**Функциональные возможности языка Python.**

Выполнил: Абдуллаев Гасан  
Группа: ИУ5-36Б  
Дата: 17.11.24г.

Описание задания:  
[lab\_python\_fp · ugapanyuk/BKIT\_2022 Wiki · GitHub](https://github.com/ugapanyuk/BKIT_2022/wiki/lab_python_fp)

Код программы:

1 Задание:

def field(items, \*args):

assert len(args) > 0, "Должен быть хотя бы один ключ"

if len(args) == 1:

key = args[0]

for item in items:

value = item.get(key)

if value is not None:

yield value

else:

for item in items:

result = {key: item.get(key) for key in args if item.get(key) is not None}

if result:

yield result

if \_\_name\_\_ == "\_\_main\_\_":

goods = [

{'title': 'Ковер', 'price': 2000, 'color': 'green'},

{'title': 'Диван для отдыха', 'color': 'black'}

]

print(list(field(goods, 'title')))

print(list(field(goods, 'color')))

print(list(field(goods, 'title', 'price')))

2 Задание:

import random

def gen\_random(count, min, max):

empty = None

if min > max:

empty = max

max = min

min = empty

for \_ in range (count):

yield random.randint(min, max)

if \_\_name\_\_ == "\_\_main\_\_":

print(list(gen\_random(4, 100, 1)))

3 Задание:

class Unique:

def \_\_init\_\_(self, items, \*\*kwargs):

self.ignore\_case = kwargs.get('ignore\_case', False)

self.seen = set()

self.items = iter(items)

def \_\_iter\_\_(self):

return self

def \_\_next\_\_(self):

while True:

item = next(self.items)

comparison\_item = item.lower() if self.ignore\_case and isinstance(item, str) else item

if comparison\_item not in self.seen:

self.seen.add(comparison\_item)

return item

if \_\_name\_\_ == "\_\_main\_\_":

data = [1, 1, 1, 2, 2, 2, 3, 3]

for i in Unique(data):

print(i)

data = ['a', 'A', 'b', 'B', 'a', 'A', 'b', 'B']

for i in Unique(data):

print(i)

for i in Unique(data, ignore\_case=True):

print(i)

4 Задание:

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 y: abs(y), reverse = False)

print(result\_with\_lambda)

5 Задание:

def print\_result(func):

def printer(\*args, \*\*kwargs):

result = func(\*args, \*\*kwargs)

print(func.\_\_name\_\_)

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 printer

@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 Задание:

import time

from contextlib import contextmanager

class cm\_timer\_1:

def \_\_enter\_\_(self):

self.start\_time = time.time()

return self

def \_\_exit\_\_(self, exc\_type, exc\_val, exc\_tb):

end\_time = time.time()

print(f"time: {end\_time - self.start\_time:.2f}")

@contextmanager

def cm\_timer\_2():

start\_time = time.time()

try:

yield

finally:

end\_time = time.time()

print(f"time: {end\_time - start\_time:.2f}")

if \_\_name\_\_ == "\_\_main\_\_":

from time import sleep

print("Using cm\_timer\_1:")

with cm\_timer\_1():

sleep(2)

print("Using cm\_timer\_2:")

with cm\_timer\_2():

sleep(2)

7 Задание:

import json

import sys

import random

from contextlib import contextmanager

import time

def print\_result(func):

def wrapper(\*args, \*\*kwargs):

result = func(\*args, \*\*kwargs)

print(func.\_\_name\_\_)

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

class cm\_timer\_1:

def \_\_enter\_\_(self):

self.start\_time = time.time()

return self

def \_\_exit\_\_(self, exc\_type, exc\_val, exc\_tb):

end\_time = time.time()

print(f"time: {end\_time - self.start\_time:.2f} seconds")

path = sys.argv[1] if len(sys.argv) > 1 else "data\_light.json"

with open(path, encoding="utf-8") as f:

data = json.load(f)

@print\_result

def f1(arg):

return sorted(set(job['job-name'].strip().lower().capitalize() for job in arg))

@print\_result

def f2(arg):

return list(filter(lambda x: x.lower().startswith('программист'), arg))

@print\_result

def f3(arg):

return list(map(lambda x: f"{x} с опытом Python", arg))

@print\_result

def f4(arg):

salaries = [random.randint(100000, 200000) for \_ in arg]

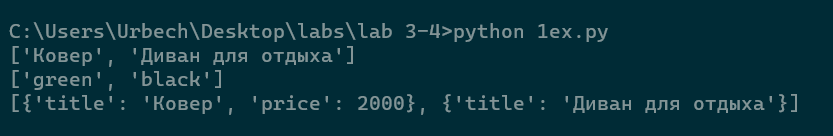
return [f"{job}, зарплата {salary} руб." for job, salary in zip(arg, salaries)]

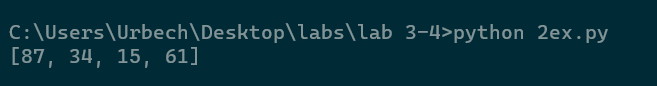
if \_\_name\_\_ == '\_\_main\_\_':

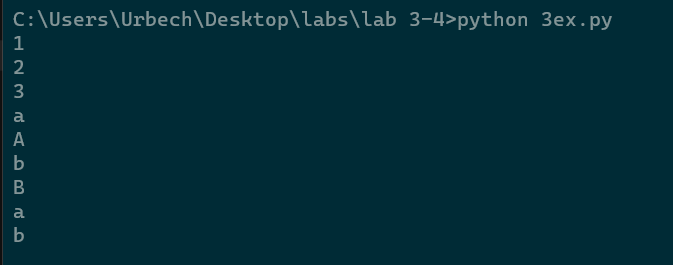
with cm\_timer\_1():

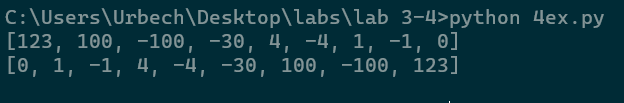
f4(f3(f2(f1(data))))

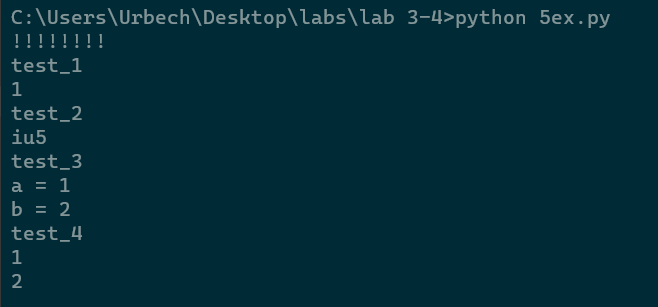
Снимки экрана:  
1 Задание:

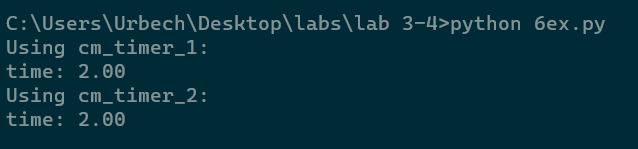


2 Задание:  


3 Задание:  


4 Задание:  


5 Задание:  


6 Задание:  


7 Задание:  
