Тест по лекции 1

1 2 3 4

Вопрос 1

Как проверить путь на факт существования там директории или файла

Ваш ответ:
- (•) os.path.exists
- ( ) os.path.isdir
- ( ) os.path.isfile

Ответить

1 2 3 4

Вопрос 2

Как найти относительный путь от исходного пути?

Ваш ответ:
- ( ) С помощью функции os.path.abspath
- (•) С помощью функции os.path.relpath
- ( ) С помощью функции os.getcwd

Ответить

1 2 3 4

Вопрос 3

Как открыть файл в режиме чтения (path - переменная, в которой находится путь, в котором находится наш файл)?

Ваш ответ:
- (•) open(path, "r")
- ( ) open(path, "w")

Ответить

1 2 3 4

Вопрос 4

Как добавить к исходному пути (пускай этот путь будет находиться в переменной root_path) дочернюю директорию, которая будет называться child_path?

Ваш ответ:
- (•) os.path.join(root_path, child_path)
- ( ) os.path.split(root_path)

Ответить

Тест по лекции 2

1 2 3

Вопрос 1

Какая структура данных не является словарем?

Ваш ответ:
- ( ) ChainMap
- ( ) Counter
- ( ) defaultdict
- (•) deque

Ответить

1 2 3

Вопрос 2

Какие из функций не являются декоратором?

Ваш ответ:
- (•) partial
- ( ) lru_cache
- ( ) total_ordering

Ответить

1 2 3

Вопрос 3

Чем отличается функция zip_longest от zip?

Ваш ответ:

- (•) zip проходится до самого короткого итерируемого объекта, zip_longest - до самого длинного
- ( ) Ничем - обе функции проходятся до самого длинного объекта

Ответить

Тест по лекции 3

1 2 3

Вопрос 1

Как спарсить строку “2023-31-12T23:59:59“ в объект datetime?

Ваш ответ:

- ( ) strptime(“2023-31-12T23:59:59“, “%Y-%M-%D %H:%M%S“)
- ( ) strptime(“2023-31-12T23:59:59“, “%Y-%D-%MT%H:%M%S“)
- (•) strptime(“2023-31-12T23:59:59“, “%Y-%d-%mT%H:%M%S“)

Ответить

1 2 3

Вопрос 2

С помощью какой функции можно отобразить текущее время?

Ваш ответ:

- (•) datetime.datetime.now().time()
- ( ) datetime.now().time()
- ( ) datetime.time.now()

Ответить

1 2 3

Вопрос 3

С помощью какой функции можно отобразить текущую дату?

Ваш ответ:

- (•) datetime.datetime.now()
- ( ) datetime.now()
- ( ) datetime.time.today()
- ( ) datetime.date.today()

Ответить

Домашнее задание 1

Написать функцию write_and_read, которая будет записывать в файл текст как параметр функции и читать текст из этого файла и передавать на выход функции.

```
import os



def write_and_read(text):
    file_path = os.path.join(os.path.abspath("/tmp"), "my_file")

    with open(file_path, 'w') as file:
        file.write(text)

    with open(file_path, 'r') as file:
        read_text = file.read()

    return read_text



text = input()
print(write_and_read(text))
```

Домашнее задание 2

Написать функцию changed_div, которая считает отношение делителя к частному. Желательно, чтобы случай с нулевым делимым должен быть обработан с помощью try-except: при ошибке ZeroDivisionError должен возвращаться делитель

```
def changed_div(numerator, denominator):
    try:
        result = round(numerator / denominator, 2)
    except ZeroDivisionError:
        result = numerator
    return result

numerator, denominator = int(input()), int(input())
print(changed_div(numerator, denominator))
```

Домашнее задание 3

Написать функцию fill_specializations, которая принимает список кортежей из специальности и имени и возвращает словарь, где в качестве ключей специальности, а в качестве значений - списки имен. Желательно, чтобы это было реализовано через словарь со значением по умолчанию.

```
from collections import defaultdict
from typing import List, Tuple


def fill_specializations(specializations: List[Tuple[str, str]]):
    specialization_dict = defaultdict(list)
    for specialization, name in specializations:
        specialization_dict[specialization].append(name)
    return specialization_dict


code = []
while data := input():
   code.append(data)
code = "\n".join(code)
exec(code)
```

Домашнее задание 4

Написать функцию fill_missing_values, которая принимает два списка int-ов и делает из них один список кортежей, где в качестве элементов кортежа элементы списков на одинаковых позициях. Если один из список закончился, то в нужно заполнить значение в кортеже единицей.

```
from itertools import zip_longest
from typing import List, Tuple


def fill_missing_values(values_list_1: List[int], values_list_2: List[int]) -
> List[Tuple[int, int]]:
    filled_values = [(x if x is not None else 1, y if y is not None else 1)
for x, y in
                     zip_longest(values_list_1, values_list_2)]
    return filled_values


code = []
while data := input():
    code.append(data)
code = "\n".join(code)
exec(code)
```

Домашнее задание 5

Написать функцию shift_time, которая принимает 2 параметра: количество дней и количество секунд и сдвигает дату и время 01.01.2023 12:30:00 на указанное количество дней и секунд. В качестве выходного значения нужно вывести кортеж: день и секунда от сдвинутого времени.

```
import datetime


def shift_time(days: int, seconds: int):
    base_time = datetime.datetime(2023, 1, 1, 12, 30, 0)
    shifted_time = base_time + datetime.timedelta(days=days, seconds=seconds)
    return shifted_time.day, shifted_time.second


days, seconds = int(input()), int(input())
print(shift_time(days, seconds))
```

Домашнее задание 6

Написать функцию parse_time, которая принимает строку в качестве параметра, которая является временем формата “ГГГГ ММ ДД ЧЧ ММ СС” и парсит эту строку в объект datetime.datetime и сдвигает ее на один день вперед.

```
import datetime


def parse_time(s):
    year, month, day, hour, minute, second = map(int, s.split())
    parsed_datetime = datetime.datetime(year, month, day, hour, minute, second)
    shifted_datetime = parsed_datetime + datetime.timedelta(days=1)
    return shifted_datetime.day

string_datetime = input()
print(parse_time(string_datetime))
```

Домашнее задание 7

Написать функцию most_common_months, которая принимает в качестве параметра список строк, которые являются датами формата “ГГГГ-ММ-ДДТЧЧ-ММ-СС” и некоторое целое число n, и выводит топ n самых частых месяцев этих дат. Желательно, чтобы это было реализовано через Counter из модуля collections.

```
import datetime
from collections import Counter
from typing import List

def most_common_months(dates: List[str], n) -> List[int]:
    months = []
    date_format = '%Y-%m-%dT%H:%M:%S'
    for date_str in dates:
        date = datetime.datetime.strptime(date_str, date_format)
        months.append(date.month)
    month_counter = Counter(months)
    return [x for x in month_counter.keys()][:n]

code = []
while data := input():
   code.append(data)
code = "\n".join(code)
exec(code)
```

Домашнее задание 8

Написать функцию rotate_list, которая принимает список целых чисел и целое число, которое будет задавать, сколько крутить список. Под кручением списка подразумевается забор элемента из конца списка и вставка его в начало списка. Желательно, чтобы это было реализовано через двустороннюю очередь.

```
from collections import deque
from typing import List


def rotate_list(nums: List[int], n: int):
    queue = deque(nums)
    for _ in range(n):
        queue.appendleft(queue.pop())
    return list(queue)


code = []
while data := input():
   code.append(data)
code = "\n".join(code)
exec(code)
```

Домашнее задание 9

Написать класс Twitter, в котором есть следующие методы:

- __init__(self), который инициализирует класс
- post_tweet(self, user_id, tweet_id), который создает новый твит с идентификатором tweet_Id по идентификатору пользователя user_id. Каждый вызов этой функции будет осуществляться с уникальным идентификатором твита. Желательно, чтобы впоследствии твиты можно было бы быстро получить по user_id.
- get_news_feed(self, user_id), который получает 10 последних идентификаторов твитов в ленте новостей пользователя. Каждый элемент в ленте новостей должен быть опубликован пользователями, на которых подписан пользователь. Твиты должны быть упорядочены от самого позднего до самого раннего. Подумайте, как организовать упорядочивание твитов по времени:
    - возможно для этого лучше записывать в структуру данных с твитами
        - информацию о твите (то есть tweet_id)
        - какое то число, отвечающее за время (например какой нибудь счетчик)
    - и это например завернуть в кортеж
- follow(self, follower_id, followee_id), в котором пользователь с идентификатором follower_id подписался на пользователя с идентификатором followee_id. Желательно, чтобы впоследствии подписки можно было бы быстро получить по follower_id.
- unfollow(self, follower_id, followee_id), в котором пользователь с идентификатором follower_id отписался от пользователя с идентификатором followee_id.

```
from typing import List



class Post:

    def __init__(self, post_id: int, tweet_id: int):
        self.post_id = post_id
        self.tweet_id = tweet_id



class User:

    def __init__(self):
        self.follows = []
        self.posts = []

    def add_post(self, post: Post):
        self.posts.append(post)

    def follow(self, user_id: int):
        self.follows.append(user_id)

    def unfollow(self, user_id: int):
        self.follows.remove(user_id)



class Twitter:

    def __init__(self):
        self.users = {}
        self.last_post_id = 0

    def post_tweet(self, user_id: int, tweet_id: int):
        if not (user_id in self.users.keys()):
            self.users[user_id] = User()

        user = self.users[user_id]
        user.add_post(Post(self.last_post_id, tweet_id))
        self.last_post_id += 1
```

```
    def get_news_feed(self, user_id: int) -> List[int]:
        posts: List[Post] = []
        user = self.users[user_id]
        for i in user.follows:
            follow_user = self.users[i]
            follow_posts = follow_user.posts
            posts.extend(follow_posts)

        sorted_posts = []
        for i in range(0, self.last_post_id):
            for post in posts:
                if post.post_id == i:
                    sorted_posts.append(post.tweet_id)
                    break

        return (sorted_posts[::-1])[:10]

    def follow(self, follower_id: int, followee_id: int):
        if not(follower_id in self.users.keys()):
            self.users[follower_id] = User()

        user = self.users[follower_id]
        user.follow(followee_id)

    def unfollow(self, follower_id: int, followee_id: int):
        if not (follower_id in self.users.keys()):
            self.users[follower_id] = User()

        user = self.users[follower_id]
        user.unfollow(followee_id)


code = []
while data := input():
    code.append(data)
code = "\n".join(code)
exec(code)
```