Московский государственный технический университет им. Н. Э. Баумана

Факультет «Информатика и системы управления»

Кафедра ИУ5 «Система обработки информации и управления»

Курс «Парадигмы и конструкции языков программирования»

Лабораторная работа на тему “ Функциональные возможности языка Python.”

Выполнил:

Студент ИУ5-34Б

Фролов М. К.

Подпись и Дата:

Проверил:

Преподаватель каф. ИУ5

Гапанюк Ю. Е.

Подпись и Дата:

Москва, 2023 г

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.

```python
# 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}
def field(items, *args):
    assert len(args) > 0
    # Необходимо реализовать генератор
    list = []
    if len(args) == 1:
        for i in range(len(items)):
            for j in range(len(args)):
                if args[j] in items[i]:
                    list.append(items[i][args[j]])
    else:
        for i in range(len(items)):
            dictionary = {}
            for j in range(len(args)):
                if items[i].get(args[j]) != None :
                    dictionary[args[j]] = items[i].get(args[j])
            list.append(dictionary)
    return list
```

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`

Шаблон для реализации генератора:

```
# 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
import random
def gen_random(num_count, begin, end):
    # Необходимо реализовать генератор
    list = [int(random.uniform(begin, end)) for i in range(num_count) ]
    return list
```

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.

```
# Итератор для удаления дубликатов
class Unique(object):
    def __init__(self, items, **kwargs):
        # 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
        self.items = items
        self.current_value = 0
        self.ignore_case = kwargs.get('ignore_case', False)
        if not self.ignore_case:
            self.set_arr = []
            for i in self.items:
                if (type(i) is str) and (i.upper() not in self.set_arr and
i.lower() not in self.set_arr):
                    self.set_arr.append(i)
```

```
                elif i not in self.set_arr and (type(i) is not str):
self.set_arr.append(i)
            self.set_arr = list(set(self.items))
    def __next__(self):
        if self.current_value < len(self.set_arr):
            value = self.set_arr[self.current_value]
            self.current_value += 1
            return value
        else:
            raise StopIteration
    def __iter__(self):
        return self
```

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.

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

if __name__ == '__main__':
    print(sorted(data, key = abs, reverse=True))
    print(sorted(data, key = lambda n : abs(n), reverse = True))
```

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.

```
# Здесь должна быть реализация декоратора
def print_result(func):
    def wrapper(*args, **kwargs):
        a = func(*args, **kwargs)
        if isinstance(a, list):
            for i in args: print(args)
        elif isinstance(a, dict):
            for i, j in a.items():
                print(i, " = ", j)
        else:
            print(a)
        return a
    return wrapper
```

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).

```python
import time

class cm_timer_1:
    def __enter__(self):
        self.start_time = time.time()
        return self
    def __exit__(self, exc_type, exc_val, exc_tb):
        elapsed_time = time.time() - self.start_time
        print(f"Elapsed time: {elapsed_time} seconds")

from contextlib import contextmanager
@contextmanager
def cm_timer_2():
    start_time = time.time()
    try:
        yield
    finally:
        elapsed_time = time.time() - start_time
        print(f"Elapsed time: {elapsed_time} seconds")
```

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.

```
import json
from unique import Unique
from print_result import print_result
from field import field
from gen_random import gen_random
from cm_timer import cm_timer_1

path = "C:/Users/fmmf2/Downloads/data_light.json"

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

@print_result
def f1(arg):
    return sorted([x for x in Unique(field(arg, 'title'), ignore_case = True)])

@print_result
def f2(arg):
    return list(filter(lambda x: x.split()[0] + " программист", arg))

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

@print_result
def f4(arg):
```

```
    return  [f"{employee}, зарплата {salary} руб." for employee, salary in
zip(arg, gen_random(len(arg), 100000, 200000))]

if __name__ == "main":
    with cm_timer_1():
        print(f4(f3(f2(f1(data)))))
```

Результат выполнения:

```
эколог
экономист
электрик
электрогазосварщик
электромонтер
электромонтер -линейщик по монтажу воздушных линий высокого напряжения и контак
ти
электромонтер по испытаниям и измерениям 4-6 разряд
электромонтер по ремонту и обслуживанию электрооборудования
электромонтер станционного телевизионного оборудования
электросварщик
электросварщик на автоматических и полуавтоматических машинах
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 183265 руб.
программист 1С с опытом Python, зарплата 143425 руб.
time: 5.989
```