Московский государственный технический университет им. Н.Э. Баумана

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

Курс «Парадигмы и конструкции языков программирования»

Отчет по лабораторной работе №3-4

Выполнил:
студент группы ИУ5-34Б:

Шимко Даниил
Подпись и дата:

Проверил:
преподаватель каф. ИУ5
Гапанюк Ю.
Подпись и дата:

Москва, 2023 г.

Задание:

Задача 1 (файл field.py)

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

Код:

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

Код:

```
import random
def gen_random(num_count, begin, end):
    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.

Код:

```
class Unique(object):
    def __init__(self, items, **kwargs):
        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.

Код:

```
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.

Код:

```
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.

Код:

```
import time
from contextlib import ContextDecorator

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")

class cm_timer_2(ContextDecorator):
    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")
```