**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 №2-3

«Функциональные возможности языка Python»

|  |  |  |
| --- | --- | --- |
| Выполнила: |  | Проверил: |
| студент группы ИУ5-33Б |  | преподаватель каф. ИУ5 |
| Хабленко Инна |  | Гапанюк Ю. Е. |
|  |  |  |

Москва, 2023 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.

1. 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.

1. 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
2. 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.

1. 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:

С использованием lambda-функции.

Без использования lambda-функции.

1. 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.

1. 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.

1. 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

|  |  |
| --- | --- |
| 1 | def field(items, \*args):      assert len(args) > 0      for i in items:          if all(arg in i for arg in args):              for arg in args:                  print(i[arg], end=" ")      print("")  if \_\_name\_\_ == '\_\_main\_\_':      goods = [      {'title': 'Ковер', 'price': 2000, 'color': 'green'},      {'title': 'Диван для отдыха', 'price': 5300, 'color': 'black'}      ]      field(goods, 'title')      field(goods, 'title', 'price') |
| 2 | import random  def gen\_random(num\_count, begin, end):      numbers = []      for i in range(num\_count):          numbers.append(random.randint(begin, end))      return numbers  if \_\_name\_\_ == '\_\_main\_\_':      numbers = gen\_random(5, 1, 3)      print(numbers) |
| 3 | import random  def gen\_random(num\_count, begin, end):      numbers = []      for i in range(num\_count):          numbers.append(random.randint(begin, end))      return numbers  # Итератор для удаления дубликатов  class Unique(object):      def \_\_init\_\_(self, items, \*\*kwargs):          self.items = items          self.index = 0          self.ignore\_case = kwargs.get('ignore\_case', False)          self.seen = set()      def \_\_next\_\_(self):          while self.index < len(self.items):              item = self.items[self.index]              self.index += 1              item\_key = item.lower() if self.ignore\_case else item              if item\_key not in self.seen:                  self.seen.add(item\_key)                  return item          raise StopIteration()      def \_\_iter\_\_(self):          return self    if \_\_name\_\_ == '\_\_main\_\_':      data = [1, 1, 1, 1, 1, 2, 2, 2, 2, 2]      unique\_data = Unique(data)      print(list(unique\_data))      data = gen\_random(10, 1, 3)      unique\_data = Unique(data)      print(list(unique\_data))      data = ["a", "A", "b", "B", "a", "A", "b", "B"]      unique\_data = Unique(data)      print(list(unique\_data))      data = ["a", "A", "b", "B", "a", "A", "b", "B"]      unique\_data = Unique(data, ignore\_case=True)      print(list(unique\_data)) |
| 4 | import math  data = [4, -30, 100, -100, 123, 1, 0, -1, -4]  result = sorted(list(x for x in data), key=abs, reverse=True)  print(result)  result\_with\_lambda = sorted(data, key = lambda x: abs(x), reverse=True)  print(result\_with\_lambda) |
| 5 | def print\_result(func):      def wrapper(\*args, \*\*kwargs):          res = func(\*args, \*\*kwargs)          print(f"{func.\_\_name\_\_}")          if isinstance(res, list):              for r in res:                  print (r)          elif isinstance(res, dict):              for x, y in res.items():                  print (x, " = ", y)          else:              print(res)          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\_\_':      print('!!!!!!!!')      test\_1()      test\_2()      test\_3()      test\_4() |
| 6 | import time  from contextlib import contextmanager  class cm\_timer\_1:      def \_\_enter\_\_(self):          self.start\_time = time.time()        def \_\_exit\_\_(self, exc\_type, exc\_val, exc\_tb):          elapsed\_time = time.time() - self.start\_time          print(f"time\_1: {elapsed\_time}")  @contextmanager  def cm\_timer\_2():      start\_time = time.time()      yield      elapsed\_time = time.time() - start\_time      print(f"time\_2: {elapsed\_time}")  if \_\_name\_\_ == '\_\_main\_\_':      with cm\_timer\_1():          time.sleep(5.5)      with cm\_timer\_2():          time.sleep(5.5) |
| 7 | import json  from gen\_random import gen\_random  from cm\_timer import cm\_timer\_1  from print\_result import print\_result  path = "C:/Users/khabl/OneDrive/Рабочий стол/Py/lab\_2/data\_light.json"  with open(path, encoding='utf-8') as f:      data = json.load(f)  @print\_result  def f1(arg):      return sorted(set(item['job-name'].lower() for item in arg))  @print\_result  def f2(arg):      return list(filter(lambda s: s.startswith('программист'), arg))  @print\_result  def f3(arg):      return list(map(lambda s: s + ' с опытом Python', arg))  @print\_result  def f4(arg):      salaries = gen\_random(len(arg), 100000, 2000000)      return ['{} зарплата {}'.format(job, salary) for job, salary in zip(arg, salaries)]  if \_\_name\_\_ == '\_\_main\_\_':      with cm\_timer\_1():          f4(f3(f2(f1(data)))) |

# Результат

|  |  |
| --- | --- |
| 1 |  |
| 2 |  |
| 3 |  |
| 4 |  |
| 5 |  |
| 6 |  |
| 7 |  |