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

Факультет «Информатика и управление»

Кафедра «Системы обработки информации и управления»

Курс «БКИТ»
Отчет по лабораторной работе №3,4

Выполнил:
студент группы ИУ5-35Б
Крылов Дмитрий
Подпись и дата:

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

Москва, 2022 г.

Описание задания

1) Необходимо реализовать генератор field. Генератор field последовательно выдает значения ключей словаря
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
3) Необходимо реализовать итератор Unique(данные), который принимает на вход массив или генератор и итерируется по элементам, пропуская дубликаты
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
5) Необходимо реализовать декоратор print_result, который выводит на экран результат выполнения функции
6) Необходимо написать контекстные менеджеры cm_timer_1 и cm_timer_2, которые считают время работы блока кода и выводят его на экран
7) В предыдущих задачах были написаны все требуемые инструменты для работы с данными. Применить их на реальном примере

Текст программы

Task1.py

```
def field(items, *args):
    assert len(args) > 0
    for a in range(len(args)):
        for i in range(len(items)):
            try: print(items[i][args[a]])
            except: print()



goods = [
    {'title': 'Ковер', 'price': 2000, 'color': 'green'},
    {'title': 'Диван для отдыха', 'color': 'black'}
]


field(goods, "title", "price")
```

Task2.py

```
import random

def gen_random(num_count, begin, end):
      a=[]
      for i in range(num_count): a.append(random.randint(begin,end))
      print(a)

gen_random(5,1,10)
```

Task3.py

```python
# Итератор для удаления дубликатов
class Unique(object):
    def __init__(self, items, **kwargs):
        self.items = items
        for k, v in kwargs.items():
            setattr(self, k, v)
        if not hasattr(self, 'ignore_case'):
            self.ignore_case = False

    def __next__(self):
        if self.ignore_case:
            moment_array = [x.lower() for x in self.items]
        else:
            moment_array = self.items
        self.items = []
        for element in moment_array:
            if element not in self.items:
                    self.items.append(element)
        print(self.items)
        return None

    def __iter__(self):
        return self

data = ["a", "A", "b", "B", "a", "A", "b", "B"]
class1 = Unique(data, ignore_case=True)
next(class1)
```

Task4.py

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

if __name__ == '__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)
```

Task5.py

```python
# Здесь должна быть реализация декоратора
def print_result(func):
    def function():
        print(func.__name__)
        func_result = func()

        if isinstance(func_result, int) or isinstance(func_result, str):
            print(func_result)
        elif isinstance(func_result, dict):
            all_keys = list(func_result.keys())
            all_values = list(func_result.values())
            for elem in range(len(all_keys)):
                print(all_keys[elem], '=', all_values[elem])
        elif isinstance(func_result, list):
            for elem in func_result:
                print(elem)

    return function



@print_result
def test_1():
    return 1


@print_result
def test_2():
    return 'iu5'
```

```python
@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()
```

Task6.py

```python
import time
from contextlib import contextmanager

class cm_timer_1:
    def __init__(self):
        self.start_time = time.time()
        pass

    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(round(time.time() - self.start_time, 1))

@contextmanager
def cm_timer_2():
    start_time = time.time()
    yield 0
    print(round(time.time() - start_time, 1))

with cm_timer_1():
    time.sleep(5.5)

with cm_timer_2():
    time.sleep(5.5)
```

Task7.py

```python
import json, time, sys, functools
import random
from contextlib import contextmanager

path = 'data_light.json'

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

class Unique(object):
    def __init__(self, items, **kwargs):
        self.items = items
        for k, v in kwargs.items():
            setattr(self, k, v)
        if not hasattr(self, 'ignore_case'):
            self.ignore_case = False

    def __next__(self):
        if self.ignore_case:
            moment_array = [x.lower() for x in self.items]
        else:
            moment_array = self.items
        self.items = []
        for element in moment_array:
            if element not in self.items:
                self.items.append(element)
        return self.items
```

```
    def __iter__(self):
        return self

def field(items, *args):
    res = []
    assert len(args) > 0
    for a in range(len(args)):
        for i in range(len(items)):
            try:
                res.append(items[i][args[a]])
            except:
                pass
    return res


def print_result(func):
    @functools.wraps(func)
    def wrapper(*func_args, **func_kwargs):
        print('Результат выполнения функции', func.__name__, ':')
        func_result = func(*func_args, **func_kwargs)
        if isinstance(func_result, int) or isinstance(func_result, str):
            print(func_result)
        elif isinstance(func_result, dict):
            all_keys = list(func_result.keys())
            all_values = list(func_result.values())
            for elem in range(len(all_keys)):
                print(all_keys[elem], '=', all_values[elem])
        elif isinstance(func_result, list):
            for elem in func_result:
```

```
            print(elem)
        return func_result

    return wrapper

def filter_func(elem):
    if elem.lower().startswith('программист'):
        return True
    else:
        return False

def add_end_stage(elem):
    elem += ' с опытом Python'
    return elem

@contextmanager
def cm_timer_1():
    start_time = time.time()
    yield 0
    print('===\nВремя выполнения пограммы:', round(time.time() - start_time, 10), 'секунд')

@print_result
def f1(arg):
    return next(Unique(field(arg, "job-name"), ignore_case=True))

@print_result
def f2(arg):
    return list(filter(filter_func, arg))
```

```
@print_result
def f3(arg):
    return list(map(add_end_stage, arg))

@print_result
def f4(arg):
    temp = []
    for i in range(len(arg)):
        random_zp = random.randint(100000, 200000)
        temp.append(f', зарплата {random_zp} руб.')
    return list(zip(arg, temp))

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

Результаты выполнения программы

Task1.py

```
/Users/dmitrykrylov/PycharmProjects/newlaba3/venv/bin/python /Users/dmitrykrylov/PycharmProjects/newlaba3/1.py
Ковер
Диван для отдыха
2000


Process finished with exit code 0
```

Task2.py

```
/Users/dmitrykrylov/PycharmProjects/newlaba3/venv/bin/python /Users/dmitrykrylov/PycharmProjects/newlaba3/2.py
[1, 3, 9, 3, 5]

Process finished with exit code 0
```

Task3.py

```
/Users/dmitrykrylov/PycharmProjects/newlaba3/venv/bin/python /Users/dmitrykrylov/PycharmProjects/newlaba3/3.py
['a', 'b']

Process finished with exit code 0
```

Task4.py

```
/Users/dmitrykrylov/PycharmProjects/newlaba3/venv/bin/python /Users/dmitrykrylov/PycharmProjects/newlaba3/4.py
[123, 100, -100, -30, 4, -4, 1, -1, 0]
[123, 100, -100, -30, 4, -4, 1, -1, 0]

Process finished with exit code 0
```

Task5.py

```
/Users/dmitrykrylov/PycharmProjects/newlaba3/venv/bin/python /Users/dmitrykrylov/PycharmProjects/newlaba3/5.py
test_1
1
test_2
iu5
test_3
a = 1
b = 2
test_4
1
2

Process finished with exit code 0
```

Task6.py

```
/Users/dmitrykrylov/PycharmProjects/newlaba3/venv/bin/python /Users/dmitrykrylov/PycharmProjects/newlaba3/6.py
5.5
5.5

Process finished with exit code 0
```

Task7.py

```
/Users/dmitrykrylov/PycharmProjects/newlaba3/venv/bin/python /Users/dmitrykrylov/PycharmProjects/newlaba3/7.py
Результат выполнения функции f1 :
администратор на телефоне
медицинская сестра
охранник сутки-день-ночь-вахта
врач анестезиолог реаниматолог
теплотехник
разнорабочий
электро-газосварщик
водитель gett/гетт и yandex/яндекс такси на личном автомобиле
монолитные работы
организатор - тренер
помощник руководителя
автоэлектрик
врач ультразвуковой диагностики в детскую поликлинику
```

...

```
программист c# с опытом Python
Результат выполнения функции f4 :
('программист с опытом Python', ', зарплата 196690 руб.')
('программист с++/c#/java с опытом Python', ', зарплата 117715 руб.')
('программист 1с с опытом Python', ', зарплата 100969 руб.')
('программистр-разработчик  информационных систем с опытом Python', ', зарплата 105147 руб.')
('программист с++ с опытом Python', ', зарплата 193569 руб.')
('программист/ junior developer с опытом Python', ', зарплата 147181 руб.')
('программист / senior developer с опытом Python', ', зарплата 109095 руб.')
('программист/ технический специалист с опытом Python', ', зарплата 149419 руб.')
('программист c# с опытом Python', ', зарплата 133297 руб.')
===
Время выполнения программы: 0.029602766 секунд

Process finished with exit code 0
```