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 ИУ5

«Системы обработки информации и управления»

Отчёт по лабораторной работе №3-4

Выполнила:
Студент группы ИУ5-31Б
Сигал Д.Э.

2022 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 раполагаться 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.

Текст программы:

```
def field(items, *args):
    assert len(args) > 0
    for x in items:
      for y in args:
        if y in x:
          print(x[y])
goods = [
    {'title': 'Ковер', 'price': 2000, 'color': 'green'},
   {'title': 'Диван для отдыха', 'price': 5300, 'color': 'black'}
]

field(goods,'title', 'price','xc','dasd')
```

Экранные формы с примерами выполнения программы:

```
Ковер
2000
Диван для отдыха
5300
```

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.

Текст программы:

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

```
print(gen_random(5, 1, 3))
```

Экранные формы с примерами выполнения программы:

```
3
1
2
3
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.

Текст программы:

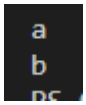
```
class Unique(object):
    def __init__(self, items, **kwargs):
      self.i=[]
      for key, value in kwargs.items():
        if key == 'ignore_case' and value == True:
          items =[j.lower() for j in items]
      for j in items:
        if j not in self.i:
          self.i.append(j)

    def __next__(self):
      try:
        x = self.i[self.begin]
        self.begin += 1
        return x
      except:
        raise StopIteration

    def __iter__(self):
      self.begin = 0
      return self
```

```
data = ['a', 'A', 'b', 'B', 'a', 'A','b', 'B']
for i in Unique(data,ignore_case=True):
  print(i)
```

Экранные формы с примерами выполнения программы:



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.

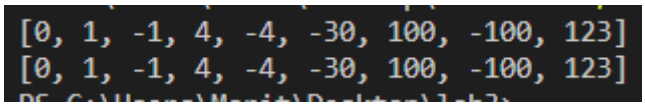
Текст программы:

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

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

    result_with_lambda = sorted(data,key=lambda a:abs(a),reverse=False)
    print(result_with_lambda)
```

Экранные формы с примерами выполнения программы:



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.

Текст программы:

```
def print_result(f):
    def wrapper(a):
```

```
        print(f.__name__)
        res = f(a)
        if type(res) == list:
            for i in res:
                print(i)
        elif type(res) == dict:
            for k,v in res.items():
                print(k, '=', v)
        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__':
    test_1()
    test_2()
    test_3()
    test_4()
```

Экранные формы с примерами выполнения программы:

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

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.

Текст программы:

```
from contextlib import contextmanager
import time
@contextmanager
def cm_timer_1():
  start = time.monotonic()
  yield
  end = time.monotonic()
  print(end-start)
class cm_timer_2(object):
  def __enter__(self):
    cm_timer_2.start = time.monotonic()
  def __exit__(self, exc_type, exc_value, traceback):
    cm_timer_2.end = time.monotonic()
  def __del__(self):
    print(cm_timer_2.end-cm_timer_2.start)

with cm_timer_1():
  time.sleep(1)
with cm_timer_2():
  time.sleep(1)
```

Экранные формы с примерами выполнения программы:

```
1.0
1.0
```

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:

```
import json
from print_result import print_result
from cm_timer import cm_timer_1
from unigue import Unique
from field import field
from gen_random import gen_random
from operator import concat


def f1(a):
    return Unique([i['job-name'] for i in field(data, 'job-name')],
ignore_case=True)
def f2(a):
    return filter(lambda a: a.startswith('программист'), a)
def f3(a):
    return list(map(lambda x: concat(x, ' с опытом Python'), a))
def f4(a):
    return zip(a, gen_random(len(a),137287, 200000))

with open('data_light.json','r',encoding='utf-8') as f:
    data = json.load(f)
    for i in f4(f3(f2(f1(data)))):
        print(i)
```