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)

Факультет «Информатика и системы управления»
Кафедра «Системы обработки информации и управления»

Расулов Арсен ИУ5-35Б

Парадигмы и конструкции языков программирования

ОТЧЁТ ПО
Лабораторной работе №3.
Функциональные возможности языка Python.

Москва
2023

Задание.

Задание лабораторной работы состоит из решения нескольких задач.

Файлы, содержащие решения отдельных задач, должны располагаться в пакете lab_python_fp. Решение каждой задачи должно раполагаться в отдельном файле.

При запуске каждого файла выдаются тестовые результаты выполнения соответствующего задания.

- field.py: Необходимо реализовать генератор field. Генератор field последовательно выдает значения ключей словаря.
- 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.
- unique.py: Необходимо реализовать итератор Unique(данные), который принимает на вход массив или генератор и итерируется по элементам, пропуская дубликаты.
- 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.
- print_result.py: Необходимо реализовать декоратор print_result, который выводит на экран результат выполнения функции.
- 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.
- process_data.py: 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.

Текст программы.

field.py

```
def field(dicts, *args):
    assert len(args) > 0

    if len(args) == 1:
        key = args[0]
        for i in dicts:
            if i.get(key):
                yield i[key]
    else:
        for dict_ in dicts:
            if all(not dict_.get(i) for i in args):
```

```
                continue
            res = {}
            for i in args:
                if dict_.get(i):
                    res[i] = dict_.get(i)
            yield res


if __name__ == '__main__':
    goods = [
        {'title': 'Ковер', 'price': 2000, 'color': 'green'},
        {'title': 'Диван для отдыха', 'color': 'black'}
    ]
    a = field(goods, 'title', 'price')  # должен выдавать {'title': 'Ковер',
'price': 2000}, {'title': 'Диван для отдыха', 'price': 5300}
    print(next(a))
    print(next(a))
    print(next(a))
```

gen_random.py

```
import random


# Пример:
# gen_random(5, 1, 3) должен выдать выдать 5 случайных чисел
# в диапазоне от 1 до 3, например 2, 2, 3, 2, 1
# Hint: типовая реализация занимает 2 строки
def gen_random(num_count, begin, end):
    random.seed()

    for _ in range(num_count):
        yield random.randint(begin, end)


if __name__ == '__main__':
    a = gen_random(5, 1, 3)
    print(next(a))
    print(next(a))
    print(next(a))
    print(next(a))
    print(next(a))
```

unique.py

```
class Unique(object):
    def __init__(self, items, **kwargs):
        ignore_case = bool(kwargs.get('ignore_case'))
        self.items = []
        used_items = set()
        for i in items:
            if ignore_case and i.lower() not in used_items:
                self.items.append(i)
                used_items.add(i.lower())
            elif not ignore_case and i not in used_items:
                self.items.append(i)
                used_items.add(i)
        self.index = 0
```

```
    def __next__(self):
        if self.index < len(self.items):
            res = self.items[self.index]
            self.index += 1
            return res
        raise StopIteration

    def __iter__(self):
        self.index = 0
        return self


if __name__ == '__main__':
    data = ['a', 'A', 'b', 'B', 'a', 'A', 'b', 'B']

    a = Unique(data, ignore_case=True)
    print(next(a))
    print(next(a))
```

sort.py

```
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)
```

print_result.py

```
# Здесь должна быть реализация декоратора
def print_result(func):
    def wrapper(*args, **kwargs):
        res = func(*args, **kwargs)
        print(func.__name__)

        if type(res) == list:
            print(*res, sep='\n')
        elif type(res) == dict:
            for k, w in res.items():
                print(f'{k} = {w}')
        else:
            print(res)
        return res
    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__':
    print('!!!!!!!!!')
    test_1()
    test_2()
    test_3()
    test_4()
```

cm_timer.py

```
import contextlib
import time
from datetime import timedelta


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()
        print("Execution time in seconds:", timedelta(seconds=end_time - self.s-
tart_time).seconds)
        return False


@contextlib.contextmanager
def cm_timer_2():
    start_time = time.time()
    yield start_time
    end_time = time.time()
    print("Execution time in seconds: {}".format(timedelta(seconds=end_time -
start_time).seconds))


if __name__ == '__main__':
    with cm_timer_1():
        time.sleep(5)

    with cm_timer_2() as start_time:
        time.sleep(5)
```

process_data.py

```
import json

import unique
import field
import gen_random
```

```
from print_result import print_result
from cm_timer import cm_timer_1

path = './data_light.json'

with open(path, encoding='utf-8') as f:
    data = json.load(f)


@print_result
def f1(arg):
    return sorted(unique.Unique(field.field(data, 'job-name'),
ignore_case=True))


@print_result
def f2(arg):
    return list(filter(lambda x: x.lower().startswith("программист"), arg))


@print_result
def f3(arg):
    return list(map(lambda x: (x + " с опытом Python"), arg))


@print_result
def f4(arg):
    salaries = gen_random.gen_random(len(arg), 100_000, 200_000)
    result = []
    for job, salary in zip(arg, salaries):
        result.append(f"{job}, зарплата {salary} руб")
    return result


if __name__ == '__main__':
    with cm_timer_1():
        f4(f3(f2(f1(data))))
```

Экранные формы:

field.py

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

gen_random.py

```
3
1
2
3
3
```

unique.py

```
a
b
```

sort.py

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

print_result.py

```
!!!!!!!!
test_1
1
test_2
iu5
test_3
a = 1
b = 2
test_4
1
2
```

cm_timer.py

```
Execution time in seconds: 5
Execution time in seconds: 5
```

process_data.py

```
f1
1С программист
2-ой механик
3-ий механик
4-ый механик
4-ый электромеханик
ASIC специалист
JavaScript разработчик
RTL специалист
Web-программист
[химик-эксперт
web-разработчик
Автожестянщик
Автоинструктор
Автомаляр
Автомойщик
Автор студенческих работ по различным дисциплинам
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. Зея, г. Тында
Агент торговый
```

```
f2
Программист
Программист / Senior Developer
Программист 1С
Программист C#
Программист C++
Программист C++/C#/Java
Программист/ Junior Developer
Программист/ технический специалист
Программистр-разработчик  информационных систем
```

```
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
Программист/ Junior Developer с опытом Python
Программист/ технический специалист с опытом Python
Программистр-разработчик  информационных систем с опытом Python
```

```
f4
Программист с опытом Python, зарплата 188613 руб
Программист / Senior Developer с опытом Python, зарплата 110507 руб
Программист 1С с опытом Python, зарплата 187479 руб
Программист C# с опытом Python, зарплата 139514 руб
Программист C++ с опытом Python, зарплата 142585 руб
Программист C++/C#/Java с опытом Python, зарплата 190319 руб
Программист/ Junior Developer с опытом Python, зарплата 126768 руб
Программист/ технический специалист с опытом Python, зарплата 136483 руб
Программистр-разработчик  информационных систем с опытом Python, зарплата 174384 руб
Execution time in seconds: 0
```