Бирюкова Екатерина.
Лабораторная работа № 4-5.

Условие.

Задача 1 (файл field.py)
Необходимо реализовать генератор field. Генератор field последовательно выдает значения ключей словаря. Пример:
goods = [{'title': 'Ковер', 'price': 2000, 'color': 'green'}, {'title': 'Диван для отдыха', 'color': 'black'}]
field(goods, 'title') должен выдавать 'Ковер', 'Диван для отдыха'
field(goods, 'title', 'price') должен выдавать {'title': 'Ковер', 'price': 2000}, {'title': 'Диван для отдыха'}

- В качестве первого аргумента генератор принимает список словарей, дальше через *args генератор принимает неограниченное количествово 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. Пример:
gen_random(5, 1, 3) должен выдать 5 случайных чисел в диапазоне от 1 до 3, например 2, 2, 3, 2, 1

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, которые содержит значения массива 1, отсортированные по модулю в порядке убывания. Сортировку необходимо осуществлять с помощью функции sorted. Пример:
data = [4, -30, 30, 100, -100, 123, 1, 0, -1, -4]
Вывод: [123, 100, -100, -30, 30, 4, -4, 1, -1, 0]
Необходимо решить задачу двумя способами:

1. С использованием lambda-функции.
2. 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 выводить 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. Пример:

```
with cm_timer_1():
    sleep(5.5)
```

После завершения блока кода в консоль должно вывестись time: 5.5 (реальное время может несколько отличаться).
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 предыдущих задачах были написаны все требуемые инструменты для работы с данными. Применим их на реальном примере.
- 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 C# 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 C# с опытом Python, зарплата 137287 руб. Используйте zip для обработки пары специальность — зарплата.

Текст программы.

Задача 1 (файл field_file.py)

```
#!/usr/bin/python
# -*- coding: cp1251 -*-
```

```
def field(goods,*args):
    fields = args
    for good in goods:
        result = dict([(field, good[field]) for field in fields if good.get(field) is not None])
        if any(result.values()):
            yield result

if __name__ == '__main__':
    goods = [
        {'title': 'Ковер', 'price': 2000, 'color': 'green'},
        {'title': 'Диван для отдыха', 'color': 'black'}
    ]

    for result in field(goods, "title"):
        print(result)
    print(" ")
    for result in field(goods, "title", "price"):
        print(result)
```

Задача 2 (файл gen_random_file.py)

```
import random

def gen_random(num_count, begin, end):
    for i in range(num_count):
        yield random.randint(begin, end)

if __name__ == '__main__':
    for num in gen_random(5, 1, 4):
        print(num)
```

Задача 3 (файл unique_file.py)

```
#!/usr/bin/python
# -*- coding: cp1251 -*-
class Unique:
    def __init__(self, data, **params):
        self.used_elements = set()
        self.data = data
        self.index = 0
        if len(params)>0:
            for param,valueOfParam in params.items():
                self.register = valueOfParam
        else:
            self.register = False
        #Если True - игнорируем регистр, False - не игнорируем

    def __iter__(self):
        return self

    def __next__(self):
        while True:
            if self.index >= len(self.data):
                raise StopIteration
            else:
                current = self.data[self.index]
                self.index = self.index + 1
                flag = str(current).isdigit()
                if self.register == True and flag == False:
                    if current not in self.used_elements and current.upper() not in self.used_elements and current.lower() not in
self.used_elements:
                        self.used_elements.add(current)
                        return current
                else:
                    if current not in self.used_elements:
                        self.used_elements.add(current)
                        return current
```

```
if __name__ == '__main__':
    lst1 = ['a', 'A', 'b', 'B', 'a', 'A', 'b', 'B']
    lst2 = [1,3,2,3,2,1,4,3,3,3]
    lst3 = ['a', 'A', 'b', 'B', 'a', 'A', 'b', 'B',1,3,2,3,2,1,4,3,3,3]
    for i in Unique(lst1, register=False):
        print(i)
    print(' ')
    for i in Unique(lst1, register=True):
        print(i)
    print(' ')
    for i in Unique(lst2):
        print(i)
    print(' ')
    for i in Unique(lst3, register=False):
        print(i)
    print(' ')
    for i in Unique(lst3, register=True):
        print(i)
```

Задача 4 (файл sort.py)

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

print(' ')

result = sorted(data, key=lambda x: abs(x), reverse=True)
print(result)
result = sorted(data, key=abs, reverse=True)
print(result)
```

Задача 5 (файл print_result.py)

```
#!/usr/bin/python
# -*- coding: cp1251 -*-

def print_result(func):
    def wrapper(*args, **kwargs):
        result = func(*args, **kwargs)
        print(f"Вызывается функция: {func.__name__}")
        if isinstance(result, list):
            for item in result:
                print(item)
        elif isinstance(result, dict):
            for key, value in result.items():
                print(f"{key} = {value}")
        else:
            print(result)
        return result
    return wrapper

@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()
```

Задача 6 (файл cm_timer.py)

```
import time
from contextlib import contextmanager

class cm_timer_1:
    def __enter__(self):
        self.start_time = time.time()

    def __exit__(self, exc_type, exc_val, exc_tb):
        end_time = time.time()
        exec_time = end_time - self.start_time
        print("time_1:", exec_time)

@contextmanager
def cm_timer_2():
    start_time = time.time()
    yield
    end_time = time.time()
    exec_time = end_time - start_time
    print("time_2:", exec_time)

if __name__ == '__main__':
    with cm_timer_1():
        time.sleep(2.2)
    with cm_timer_2():
        time.sleep(4.4)
```

Задача 7 (файл process_data.py)

```
#!/usr/bin/python
# -*- coding: cp1251 -*-

import json
import sys
from unique_file import Unique
from print_result import print_result
from cm_timer import cm_timer_1
from field_file import field
import random

def f1(arg):
    mas_1=[]
    mas_2=[]
    for result in field(arg, 'job-name'):
        mas_1.append(result['job-name'])
    for i in Unique(mas_1, register=True):
        mas_2.append(i)
    return sorted(mas_2)

def check(elem):
    if elem[:11].upper() == "ПРОГРАММИСТ":
        return True
    else:
        return False

def f2(arg):
  return list(filter(check, arg))

def add_stroka(elem):
    return str(elem+" с опытом python")
```

```
def f3(arg):
  return list(map(add_stroka, arg))

def add_stroka_1(elem):
   return str(elem+" , зарплата " + str(random.randint(100000,200000)) + " руб")

@print_result
def f4(arg):
  return list(map(add_stroka_1, arg))


with open("./data_light.json",encoding='utf-8') as f:
   data = json.load(f)
   with cm_timer_1():
      print("НАЧАЛО ЗАДАНИЕ 1")
      f1(data)
      print("КОНЕЦ ЗАДАНИЕ 1")
   print(" ")
   with cm_timer_1():
      print("НАЧАЛО ЗАДАНИЕ 2")
      f2(f1(data))
      print("КОНЕЦ ЗАДАНИЕ 2")
   print(" ")
   with cm_timer_1():
      print("НАЧАЛО ЗАДАНИЕ 3")
      f3(f2(f1(data)))
      print("КОНЕЦ ЗАДАНИЕ 3")
   with cm_timer_1():
      print("НАЧАЛО ЗАДАНИЕ 4")
      f4(f3(f2(f1(data))))
      print("КОНЕЦ ЗАДАНИЕ 4")
```

Результаты выполнения.

Задача 1 (файл field_file.py)

```
{'title': 'Ковер'}
{'title': 'Диван для отдыха'}

{'title': 'Ковер', 'price': 2000}
{'title': 'Диван для отдыха'}
```

Задача 2 (файл gen_random_file.py)

```
1
2
2
4
2
```

Задача 3 (файл unique_file.py)

```
a
A
b
B

a
b

1
3
2
4

a
A
b
B
1
3
2
4

a
b
1
3
2
4
```

Задача 4 (файл sort.py)

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

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

Задача 5 (файл print_result.py)

```
Вызывается функция: test_1
1
Вызывается функция: test_2
iu5
Вызывается функция: test_3
a = 1
b = 2
Вызывается функция: test_4
1
2
```

Задача 6 (файл cm_timer.py)

```
time_1: 2.201469659805298
time_2: 4.407463550567627
```

Задача 7 (файл process_data.py)

```
НАЧАЛО ЗАДАНИЕ 1
КОНЕЦ ЗАДАНИЕ 1
time_1: 0.011969566345214844

НАЧАЛО ЗАДАНИЕ 2
КОНЕЦ ЗАДАНИЕ 2
time_1: 0.011860370635986328

НАЧАЛО ЗАДАНИЕ 3
КОНЕЦ ЗАДАНИЕ 3
time_1: 0.012994766235351562

НАЧАЛО ЗАДАНИЕ 4
Вызывается функция: f4
Программист с опытом python , зарплата 116150 руб
Программист / Senior Developer с опытом python , зарплата 113009 руб
Программист 1С с опытом python , зарплата 188852 руб
Программист C# с опытом python , зарплата 168021 руб
Программист C++ с опытом python , зарплата 132520 руб
Программист C++/C#/Java с опытом python , зарплата 195881 руб
Программист/ Junior Developer с опытом python , зарплата 102315 руб
Программист/ технический специалист с опытом python , зарплата 102849 руб
Программистр-разработчик  информационных систем с опытом python , зарплата 194579 руб
программист с опытом python , зарплата 194415 руб
программист 1С с опытом python , зарплата 105123 руб
КОНЕЦ ЗАДАНИЕ 4
time_1: 0.015525341033935547
```