МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
ИМ. Н. Э. БАУМАНА

ФАКУЛЬТЕТ
«ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ»

КАФЕДРА
«АВТОМАТИЗИРОВАННЫЕ СИСТЕМЫ ОБРАБОТКИ ИНФОРМАЦИИ И УПРАВЛЕНИЯ»

Лабораторная работа №3

по учебной дисциплине
«Объектно-ориентированные возможности языка Python»

на тему
«Основы языка Python»

Вариант №1

Группа: *ИУ5Ц-71Б*

Студент: *Бабакулыев Ф.А.*

Преподаватель: *Гапанюк Ю.Е.*

Москва, 2021

1. Описание задания

Задача 1 (файл field.py)

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

- 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.

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.

Необходимо решить задачу двумя способами:

- С использованием lambda-функции.
- Без использования lambda-функции.

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.

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).

Задача 7 (файл process_data.py)

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 модификациииспользуйте 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.

2. Текст программы

Файл «field.py»

```
goods = [
    {'title': 'Ковер',              'price': 2000,  'color': 'green'},
    {'title': 'Диван для отдыха',   'price': 5800,  'color': 'black'},
    {'title': 'Софа',               'price': 3000,  'color': 'blue'},
    {'title': 'Телевизор',          'price': 4000,  'color': 'black'},
    {'title': 'Атомная бомба',      'price': 6000,  'color': 'new
sun'},
    {'title': 'Зачёт (бесценно)',                  'color':
'rainbow'},
]

def field(items, *args):
    assert len(args) > 0, 'Список аргументов не должен быть пустым'
    for i in items:
        arr = {}
        for j in args:
            if j in i:
                arr.update({j: i[j]})
        if len(args) == 1:
            if args[0] in arr:
                print(arr[args[0]])
        else:
            print(arr)

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

Файл «gen_random.py»

```
import random

def gen_random(num_count, begin, end):
    i = 0
    arr = []
    while i < num_count:
        arr.append(random.randint(begin, end))
        i+=1
    return arr

if __name__ == '__main__':
    print("Рандомные числа:", gen_random(5,0,10))
```

Файл «print_result.py»

```
def print_result(result):
    def wrapper():
        print(result.__name__)
        res = result()
        if isinstance(res,list):
            for i in res:
                print(i)
        elif isinstance(res,dict):
            for i in res:
                print(i, ' = ', res[i])
        else: print(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()
```

Файл «process_data.py»

```
import json
import sys
from Tasks.cm_timer import cm_timer_1
from Tasks.gen_random import gen_random
from Tasks.unique import Unique

def print_result(result):
    def wrapper(data):
        res = result(data)
        print(res)
        return res
    return wrapper

path = 'Tasks/datalight.json'

with open(path, encoding="utf8") as f:
    data = json.load(f)
```

```
# @print_result
def f1(arg):
    arr = []
    for i in arg:
        arr.append(i['job-name'])
    res = Unique(arr, ignore_case=True)
    return res.data


# @print_result
def f2(arg):
    return list(filter(lambda x: x.lower().find('программист') != -1, arg))


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


@print_result
def f4(arg):
    return list(map(lambda x: x + ', зарплата {} руб.'.format(gen_random(1,
100000, 250000)[0]), arg))


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

Файл «sort.py»

```
import math

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

print( sorted(data, key=lambda x: math.fabs(x), reverse=True) )

def funcSort(x):
    return math.fabs(x)

print( sorted(data, key=funcSort, reverse=True) )
```

Файл «switch.py»

```
# Класс для обеспечения функционала Switch-Case
class switch(object):
    def __init__(self, value):
```

```
        self.value = value  # значение, которое будем искать
        self.fall = False   # для пустых case блоков

    def __iter__(self):     # для использования в цикле for
        """ Печатает о выполнении и возвращается """
        #print('Пункт выбран: ', self.value)
        yield self.match    # Выполняет метод math и возвращается
        #raise StopIteration    # Почему-то вызывает исключение
(Почему?)
        return

    def match(self, *args):
        """ Указывает, нужно ли заходить в тестовый вариант """
       # Проверка: Было ли уже "попадание"?
        if self.fall:
            # fall означает, что ранее сработало условие

            # Если возвращает False - исполняет только выбранный Case
            # Если возвращает True - исполняет каждый case до первого
break
            return False
        # Проверка: Закончились ли "варианты Case"?
        # (Если не было ни одного попадания)
        elif not args:
            # пустой список аргументов означает последний блок case

            # Если возвращает False - НЕ исполняет case по-умолчанию
            # Если возвращает True - исполняет case по-умолчанию
            return True
        # Проверка: "нужный case" найден?
        # (Если не было ни одного попадания и варианты ещё не
закончились)
        elif self.value in args:
            self.fall = True    # Отметка: Найдено совпадение
                                # Нужно для *фиксирования попадания* и
*не захода в вариант по-умолчанию*

            # Если стоит True - выполняет выбранный Case
            # Если стоит False - НЕ выполяет выбранный Case
            return True
        return False
# ***
```

Файл «Menu.py»

```
from Menu.switch import switch
from subprocess import call
from shutil import get_terminal_size

# Класс для обеспечения функционала Switch-Case
class menu(object):

    def __init__(self, options):
        self.options = options

    def input(self):
        while True:
            try:
                print()
                self.value = int(input('| Введите пункт меню: '))
                print()
                print()
                print("|***|***|***|")
                print()
            except BaseException:
                print('Вы не выбрали пункт меню!')
                print('Повторите ввод!')
                print()
                continue
            return switch(self.value)

    def print(self, options=None, indent=0):
        if not isinstance(options, tuple):
            options = self.options
        #deep - глубина списка. Нужна для регулирования вертикальных
отступов
        deep = indent
        for item in options:
            if type(item) == tuple:
                deep = self.print(item, indent+1)
                if (deep >= indent):
                    deep=0
                    print()
            else:
                print("|"+"-"*indent+"| "+item)
        return deep

    def end(cont):
        print()
        print("|***|***|***|")
        print()
        pause()
        clear()
# ***
```

```
# Функция для обеспечения вызова другого файла
# Запускаем другой файл (в том же терминале)
def run(runfile):
    call(["python", runfile])


# Функция обеспечения паузы
def pause():
    programPause = input("Press the <ENTER> key to continue...")

def clear():
    print("\n" * get_terminal_size().lines, end='')
```

Файл «Test.py, Test1.py»

```
print("Test")


A = "Test1"
print(A)
```

Файл «cm_timer.py»

```
import time
from contextlib import contextmanager

class cm_timer_1():
    def __enter__(self):
        self.start = time.time()
    def __exit__(self, type, value, traceback):
        print(time.time() - self.start)

@contextmanager
def cm_timer_2():
    start = time.time()
    yield
    print(time.time() - start)

if __name__ == '__main__':
    with cm_timer_1():
        time.sleep(3)

    with cm_timer_2():
        time.sleep(2)
```

Файл «LR3.py»

```
from Menu.Menu import menu, run, pause

# Задаём пункты меню
MenuMain = menu((
    "1. Задача 1",
    (
        "[файл field.py]",
        (
            "Необходимо реализовать генератор field. Генератор field
последовательно выдает значения ключей словаря.",
        ),
    ),
    "2. 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.",
        ),
    ),
    "3. 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.",
        ),
    ),
    "4. 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.",
```

```
            "Необходимо решить задачу двумя способами:",
            (
                "С использованием lambda-функции.",
                "Без использования lambda-функции.",
            )
        ),
    ),
    "5. 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.",
            )
        ),
    ),
    "6. 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.",
            "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.",
            ),
        ),
    ),
    "7. Задача 7",
    (
        "[файл process_data.py]",
        (
            "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). 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. Используйте zip для обработки пары специальность —
зарплата.",
        ),
    ),
    "0. Exit"
))

# Обеспечиваем "вечную работу", пока пользователь не захочет выйти
cont = True
while cont:

    # Печать пунктов меню
    MenuMain.print()

    # Осуществляем выполнение пунктов меню
    for case in MenuMain.input():
        if case(1): run("Tasks/field.py")
        if case(2): run("Tasks/gen_random.py")
        if case(3): run("Tasks/unique.py")
        if case(4): run("Tasks/sort.py")
        if case(5): run("Tasks/print_result.py")
        if case(6): run("Tasks/cm_timer.py")
        if case(7): run("Tasks/process_data.py")

        if case(0): cont=0
        # default
        if case():
            print('Вы хотите выйти?')
            print('Нажмите "0" !')

    # Окончание действия меню
    MenuMain.end()
```

3. Вывод программы

Задача 1.

```
| Введите пункт меню: 1


|***|***|***|

{'title': 'Ковер', 'price': 2000}
{'title': 'Диван для отдыха', 'price': 5800}
{'title': 'Софа', 'price': 3000}
{'title': 'Телевизор', 'price': 4000}
{'title': 'Атомная бомба', 'price': 6000}
{'title': 'Зачёт (бесценно)'}

|***|***|***|

Press the <ENTER> key to continue...
```

Задача 2.

```
| Введите пункт меню: 2


|***|***|***|

Рандомные числа: [4, 2, 4, 6, 8]

|***|***|***|

Press the <ENTER> key to continue...
```

Задача 3.

```
| Введите пункт меню: 3


|***|***|***|

B
A
a
b

|***|***|***|

Press the <ENTER> key to continue...
```

Задача 4.

```
| Введите пункт меню: 4


|***|***|***|

[123, 100, -100, -30, 30, 4, -4, 1, -1, 0]
[123, 100, -100, -30, 30, 4, -4, 1, -1, 0]

|***|***|***|

Press the <ENTER> key to continue...
```

Задача 5.

```
| Введите пункт меню: 5


|***|***|***|

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

|***|***|***|

Press the <ENTER> key to continue...
```

Задача 6.

```
| Введите пункт меню: 6


|***|***|***|

3.001065969467163
2.0001800060272217

|***|***|***|

Press the <ENTER> key to continue...
```

Задача 7.

```
| Введите пункт меню: 7


|***|***|***|

['помощник веб-программиста с опытом Python, зарплата 155772 руб.', 'программист / senior developer с опытом Python, зарплата 2
18935 руб.', 'старший программист с опытом Python, зарплата 166302 руб.', 'web-программист с опытом Python, зарплата 196599 руб
.', 'программист с опытом Python, зарплата 146940 руб.', 'системный программист (c, linux) с опытом Python, зарплата 140394 руб
.', 'инженер-программист 1 категории с опытом Python, зарплата 181407 руб.', 'программист/ технический специалист с опытом Pyth
on, зарплата 173260 руб.', 'инженер-программист сапоу (java) с опытом Python, зарплата 236909 руб.', 'инженер-программист (орех
ово-зуевский филиал) с опытом Python, зарплата 246884 руб.', 'педагог программист с опытом Python, зарплата 212589 руб.', 'прог
раммист c# с опытом Python, зарплата 217253 руб.', 'веб-программист с опытом Python, зарплата 122233 руб.', 'программистр-разра
ботчик  информационных систем с опытом Python, зарплата 115266 руб.', 'инженер-программист (клинский филиал) с опытом Python, з
арплата 103362 руб.', 'инженер-программист ккт с опытом Python, зарплата 123623 руб.', 'программист/ junior developer с опытом
Python, зарплата 230654 руб.', 'инженер-программист плис с опытом Python, зарплата 215583 руб.', 'инженер-программист с опытом
Python, зарплата 166085 руб.', 'инженер-электронщик (программист асу тп) с опытом Python, зарплата 237633 руб.', 'инженер - про
граммист с опытом Python, зарплата 201922 руб.', 'инженер - программист асу тп с опытом Python, зарплата 149782 руб.', 'веб - п
рограммист (php, js) / web разработчик с опытом Python, зарплата 201593 руб.', 'ведущий программист с опытом Python, зарплата 2
45328 руб.', 'ведущий инженер-программист с опытом Python, зарплата 186453 руб.', '1с программист с опытом Python, зарплата 206
906 руб.', 'программист c++/c#/java с опытом Python, зарплата 166939 руб.', 'программист c++ с опытом Python, зарплата 213943 р
уб.', 'программист 1с с опытом Python, зарплата 153241 руб.']
0.016390562057495117

|***|***|***|

Press the <ENTER> key to continue...
```