**6-зертхана: Функционалды реактивті бағдарламалау (FRP)**

**Мақсат:**

Python контекстінде функционалдық реактивті бағдарламалау (FRP) принциптерін үйреніңіз және қолданыңыз. Жұмыстың мақсаты - деректер ағындарын, реактивті айнымалыларды және асинхронды оқиғаларды өңдеуді қоса алғанда, FRP негізгі тұжырымдамаларын түсіну және жауап беретін және модульдік қосымшаларды жасау үшін осы тұжырымдамаларды пайдалану дағдыларын дамыту.

Тапсырмалар:

1. FRP негіздері:

* Ағындар, сигналдар және реактивті айнымалылар сияқты негізгі FRP тұжырымдамаларын үйреніңіз.
* реактивті өзгерістерді тарату және тәуелділікті басқару механизмдерін талдау.

2. Қарапайым FRP сценарийлерін іске асыру:

* өзгермелі деректерді өңдеу және көрсетумен байланысты мәселелерді шешу үшін FRP қолдану.
* Пайдаланушы әрекеттеріне немесе сыртқы оқиғаларға жауап ретінде реактивті деректерді жаңартуды көрсететін мысалдарды әзірлеу.

3. Қолданыстағы қолданбалармен интеграция:

* Қолданыстағы Python қолданбаларына FRP тәсілдерін біріктіру жолдарын қарастыру.
* FRP қолдану арқылы жауап беруді жақсарту және код күрделілігін азайту мүмкіндіктерін талдау.

4. FRP-дегі қосымша тақырыптар:

* Қателерді өңдеу, ағынды біріктіру және асинхронды операцияларды қоса алғанда, FRP жүйесіндегі озық әдістер мен үлгілерді үйреніңіз.
* көптеген деректер көздерін тиімді өңдеуге қабілетті күрделі реактивті жүйелерді жасау.

5. Сыни талдау және рефлексия:

* Нақты әлемдегі қолданбалар контекстінде FRP артықшылықтары мен шектеулерін бағалау.
* Нақты бағдарламалық тапсырмалар үшін тәсілдер мен құралдарды таңдауда сыни тұрғыдан ойлауды дамыту.

Зертханалық жұмыстың маңыздылығы:

Зертхана студенттерді қазіргі заманғы бағдарламалық қамтамасыз етуді әзірлеуде маңыздырақ болып келе жатқан функционалдық реактивті бағдарламалау парадигмасымен таныстыруға арналған. FRP интерактивті және асинхронды қолданбаларды құруға арналған талғампаз шешімдерді ұсынады, бұл күй мен деректер ағынын басқаруды жеңілдетеді. Жұмыс реактивті жүйелерді терең түсінуді дамытады және анағұрлым жауап беретін және модульдік қосымшаларды жобалау дағдыларын жетілдіреді.

**Жеке тапсырмалар:**

Әрбір студентке топ тізіміндегі санына сәйкес бірегей тапсырма беріледі (SSO қараңыз).

Функционалды реактивті бағдарламалау (FRP) өзгерістерге жауап беретін жүйелерді құру үшін реактивті және функционалды бағдарламалау идеяларын біріктіреді.

1. Реактивті санауыш

- Пайдаланушы оқиғаларына жауап ретінде жоғарылайтын немесе төмендететін реактивті есептегішті әзірлеу.

2. FRP істер тізімі

- Тапсырмаларды қосуға және жоюға жауап беретін FRP көмегімен қарапайым істер тізімі қолданбасын жасаңыз.

3. Асинхронды деректерді сұрау

- Сыртқы API интерфейсінен асинхронды деректерді алуды және оны FRP көмегімен көрсетуді жүзеге асыру.

4. Реактивті дауыс беру

- Дауыстар нақты уақыт режимінде есептелетін дауыс беру жүйесін жасаңыз.

5. «Тас, қағаз, қайшы» ойыны

- Реактивті «Тас, қағаз, қайшы» ойынын жасаңыз, онда компьютер таңдауы пайдаланушының таңдауына жауап ретінде жасалады.

6. FRP сырғытпасы

- Басқа интерфейс құрамдастарын реактивті түрде жаңартып отыратын мәнді өзгерту үшін сырғытпа жасаңыз.

7. Jet Currency Converter

- Валюта бағамдары өзгерген кезде реактивті түрде жаңартылатын валютаны айырбастауға арналған қосымшаны жасаңыз.

8. Температура мониторы

- Өзгерістерге жауап беретін және сыни мәндерге жеткенде ескертулер беретін температураны бақылау жүйесін енгізіңіз.

9. Реактивті кесте сүзгісі

- Критерийлер өзгерген кезде нақты уақытта сүзгілер қолданылатын деректері бар кестені жасаңыз.

1. Музыка ойнатқышы
   * Жолды ойнатуды, кезекті және аудио параметрлерін басқаратын реактивті музыка ойнатқышын жасаңыз.
2. Реактивті чат қолданбасы
   * Хабарламалар нақты уақытта көрсетілетін қарапайым чат қолданбасын жасаңыз.
3. FRP анимациясы
   * Пайдаланушы әрекеттеріне немесе деректер өзгерістеріне жауап беретін анимациялар жасау үшін FRP пайдаланыңыз.
4. FRP жүктеп алу барысының көрсеткіші
   * Файлды жүктеу процесі кезінде реактивті түрде жаңартылатын орындалу көрсеткішін жасаңыз.

from rx import interval, operators as op  
from time import sleep  
def simulate\_download():  
 total\_size = 1000 # Общий размер файла  
 downloaded = 0 # Количество загруженных байтов  
 while downloaded < total\_size:  
 downloaded += 100  
 yield downloaded / total\_size  
 sleep(1)  
progress\_stream = interval(1).pipe(op.map(lambda \_: next(simulate\_download())), op.take\_while(lambda progress: progress < 1))  
progress\_stream.subscribe(lambda progress: print(f"Progress: {progress \* 100:.1f}%"))  
while True:  
 pass

import sys  
import requests  
def download(url, fil\_ename):  
 with open(fil\_ename, 'wb') as f:  
 res = requests.get(url, ag\_yn=True)  
 jami = res.headers.get('content-length')  
  
 if jami is None:  
 f.write(res.content)  
 else:  
 zhuktelgen = 0  
 jami = int(jami)  
 for data in res.iter\_content(chunk\_size=max(int(jami / 1000), 1024 \* 1024)):  
 zhuktelgen += len(data)  
 f.write(data)  
 done = int(50\*zhuktelgen / jami)  
 sys.stdout.write('\r[{}{}]'.format('█' \* done, '.' \* (50 - done)))  
 sys.stdout.flush()  
 sys.stdout.write('\n')

Python файлын жүктеп алуға орындалу жолағын қосу үшін кітапхананы пайдалана аласыз **tqdm**. Бұл қадамдық нұсқаулық процесті үш негізгі бөлікке бөледі: HTTP сұрауын жасау, деректерді тасымалдауды өлшеу және консольде орындалу барысын көрсету.

**1-қадам: HTTP сұрауын дайындау** HTTP сұрауын бастау және орындалу жолағы жоқ файлды жүктеп алу үшін кітапхананы пайдалануға болады **requests**. Мысалы:

import requests

url = 'https://www.example.com/file.txt'

with requests.get(url) as r:

with open('download.txt', 'wb') as file:

file.write(r.content)

Бұл негізгі код файлды жүктеп алып, оны жергілікті түрде сақтайды. Дегенмен, оған прогресс жолағы жоқ.

**2 және 3-қадамдар: Деректерді тасымалдауды өлшеу және көрсету** Орындалу жолағын енгізу үшін кітапхананы **tqdm**кітапханамен бірге пайдалануға болады **requests**. Біріншіден, қолмен жүзеге асыру:

import time

import requests

url = 'https://www.example.com/file.txt'

with requests.get(url, stream = True) as r:

with open('download.txt', 'wb') as file:

total\_size = int(r.headers.get('Content-Length'))

chunk\_size = 1

for i, chunk in enumerate(r.iter\_content(chunk\_size = chunk\_size)):

c = i \* chunk\_size / total\_size \* 100

print(f "\r{round(c, 4)}%", end = '')

time.sleep(0.1)

Бұл код консольде жүктеу пайызын көрсете отырып, файл мазмұны бойынша қайталанады. Дегенмен, оның кейбір кемшіліктері бар, соның ішінде әрбір бөлік үшін кідіріс.

**tqdm Implementation** Қолданылатын жеңілдетілген тәсіл **tqdm**:

import os

import requests

import shutil

from tqdm.auto

import tqdm

url = 'https://www.example.com/file.txt'

with requests.get(url, stream = True) as r:

total\_length = int(r.headers.get("Content-Length"))

with tqdm.wrapattr(r.raw, "read", total = total\_length, desc = "") as raw:

with open(f "{os.path.basename(r.url)}", 'wb') as output:

shutil.copyfileobj(raw, output)

Бұл кодта **tqdm**жалпы орындалу барысы, динамикалық түрде жаңартылған орындалу жолағы, жүктеп алынған жалпы байттар, өткен уақыт және деректерді тасымалдау жылдамдығы сияқты мәліметтерді қоса алғанда, орындалу жолағын көрсетудің қысқаша әдісі берілген.

Модульдің **tqdm**жеңіл ізі бар және **colorama**дисплей мәтінінің түсін реттеуге мүмкіндік беретін теңшеу үшін кітапхананы қажет етеді. Бұл тәсіл Python-да файлдарды жүктеп алу кезінде пайдаланушы тәжірибесін айтарлықтай жақсартады.

Қажетті кітапханаларды орнату үшін келесі әрекеттерді орындаңыз:

pip install tqdm colorama

[github.com](https://github.com/tartley/colorama)**colorama** кітапханасын және Python [github.com](https://github.com/tqdm/tqdm) кітапханасын зерттеңіз .**tqdm**

Мұндағы жалғыз шынайы сиқыр консольге ағынмен жіберу үшін sys.stdout.write әдісін, .1 миллисекундқа үзіліс жасау үшін time.sleep әдісін, содан кейін курсорды бірінші таңбаға қайта жылжыту үшін sys.stdout.flush әдісін пайдалану болып табылады. . Бұл ұқыпсыз, тіпті оның қазіргі көзқарасында да айтарлықтай жақсартылуы мүмкін және әрбір бөлік өлшемі үшін .1 мс кідіріс қосады — бұл жоғарыдағы мысалдағы бір байт. Неғұрлым жеңілдетілген тәсілді қарастырайық. Бұл жерде мәтінмәндік басқарушыларды пайдалану өз қалауынша, сұрау.get қоңырауы онсыз орындалады — оны әдет күші деп атаңыз.

1. Реактивті пішінді басқару
   * FRP көмегімен нақты уақыттағы деректерді тексеру арқылы пішінді әзірлеу. 15. FRP бойынша «Жылан» ойыны
   * «Жылан» ойынының реактивті нұсқасын жасаңыз, онда жыланның қозғалысы және жаңа элементтердің пайда болуы оқиғалар ағымымен басқарылады.

**Бағалау критерийлері:**

* + Жеке есепті шешу үшін код жазу: 1 ұпай
  + Қорғау кезінде жазылған кодты түсіндіру және түсіну: 2 ұпай
  + Мұғалім таңдаған теориялық сұрақтардың біріне жауап: 1 ұпай

**Дайындық сұрақтары:**

1. Функционалды реактивті бағдарламалау дегеніміз не және оны қандай жағдайларда қолдану керек?

- Мақсаты: FRP анықтамасын және оны қолдану сценарийлерін түсіну.

2. FRP негізгі ұғымдары қандай?

- Мақсаты: Деректер ағындары және реактивті күй сияқты FRP құрылыс блоктары туралы білімді бағалау.

3. FRP оқиғаны өңдеудің дәстүрлі тәсілінен несімен ерекшеленеді?

- Мақсат: FRP және басқа оқиғаларды өңдеу үлгілерінің арасындағы айырмашылықтарды түсіну.

4. FRP бағдарламалық жасақтаманы әзірлеушілерге қандай артықшылықтар береді?

- Мақсаты: Күйді басқаруды жеңілдету және код модульділігі сияқты артықшылықтарды түсінуді тексеру.

5. FRP енгізу үшін қандай Python кітапханаларын пайдаландыңыз және неліктен?

- Мақсаты: Оқушылардың қол жетімді құралдар туралы білімін және мәселені шешу үшін таңдауын бағалау.

6. FRP-де деректер ағынын жүзеге асыру жолын түсіндіре аласыз ба?

- Мақсаты: Студенттің FRP-де деректер ағындарын құру және басқару механизмін түсінуін тексеру.

7. FRP асинхронды операцияларды қалай өңдейді?

- Мақсаты: Оқушылар FRP асинхронды кодты өңдеуге және жанама әсерлерді өңдеуге қалай мүмкіндік беретінін түсінеді.

1. FRP пайдалануда қандай қиындықтар болуы мүмкін және оларды қалай жеңуге болады?

Мақсаты: Оқушылардың мүмкін болатын қиындықтар мен оларды шешу жолдары туралы хабардарлығын анықтау.

1. Қолданбаңызда немесе ойыныңызда реактивті әрекетті қалай тексердіңіз? - Мақсаты: Реактивті жүйелерді сынау әдістерін түсіну.
2. FRP-де өнімділік мәселелері қандай болуы мүмкін және оларды қалай оңтайландыруға болады?
   * Мақсаты: Реактивті қолданбалардың өнімділігі және оңтайландыру әдістері туралы білімдерін бағалау.
3. FRP-де жалқау бағалау қолданылатын мысал келтіре аласыз ба?
   * Мақсаты: Жалқау бағалауды түсінуді және оны FRP контекстінде қолдануды тексеру.
4. FRP қолданбадағы күйді басқаруға қалай әсер етеді?
   * Мақсаты: FRP жағдайды басқаруға әсерін және оның нақты уақыттағы өзгерістерін түсіну.
5. FRP-мен жұмыс істегенде қандай дизайн үлгілері пайдалы болуы мүмкін?
   * Мақсаты: FRP-ны тиімді толықтыратын дизайн үлгілері туралы білімді бағалау.
6. Қолданыстағы қолданбаларға FRP қалай біріктіруге болады?
   * Мақсат: кодты толығымен қайта жазбай-ақ, ағымдағы жобаларға FRP біріктіру тәсілдерін түсіну.
7. FRP қолданбаңыздағы деректер ағындарына жазылудан бас тартуды қалай басқарасыз?
   * Мақсаты: Реактивті қолданбаларда жазылудың өмірлік циклін дұрыс басқару туралы білімдерін тексеру.

**Пример для решения**

Задача: Реактивное Отслеживание и Отображение Данных Температуры

Представьте, что у вас есть поток данных о температуре, поступающих из различных источников (например, датчиков температуры). Задача состоит в том, чтобы реализовать простую реактивную систему на Python, которая отслеживает эти данные в реальном времени и обновляет среднюю температуру при каждом новом поступлении данных.

Цель задачи:

Создать реактивную систему, которая подписывается на поток данных о температуре, вычисляет и отображает обновленное среднее значение температуры при каждом изменении в потоке.

Решение:

Для этой задачи мы можем использовать библиотеку RxPy (Reactive Extensions for Python), которая предоставляет инструменты для реактивного программирования.

Сначала установите RxPy, если она еще не установлена:

pip install rx

Теперь давайте создадим решение:

import rx

from rx import operators as ops

# Эмуляция потока данных о температуре

temperature\_data = rx.of(20, 21, 22, 21, 20, 19, 20, 21, 22, 23)

# Реактивное вычисление средней температуры average\_temperature = temperature\_data.pipe(

ops.scan(lambda acc, temp: (acc[0] + temp, acc[1] + 1), (0, 0)), ops.map(lambda acc: acc[0] / acc[1])

)

# Подписка и вывод результата average\_temperature.subscribe( lambda avg\_temp: print(f"Средняя температура: {avg\_temp:.2f}°C")

)

#### Объяснение:

* Мы создаём поток `temperature\_data`, который эмулирует поступление данных о температуре.
* `pipe` используется для создания цепочки операторов, обрабатывающих поток данных.
* С помощью `ops.scan` мы накапливаем сумму температур и количество измерений для дальнейшего вычисления среднего значения.
* `ops.map` преобразует накопленные данные в среднее значение температуры.
* `subscribe` используется для отслеживания результатов и вывода обновленного среднего значения температуры.

Это решение демонстрирует основные принципы FRP: реактивное отслеживание изменений в данных и декларативное определение логики обработки данных. Оно показывает, как FRP позволяет легко реагировать на изменяющиеся данные и создавать отзывчивые приложения.