Функціональні вимоги до Easy Study

Для справної роботи проекту необхідно декілька функцій:

1. Функція: зчитування даних.

Опис: зчитує дані з json файлу.

Вхідні дані: json файл.

Джерело даних: користувач.

Вихідні дані: список списків. Де перший список має ключем id та назву курсу.

Пункт збереження: список

Для виконання функції необхідний json файл.

1. Функція: запису даних.

Опис: записує дані в json файл.

Вхідні дані: список списків.

Джерело даних: користувач.

Вихідні дані: json файл.

Пункт збереження: json файл.

Для виконання функції необхідні дані представлені списком списків.

1. Функція: зчитування переліку тем та підтем курсу.

Опис: функція зчитує вказаний користувачем курс і виділяє ключові слова, такі як теми підтеми та заголовки та записує у файл.

Вхідні дані: назва курсу.

Джерело даних: користувач.

Вихідні дані: ключові слова, json file з даними.

Пункт збереження: json файл.

Для виконання функції необхідна назва курсу, поле для вводу даних. Також необхідне з`єднання з інтернетом.

1. Функція: зчитування переліку курсів з Udemy API.

Опис: функція зчитує всі курси, за вказаними користувачем властивостями та повертає всі теми курсів після чого їх обробляє за коефіцієнтом подібності тем і записує у файл, також записує id курсів.

Вхідні дані: тема курсу з json файлу.

Джерело даних: json файл, користувач.

Вихідні дані: перелік тем.

Пункт збереження: json файл.

Для виконання функції необхідні id та пароль користувача Udemy API, файл з темою курсу та коефіцієнт відповідності теми, також введені користувачем властивості бажаних курсів. Також необхідне з`єднання з інтернетом.

1. Функція: зчитування даних про курси.

Опис: за id курсу функція зчитує дані про курси

Вхідні дані: id курсів.

Джерело даних: json файл.

Вихідні дані: дані про всі курси.

Пункт збереження: json файл.

Для виконання функції необхідний перелік id курсів, id та пароль користувача Udemy API. Також необхідне з`єднання з інтернетом.

1. Функція: зчитування відгуків про курс.

Опис: за переліком id курсів функція зчитує відгуки про всі курси.

Вхідні дані: id курсів.

Джерело даних: json файл.

Вихідні дані: оброблений словник курсів.

Пункт збереження: json файл.

Для виконання функції необхідний перелік id курсів, id та пароль користувача Udemy API. Також необхідне з`єднання з інтернетом.

1. Функція: аналізу отриманих даних.

Опис: функція зчитує дані і порівнює їх за повторенням слів і символів в одному і другому курсу, один це заданий користувачем курс, а другий це шуканий курс. По знайденому коефіцієнту будується список з назвами та посиланням на курси.

Вхідні дані: два json файли.

Джерело даних: користувач та Udemy API.

Вихідні дані: список оброблених курсів.

Пункт збереження: список кортежів.

Для виконання функції необхідний перелік id курсів, id та пароль користувача Udemy API. Також необхідне з`єднання з інтернетом.

1. Допоміжна функція: аналіз відгуків про курси.

Опис: якщо користувач бажає може провестися додатковий аналіз курсів на базі відгуків до цих курсів. Отже за одним з json файлів будується середня оцінка курси і вона враховується при виведенні курсів. Проте можна добавити коефіцієнт з яким буде враховано цей бал.

Вхідні дані: два json файли.

Джерело даних: користувач та Udemy API.

Вихідні дані: список оброблених курсів.

Пункт збереження: список кортежів. Для виконання функції необхідний перелік id курсів, id та пароль користувача Udemy API. Також необхідне з`єднання з інтернетом.

Нефункціональні вимоги Easy Study

Зручний користувацький інтерфейс.

Для вводу даних користувачеві необхідна клавіатура та миша.

Програма повинна проходити всі тестові випадки перед виведенням даних користувачеві, якщо ж програма несправна користувачеві повідомляється про це.

Швидка робота програми, через відсутність важких математичних дій.

Програма працює з базою даних Udemy API, тож до неї необхідний доступ.

Користувач не має мати доступу до ключів і id проекту Udemy API, тож вони мають бути закодовані. Крім цього користувач немає доступу до коду програми і немає прав адміністраторів.

Можливості файлів json, xml та html

Можливості файлів json.

Json файл – простий текстовий файл, який зрозумілий для людини і може зчитуватися будь якими мовами програмування. Одним з головних використання цього типу, являється при написанні веб додатків та веб сервісів. Оскільки комп’ютерам легше працювати з ним, ніж наприклад з xml. При цьому він займає менше місця і при виклику зразу інтерпретується в об’єкт за допомогою JavaScript.

В python формується як словник з ключ – значення.

Приклад.

{

"count": 55776,

"next": "https://www.udemy.com/api-2.0/courses/?page=2&page\_size=12",

"previous": null,

"results": [

{

"\_class": "course",

"id": 600864,

"title": "Изучите HTML и CSS за несколько часов",

"url": "/html-css-course/",

"is\_paid": true,

"price": "19,99 $",

}]

}

Плюси json це зручність, зрозумілість, легке налагодження програми, виявлення помилок та їх усунення та безпека. Також можна виділити простоту створення json файлу.

Отже, json – є зручним та простим способом представлення даних, як для людини так і для зчитування комп’ютером, крім того він займає менше місця та легко створюється. Також він підтримується більшістю доступних API.Він легко використовується та розширюється.

Проте з мінусів це відносно недавнє його створення і він не такий розповсюджений, як наприклад xml та той самий html.

Можливості xml.

XML – розширена мова розмітки, оточена тегами, як і html проте це єдина їхня подібність. XML теги визначають чим є дані, їхню структуру, також вони відповідають за те, як обробляти дані. Елементи в xml це є дані з прикріпленою міткою, розташовані ієрархічно. Зазвичай розмітка додає читабельності, проте xml файл потребує більше місця.

В загальному xml досить старий варіант розмітки, проте він все ще потрібний, наприклад для роботи з сервером, оскільки той же json не підходить, вона є достатньо структурована, що з одного боку добре, а з іншого можна заплутатись. Приклад використання xml:

<animal><meat-eating>Bear</meat-eating></animal>

Де продемонстрована ієрархія обєктів. Проте він є більш складним, як json. Проте одним з головних плюсів є його повна і активна підтримка. При виникненні запитань, завжди можна знайти на них відповідь.

Більшість документів для вебу можна створювати за допомогою xml. Також він дозволяє перевизначити та змінювати їх. Він дозволяє розмічувати різні стандартні документи, які містять текст або посилання.

Можливості html.

З самого свого початку html дозволяв створювати веб сторінки на базі однієї платформи. Проте хоча html і необхідний для відображення графічних сторінок, проте він не має можливостей для створення ієрархічно структурованих даних. Для чого і створили xml. Html також забезпечує графічне відображення сторінки. Проте html має низку мінусів, оскільки немає пошуку по візуальному представленню, тобто ви не зможете найти сайт, який хочете, тільки за допомогою html протоколів. Та шукати в інтернеті можна, але не завдяки html, а html просто показує По суті html тільки показує як повинна виглядати веб сторінка. Загальний вигляд такий:

<html>

<head>

<title> </title>

</head>

<body>

</body>

</html>

Тож в html немає єдиного стандарту, який може забезпечити перечисленні вище операції. Тому це застарілий, хоча все ще можливий варіант, для представлення даних. На зміну йому прийшов xml. Він більш придатний для парсингу сайтів.

Я буду використовувати json файли, оскільки цей метод є найбільш новим, він досить непогано підтримуваний та найбільш зрозумілий, також його досить зручно аналізувати та знаходити місце помилки.

Дані, які отримує програма від Udemy API

Як я писав раніше сама програма викликає чотири запити, отже отримує чотири різні json файли. З якими досить легко працювати. Загалом три з них мають вигляд, такий самий як в цьому прикладі.

{

"count": 55814,

"next": "https://www.udemy.com/api-2.0/courses/?page=2&page\_size=12",

"previous": null,

"results": [

{

"\_class": "course",

"id": 600864,

"title": "Изучите HTML и CSS за несколько часов",

"url": "/html-css-course/",

"is\_paid": true,

"price": "19,99 $",

"visible\_instructors": [

{

"\_class": "user",

"id": 6607464,

"title": "Sergey Nikonov",

"name": "Sergey",

"display\_name": "Sergey Nikonov",

"job\_title": "Senior web-developer",

"image\_50x50": "https://udemy-images.udemy.com/user/50x50/6607464\_c1d6.jpg",

"image\_100x100": "https://udemy-images.udemy.com/user/100x100/6607464\_c1d6.jpg",

"initials": "SN",

"url": "/user/31513/"

}

],

"image\_125\_H": "https://udemy-images.udemy.com/course/125\_H/600864\_c561\_3.jpg",

"image\_240x135": "https://udemy-images.udemy.com/course/240x135/600864\_c561\_3.jpg",

"is\_practice\_test\_course": false,

"image\_480x270": "https://udemy-images.udemy.com/course/480x270/600864\_c561\_3.jpg",

"published\_title": "html-css-course",

"predictive\_score": null,

"relevancy\_score": 1.0450184,

"input\_features": null,

"lecture\_search\_result": null

}

Цей файл має вигляд ключ – значення. Спочатку йдуть дані про курс його номер, посилання на нього та минулий курс. Дальше йде детальніша інформація про нього, така як його id, клас, заголовок, ціну та інструкторів по цьому курсі. Дальше йде інформація про інструкторів курси на яких вони працюють та посади на цих курсах, також є посилання на фотографії викладачів та самого курсу. Це відповідь на courses-list

Власне я писав, що це відповідь на courses-list – це дійсно так, якщо введені властивості такі, щоб вивести тільки один курс, проте це також відповідь на courses-detail. Швидше за все він нам дійсно не потрібний, оскільки courses-list надає таку ж саму інформацію, як і courses-detail – тож це лишня трата часу і місця.

Publiccurriculum-list – дає нам файл з дуже ретельною інформацією про курс, про кожен розділ та заголовок коли він був створений, ким був створений його клас, опис, стан завантаження та чи безплатний розділ, також є інформацію наприклад про прикріплене відео, дату відео назву та посилання, також є порядок сортування. Є назва теми також назва на англійській. Також є інформація чи можна скачати ту чи іншу частину курсу, наприклад відео. У кожного розділу, заголовку теми та відео є свій id, за яким можна робити пошук.

Також, я забув сказати, що на кожну відповідь йде статус відповіді. Тобто якщо все пройшло правильно буде статус 200, якщо ні 400 і є різні типи помилок тобто можна написати try: except: не заглиблюючись у відповідь і щоб помилка, не виникала десь всередині.

І останній варіант це course-review. Повертає приблизно так:

{

"count": 991,

"next": "https://www.udemy.com/api-2.0/courses/600864/reviews/?page=2",

"previous": null,

"results": [

{

"\_class": "course\_review",

"id": 14978178,

"title": "",

"content": "",

"rating": 5,

"created": "2018-03-25T19:44:58Z",

"modified": "2018-03-26T01:45:54Z",

"user": {

"\_class": "user",

"id": 19393246,

"title": "Golovach Inna",

"name": "Golovach",

"display\_name": "Golovach Inna"

}

}

}

Тобто йдуть дані про відгук його номер, посилання на минулий та наступний, якщо немає на наступний то null. Дальше йде клас в нашому випадку course review, id та що написано(title, content), rating, коли він створений та остання дата змінення цього відгуку. Дальше йде інформація, про користувача, що залишив цей відгук, його клас, id, ім’я та справжнє ім’я, перше це нікнейм.