
6
3
4
Press Enter to continue...
```

Задача 3 (файл unique.py)

```
C:\Program Files\Python38\python.exe
ABc
ab
bca
c
2
5
3
4
Press Enter to continue...
```

Задача 4 (файл sort.py)

```
C:\Program Files\Python38\python.exe
Реализация сортировки без lambda-функции:
[123, 100, -100, -30, 4, -4, 1, -1, 0]

Реализация сортировки с lambda-функцией:
[123, 100, -100, -30, 4, -4, 1, -1, 0]
Press Enter to continue..._
```

Задача 5 (файл print_result.py)

```
C:\Program Files\Python38\python.exe
Демонстрация работы декоратора
test_1
1
test_2
iu5
test_3
a = 1
b = 2
test_4
1
2
Press Enter to continue...
```

Задача 6 (файл cm_timer.py)

```
C:\Program Files\Python38\python.exe
Демонстрация работы контекстного менеджера, реализованного на основе класса:
time: 3.0053450999999995

Демонстрация работы контекстного менеджера, реализованного с использованием библиотеки contextlib:
time: 5.009348999999999
Press Enter to continue..._
```


Задача 7 (файл process_data.py)

Начало выполнения f1. Так как данных много, то приводится лишь начальный фрагмент выполнения данной функции.

```
C:\Program Files\Python38\python.exe
f1
1с программист
2-ой механик
3-ий механик
4-ый механик
4-ый электромеханик
Asic специалист
Javascript разработчик
Rtl специалист
web-программист
web-разработчик
[химик-эксперт
Автокстоящик
Автоинструктор
Автомалар
Автомойщик
Автор студенческих работ по различным дисциплинам
Автослесарь
Автослесарь - моторист
Автоэлектрик
Агент
Агент банка
Агент нпф
Агент по гос. закупкам недвижимости
Агент по недвижимости
Агент по недвижимости (стажер)
Агент по недвижимости / риэлтор
Агент по привлечению юридических лиц
Агент по продажам (интернет, тв, телефония) в пао ростелеком в населенных пунктах амурской области: г. благовещенск, г. белогорск, г. свободный, г. шимановск, г. зeya, г. тында
Агент торговый
Агрегатчик-топливник komatsu
Агроном
Агроном по защите растений
Агроном-полевод
Агрохимик почвовед
Администратор
Администратор (удаленно)
Администратор active directory
Администратор в парикмахерский салон
Администратор зала (предприятий общественного питания)
Администратор кофейни
Администратор на ресепшен
Администратор на телефоне
```

```
C:\Program Files\Python38\python.exe
Электрослесарь по ремонту и обслуживанию автоматики и средств измерений электростанций
Электрослесарь по ремонту оборудования в карьере
Электроэрозионист
Эндокринолог
Энергетик
Энергетик литейного производства
Энтомолог
Юрисконсульт
Юрисконсульт 2 категории
Юрисконсульт. контрактный управляющий
Юрист
Юрист (специалист по сопровождению международных договоров, английский - разговорный)
Юрист волонтер
Юристконсульт
f2
Программист
Программист / senior developer
Программист 1с
Программист c#
Программист c++
Программист c++/c#/java
f3
Программист с опытом Python
Программист / senior developer с опытом Python
Программист 1с с опытом Python
Программист c# с опытом Python
Программист c++ с опытом Python
Программист c++/c#/java с опытом Python
f4
Программист с опытом Python, зарплата 183745
Программист / senior developer с опытом Python, зарплата 164899
Программист 1с с опытом Python, зарплата 182881
Программист c# с опытом Python, зарплата 137886
Программист c++ с опытом Python, зарплата 164062
Программист c++/c#/java с опытом Python, зарплата 110075
time: 0.16755469999999995
Press Enter to continue...
```