Московский государственный технический университет им. Н.Э. Баумана.

Факультет «Информатика и управление»
Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Базовые компоненты интернет-технологий»

Отчет по лабораторной работе № 3
«Функциональные возможности языка Python»

Выполнил:
студент группы ИУ5-31Б
Абуховский Иван Александрович

Проверил:
преподаватель каф. ИУ5
Гапанюк Юрий Евгеньевич

Подпись и дата:

Подпись и дата:

Москва, 2021 г.

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 располагаться в отдельном файле. При запуске каждого файла выдаются тестовые результаты выполнения соответствующего задания.

Задача 1

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

```
# Пример:
# goods = [
#    {'title': 'Ковер', 'price': 2000, 'color': 'green'},
#    {'title': 'Диван для отдыха', 'price': 5300, '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', 'price': 5300}

goods = [
    {'title': 'Ковер', 'price': 2000, 'color': 'зеленый'},
    {'title': 'Диван для отдыха', 'color': 'черный'},
    {'title': 'Кресло', 'price': 7000, 'color': 'желтый'},
    {'class': 'human', 'name': 'Bob'}
]


def field(items, *args):
    assert (len(args) > 0)
    if len(args) == 1:
        for arg in args:
            for i in range(len(items)):
                for it in items[i]:
                    if (it == arg) and (items[i][it]) != None:
                        yield items[i][it]
    else:
        for i in range(len(items)):
            d = {}
            for arg in args:
                for it in items[i]:
                    if it == arg:
                        d[it] = items[i][it]
            yield d

if __name__ == '__main__':

    Vas = field(goods, "title", "price", "color")
    for i in Vas:
         print(i)
    print()
```

```
{'title': 'Ковер', 'price': 2000, 'color': 'зеленый'}
{'title': 'Диван для отдыха', 'color': 'черный'}
{'title': 'Кресло', 'price': 7000, 'color': 'желтый'}
```

Задача 2

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 выдать 5 случайных чисел
# в диапазоне от 1 до 3, например 2, 2, 3, 2, 1
# Hint: типовая реализация занимает 2 строки

from random import randint

def gen_random(num_count, begin, end):
    n = 0
    while True:
        if n < num_count:
            n += 1
            yield randint(begin, end)
        else:
            break

if __name__ == '__main__':

    a = gen_random(5, 1, 3)
    while True:
        try:
            print(next(a), end=' ')
        except StopIteration:
            print()
            break
```

```
2 1 3 3 3

Process finished with exit code 0
```

Задача 3

```
from lab_python_fp.field import field

goods = [
    {'title': 'Ковер', 'price': 2000, 'color': 'зеленый'},
    {'title': 'Диван для отдыха', 'color': 'черный'},
    {'title': 'Кресло', 'price': 7000, 'color': 'желтый'},
```

```
    {'title': 'Ковер', 'price': 2000, 'color': 'зеленый'}
]

def unify(v):
    return str(v).lower().strip()

class Unique(object):
    n = 0
    def __init__(self, items, ignore_case=False):
        # Нужно реализовать конструктор
        # В качестве ключевого аргумента, конструктор должен принимать bool-параметр
ignore_case,
        # в зависимости от значения которого будут считаться одинаковыми строки в разном
регистре
        # Например: ignore_case = True, Абв и АБВ - разные строки
        #           ignore_case = False, Абв и АБВ - одинаковые строки, одна из которых
удалится
        # По умолчанию ignore_case = False
        pass
        self.ignore_case = ignore_case
        self.data = items
        self.dict = []
        if ignore_case == False:
            for i in items:
                if i not in self.dict:
                    self.dict.append(i)
        else:
            for i in items:
                if unify(i) not in self.dict:
                    self.dict.append(unify(i))



    def __next__(self):
        if self.n < len(self.dict):
            x = self.dict[self.n]
            self.n += 1
            return x
        else:
            raise StopIteration

    def __iter__(self):
        return self

if __name__ == '__main__':

    data1 = ['dDDdDa', "AAA", 'bbb', 'aaa', 'CCccC', 'aaa', 'ccccc']
    data2 = [8, 7, 7, 1, 1, 2, 1, 3, 4, 5, 6]
    for u in Unique(data1, ignore_case=True):
        print(u, end=' ')
    print('\n')
    for u in Unique(data1):
        print(u, end=' ')
    print('\n')
    for u in Unique(data2):
        print(u, end=' ')
```

```
dddddda aaa bbb ccccc

dDDdDa AAA bbb aaa CCccC ccccc

8 7 1 2 3 4 5 6
Process finished with exit code 0
```

Задача 4

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. Необходимо решить задачу двумя способами:

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

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


if __name__ == '__main__':
    result = sorted((data.copy()), key=abs, reverse=True)
    print(result)

    result_with_lambda = (lambda x: sorted((x), key=abs, reverse=True))(data.copy())
    print(result_with_lambda)
```

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

Задача 5

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.

```
def print_result(funct):
```

```
    def wrapper(*args, **kwargs):
        print(funct.__name__)
        res = funct(*args, **kwargs)
        if isinstance(res, list):
            for i in res:
                print(i)
        elif isinstance(res, dict):
            for i in res:
                print(i, '=', res[i])
        else:
            print(res)
        return funct(*args, **kwargs)

    return wrapper

# Здесь должна быть реализация декоратора

@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()
```

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

Задача 6

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).

```python
import time

class cm_timer_1:

    def __init__(self):
        self.start_time=time.time()

    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('time=' + str(time.time() - self.start_time))


from contextlib import contextmanager
@contextmanager
def cm_timer_2():
    start_time=time.time()
    yield 0
    print('time='+str(time.time()-start_time))



if __name__ == '__main__':
    with cm_timer_1():
        time.sleep(1.5)

    with cm_timer_2():
        time.sleep(1.0)
```

```
time=1.5013210773468018
time=1.0139579772949219
```

Задача 7

- 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.

```
import json
import sys
from print_result import *
from cm_timer import *
from unique import Unique as uniqum
from gen_random import gen_random as gRand
path = "data_light.json"


with open(path, 'r', 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


def IT_filter(el):
    if el['job-name'][0:11].lower() == 'программист':
        return True
    else:
        return False


@print_result
def f1(arg):
    return [u for u in uniqum([el.get('job-name') for el in arg], ignore_case=True)]
@print_result
def f2(arg):
    return list(filter(lambda el: el[0:11].lower() == 'программист', arg))

@print_result
def f3(arg):

   return list(el+' с опытом Python' for el in arg)

@print_result
def f4(arg):
    zipped=list(zip(arg, gRand(len(arg), 100000, 200000)))
    return [x+' с зарплатой '+str(y) for x, y in zipped]
```

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

```
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
формовщик
фтизиатрия
художник-постановщик
швея - мотористка
шиномонтаж
шлифовщик 5 разряда
шлифовщик механического цеха
эколог
электромонтер -линейщик по монтажу воздушных линий высокого напряжения и контактной сети
электромонтер по испытаниям и измерениям 4-6 разряд
электромонтер станционного телевизионного оборудования
электросварщик
энтомолог
юрисконсульт 2 категории
```

```
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 с зарплатой 167825
Программист / Senior Developer с опытом Python с зарплатой 171825
Программист 1С с опытом Python с зарплатой 169768
Программист C# с опытом Python с зарплатой 112615
Программист C++ с опытом Python с зарплатой 141772
Программист C++/C#/Java с опытом Python с зарплатой 173686
Программист/ Junior Developer с опытом Python с зарплатой 173748
Программист/ технический специалист с опытом Python с зарплатой 161970
Программистр-разработчик  информационных систем с опытом Python с зарплатой 170644
time=7.592022657394409

Process finished with exit code 0
```