Московский государственный технический университет им. Н.Э. Баумана
Факультет «Информатика и системы управления»
Кафедра «Системы обработки информации и управления»

Отчёт по Лабораторной работе №3
“Функциональные возможности языка Python”

Отчёт
(вид документа)

Листы А4
(вид носителя)

9
(количество листов)

Исполнитель: Студент группы ИУ5-54Б
Савельев Алексей
Александрович

Москва - 2020

Цель работы: Изучения функциональных возможностей языка Python

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 (файл field.py)

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

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, которые содержит значения массива 1, отсортированные по модулю в порядке убывания. Сортировку необходимо осуществлять с помощью функции sorted.

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.

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.

Листинг программы

- field.py

```
def field(items, *args):
    assert len(args) > 0, 'No Args'
    if len(args) > 1:
        for item in items:
            dict = {}
            for arg in args:
                if arg in item and item[arg] is not None:
                    dict[arg] = item[arg]
            if len(dict.keys()) > 0:
                yield dict
    elif len(args) == 1:
        for item in items:
            for arg in args:
                if arg in item and item[arg] is not None:
                    yield item[arg]

def main():
    print(list(field(goods, 'title', 'price')))
    print(list(field(goods, 'color')))
    print(list(field(goods, 'title')))

goods = [
        {'title': 'Ковер', 'price': 2000, 'color': 'green'},
        {'title': 'Диван для отдыха', 'color': 'black'},
        {'title': None, 'price': None},
    ]

if __name__ == '__main__':
    main()
```

- gen_random.py

```
import random

# функция рандомайзер
def gen_random(num_count, begin, end):
    result = []     # лист для результата
    if begin > end: temp = begin; begin = end; end = temp; # проверка границ
интервала (корректирование)
    for i in range(0, num_count):
        result.append(random.randint(begin, end))
    return result

def main():
    print(gen_random(10, 0, 12))
    print(gen_random(15, 0, 1))
    print(gen_random(5, -5, 0))

if __name__ == "__main__":
    main()
```

- unique.py

```
class Unique():
    def __init__(self, items, **kwargs):
        ignore = kwargs["ignore"] if "ignore" in kwargs else False
        self.uniqueItems = set()
        for i in items:
            if isinstance(i, str) and ignore:
                self.uniqueItems.add(i.lower())
            else:
                self.uniqueItems.add(i)

    def __next__(self):
        if len(self.uniqueItems) > 0:
            return self.uniqueItems.pop()
        raise StopIteration()

    def __iter__(self):
        return self

def main():
    print("\033[33mlist1\033[0m:         ", list1)
    print("\033[36mUnique list1\033[0m: ", sorted(list(Unique(list1))), "\n")

    print("\033[33mlist2\033[0m:         ", list2)
    print("\033[36mUnique list2\033[0m: ", sorted(list(Unique(list2,
ignore=False))), "\n")
    print("\033[33mlist2\033[0m:                          ", list2)
    print("\033[36mUnique list2\033[0m(\033[31mignore = True\033[0m):  ",
sorted(list(Unique(list2, ignore=True))), "\n")

    print("\033[33mlist3\033[0m:         ", list3)
    print("\033[36mUnique list3\033[0m: ", list(Unique(list3, ignore=False)),
"\n")
    print("\033[33mlist3\033[0m:                          ", list3)
    print("\033[36mUnique list3\033[0m(\033[31mignore = True\033[0m):  ",
list(Unique(list3, ignore=True)),
          "\n")


list1 = [1,2,3,4,4,3,2,4,5,3,6,7,8]
list2 = ['a','B','c','d','c','e','a','B','C','A','b']
list3 = ['a','B',1,'c','d',3,'c','e',3,'a','B',2,'C',5,'A',5,'b']


if __name__ == "__main__":
    main()
```

- sort.py

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

def 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)
```

```
if __name__ == '__main__':
    main()
```

- cm_timer.py

```
from time import time, sleep
from contextlib import contextmanager


class cm_timer_1:

    def __enter__(self):
        self.start = time()
    def __exit__(self, type, value, traceback):
        self.end = time()
        print(f"\033[33mЗатраченное время\033[0m: \033[36m{(self.end -
self.start)}\033[0m")

@contextmanager
def cm_timer_2() -> None:
    begin = time()
    yield
    calcTime = time() - begin
    print(f"\033[33mЗатраченное время\033[0m: \033[36m{calcTime}\033[0m")

def main():
    with cm_timer_1():
        testList = [x for x in range(100_000)]

    with cm_timer_1():
        sleep(1)

    with cm_timer_2():
        testList = [x for x in range(100_000_000)]

    with cm_timer_2():
        sleep(1.2)

if __name__ == "__main__":
    main()
```

- print_result.py

```
def print_result(func):
  def wrapper(*args, **kwargs):
    res = func(*args, **kwargs)
    print("\033[31m" + func.__name__ + "\033[0m")
    if isinstance(res, dict):
      for k in res.keys():
        print("{} = {}".format(k, res[k]))
    elif isinstance(res, list):
      for val in res:
        print(val)
    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]


def main():
    print('!!!!!!!!')
    test_1()
    test_2()
    test_3()
    test_4()


if __name__ == '__main__':
    main()
```

- process_file.py

```
from LR3.lab_python_fp import print_result, unique, gen_random, field, cm_timer
import json

# Путь к json файлу
path = "/Users/dlnwlkmn/Desktop/dia-labs/LR3/data_light.json"

# парсим файл, получется список словарей
with open(path) as f, cm_timer.cm_timer_1():
    data = json.load(f)


@print_result.print_result
def f1(arg):
    # return [unique.Unique(field.field(i, 'job-name'), ignore=False) for i in
data]
    # return sorted(set(line["job-name"].lower() for line in arg))
    # Используем ранее разработанные функции, возващаем отсортированный набор
названий вакансий
    return sorted(set(i for i in unique.Unique(field.field(data, 'job-name'),
ignore=False)))


@print_result.print_result
def f2(arg):
    # отбираем только те названия, которые начинаются со слова "программист"
    return list(filter(lambda x: x.startswith("программист"), arg))


@print_result.print_result
```

```python
def f3(arg):
    # используя map дописываем "с опытом работы" к каждой вакансии
    return list(map(lambda x: x + " с опытом Python", arg))


@print_result.print_result
def f4(arg):
    # дописываем ЗП к каждой вакансии
    return list(i + f", зарплата
\033[34m{gen_random.gen_random(1,100000,200000)[0]}\033[0m руб." for i in arg)


def main():
    # вызываем функции с таймером
    with cm_timer.cm_timer_1():
        f4(f3(f2(f1(data))))

# точка входа
if __name__ == '__main__':
    main()
```

Результат в консоли

- field.py

```
[  3:16 ]  [ dlnwlkmn@MacBook-Pro-Aleksej:~/Desktop/dia-labs/LR3/lab_python_fp(masterx) ]
 $ python3 field.py
[{'title': 'Ковер', 'price': 2000}, {'title': 'Диван для отдыха'}]
['green', 'black']
['Ковер', 'Диван для отдыха']
```

- gen_random.py

```
[  3:17 ]  [ dlnwlkmn@MacBook-Pro-Aleksej:~/Desktop/dia-labs/LR3/lab_python_fp(masterx) ]
 $ python3 gen_random.py
[8, 8, 1, 11, 5, 0, 0, 7, 9, 10]
[1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0]
[-3, 0, -3, -5, -3]
```

- unique.py

```
[  3:17 ]  [ dlnwlkmn@MacBook-Pro-Aleksej:~/Desktop/dia-labs/LR3/lab_python_fp(masterx) ]
 $ python3 unique.py
list1:         [1, 2, 3, 4, 4, 3, 2, 4, 5, 3, 6, 7, 8]
Unique list1:  [1, 2, 3, 4, 5, 6, 7, 8]

list2:         ['a', 'B', 'c', 'd', 'c', 'e', 'a', 'B', 'C', 'A', 'b']
Unique list2:  ['A', 'B', 'C', 'a', 'b', 'c', 'd', 'e']

list2:                         ['a', 'B', 'c', 'd', 'c', 'e', 'a', 'B', 'C', 'A', 'b']
Unique list2(ignore = True):   ['a', 'b', 'c', 'd', 'e']

list3:         ['a', 'B', 1, 'c', 'd', 3, 'c', 'e', 3, 'a', 'B', 2, 'C', 5, 'A', 5, 'b']
Unique list3:  [1, 'e', 3, 2, 5, 'B', 'C', 'b', 'd', 'a', 'A', 'c']

list3:                         ['a', 'B', 1, 'c', 'd', 3, 'c', 'e', 3, 'a', 'B', 2, 'C', 5, 'A', 5, 'b']
Unique list3(ignore = True):   [1, 'e', 3, 2, 5, 'b', 'd', 'a', 'c']
```

- sort.py

```
[  3:27 ]  [ dlnwlkmn@MacBook-Pro-Aleksej:~/Desktop/dia-labs/LR3/lab_python_fp(masterx) ]
 $ python3 sort.py
[123, 100, -100, -30, 4, -4, 1, -1, 0]
[123, 100, -100, -30, 4, -4, 1, -1, 0]
```

- cm_timer.py

```
[  3:48 ]  [ dlnwlkmn@MacBook-Pro-Aleksej:~/Desktop/dia-labs/LR3/lab_python_fp(masterx) ]
 $ python3 cm_timer.py
Затраченное время: 0.005429983139038086
Затраченное время: 1.0045888423919678
Затраченное время: 4.317513942718506
Затраченное время: 1.2051079273223877
```

- print_result.py

```
[  3:49 ]  [ dlnwlkmn@MacBook-Pro-Aleksej:~/Desktop/dia-labs/LR3/lab_python_fp(masterx) ]
 $ python3 print_result.py
!!!!!!!!
test_1
1
test_2
iu5
test_3
a = 1
b = 2
test_4
1
2
```

- process_file.py

```
энтомолог
юрисконсульт
юрисконсульт 2 категории
юрист
f2
программист
программист 1С
f3
программист с опытом Python
программист 1С с опытом Python
f4
программист с опытом Python, зарплата 159688 руб.
программист 1С с опытом Python, зарплата 120208 руб.
Затраченное время: 0.05177497863769531
```

Выше выводится сортированный список названий вакансий.
Перед ним выводится
время чтения и парсинга json