Группа: ИУ5-31Б

Студент: Болотин Ю.А.

**Текст программы**

**a\_field.py**

def field(items, \*args):  
 assert len(args) > 0  
 if len(args) == 1:  
 for item in items:  
 if args[0] in item and item[args[0]] is not None:  
 yield item[args[0]]  
 else:  
 for item in items:  
 result = {}  
 all\_none = True  
 for key in args:  
 if key in item and item[key] is not None:  
 result[key] = item[key]  
 all\_none = False  
 if not all\_none:  
 yield result  
  
  
goods = [  
 {'title': 'Ковер', 'price': 2000, 'color': 'green'},  
 {'title': 'Диван для отдыха', 'price': 5300, 'color': 'black'}  
]  
  
print(str(list(field(goods, 'title')))[1:-1]) # должен выдавать 'Ковер', 'Диван для отдыха'  
print(str(list(field(goods, 'title', 'price')))[1:-1]) # должен выдавать {'title': 'Ковер', 'price': 2000}, {'title': 'Диван для отдыха', 'price': 5300}

**b\_gen\_random.py**  
  
from random import randint  
def gen\_random(num\_count, begin, end):  
 ans = []  
 for i in range(num\_count):  
 ans.append(randint(begin, end))  
 yield ans  
  
# print(str(list(gen\_random(5, 1, 3)))[1:-1])

**c\_unique.py**

from b\_gen\_random import gen\_random  
  
class Unique(object):  
 def \_\_init\_\_(self, items, \*\*kwargs):  
 self.data = iter(items)  
 self.ignore\_case = kwargs.get('ignore\_case', False)  
 self.unique\_items = set()  
  
 def \_\_next\_\_(self):  
 while True:  
 item = next(self.data)  
 check\_item = item.lower() if self.ignore\_case else item  
  
 if type(check\_item) != list and check\_item not in self.unique\_items:  
 self.unique\_items.add(check\_item)  
 return item  
  
 def \_\_iter\_\_(self):  
 return self  
  
data = [1, 1, 1, 1, 1, 2, 2, 2, 2, 2]  
unique\_data = Unique(data)  
print(list(unique\_data))  
# Output: [1, 2]  
  
data = gen\_random(10, 1, 3)  
data = list(data)[0]  
  
unique\_data = Unique(data)  
print(list(unique\_data))  
# Output: [1, 2, 3]  
  
data = ['a', 'A', 'b', 'B', 'a', 'A', 'b', 'B']  
unique\_data = Unique(data)  
print(list(unique\_data))  
# Output: ['a', 'A', 'b', 'B']  
  
unique\_data\_ignore\_case = Unique(data, ignore\_case=True)  
print(list(unique\_data\_ignore\_case))  
# Output: ['a', 'b']

**d\_sort.py**

data = [4, -30, 100, -100, 123, 1, 0, -1, -4]  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 result = sorted(data, key=lambda x: abs(x), reverse=True)  
 print(result)  
 result\_with\_lambda = sorted(data, key=abs, reverse=True)  
 print(result\_with\_lambda)

**e\_print\_result.py**

def print\_result(func):  
 def wrapper(\*args, \*\*kwargs):  
 result = func(\*args, \*\*kwargs)  
  
 if isinstance(result, list):  
 for item in result:  
 print(item)  
  
 elif isinstance(result, dict):  
 for key, value in result.items():  
 print(f'{key} = {value}')  
  
 else:  
 print(result)  
  
 return result  
  
 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()

**f\_cm\_timer.py**

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: {elapsed\_time}")  
  
  
@contextmanager  
def cm\_timer\_2():  
 start\_time = time.time()  
 yield  
 elapsed\_time = time.time() - start\_time  
 print(f"time: {elapsed\_time}")  
  
  
# Использование cm\_timer\_1  
with cm\_timer\_1():  
 time.sleep(5.5)  
  
# Использование cm\_timer\_2  
with cm\_timer\_2():  
 time.sleep(5.5)

**g\_process\_data.py**

import json  
from b\_gen\_random import gen\_random  
from e\_print\_result import print\_result  
from f\_cm\_timer import cm\_timer\_1  
  
  
path = "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) # Generate salaries for each employee individually  
 return ['{} зарплата {}'.format(job, salary) for job, salary in zip(arg, salaries)]  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 with cm\_timer\_1():  
 f4(f3(f2(f1(data))))

**data\_light.json**

Вывод:

Изображение выглядит как текст, Шрифт, программное обеспечение, снимок экрана

Автоматически созданное описание