Лабораторийн тайлан №9

Дарт программчлалын хэл ба түүний хэрэглээ

МТЭС, МКУТ, Программ хангамж

Г.Батням, 22B1NUM5578

1.Ажлын зорилго

Энэ лабораторийн ажлаар Dart программчлалын хэл болон UI Framework бүхий Flutter хэлийг ашиглан өгөгдсөн даалгавруудыг хийж гүйцэтгэнэ.

2.Онолын судалгаа

**Dart**– Android, iOS, вэб, desktop гэх мэт олон платформд зориулсан аппликейшнүүдийг хурдан бүтээхэд зориулагдсан, хэрэглэгчийн туршлагыг сайжруулсан, обьект хандалтат, орчин үеийн програмчлалын хэл юм. Дарт хэлийг Google хөгжүүлсэн.

**Dart –** Хэрэглэгчид зориулсан оновчтой, объект хандалтат програмчлалын хэл. Сүүлийн үед **Flutter**-ын ачаар алдартай болсон. Гэхдээ зөвхөн **Dart** ашиглаад бүрэн хэмжээний апп хийхэд хүндрэлтэй, учир нь олон зүйлийг өөрөө зохицуулах шаардлагатай болдог.

**Flutter** – **Dart** хэлийг ашигладаг **framework** юм. **Flutter**-ийн тусламжтайгаар Android, iOS, вэб, desktop гэх мэт платформууд дээр апп бүтээх боломжтой. Энэ **framework** нь апп хийхэд хэрэгтэй олон бэлэн хэрэгсэлтэй.

**Flutter** – Ашиглан чанартай апп бүтээхийг хүсвэл Дарт хэлийг сайн ойлгох шаардлагатай.

**Dart’s features**

* Натив код эсвэл JavaScript руу хөрвүүлэх боломжтой.
* Null safety, asynchronous programming гэх мэт орчин үеийн програмчлалын боломжуудыг санал болгодог.
* Сервер болон backend талд ч ашиглаж болно.
* Обьект хандалтат програмчлалын хэл.

**Програмчлалын үндсэн нэр томьёо**

* Кодын мөр бүхний дараа **;** тавьдаг.
* High-Level Programming Language
* **Compiler (Компайлэр):** Өндөр түвшний програмчлалын хэлийг машин хэл рүү хөрвүүлдэг програм.
* **Syntax (Синтакс):** Програмчлалын хэлийг зөв бичих дүрэм, хэв маяг.
* Dart нь платформ хамааралгүй бөгөөд **Windows, macOS, Linux** гэх мэт бүх гол үйлдлийн системүүдийг дэмждэг.
* Dart нь dart2js нэртэй компайлэртай бөгөөд энэ нь Dart кодыг JavaScript болгон хөрвүүлдэг.

**Нэмэлт мэдээлэл:** **(Environment Variable - Path)**Path гэдэг нь таны Windows системд ямар замуудаар гүйцэтгэх файл (executable) хайхыг заадаг орчны хувьсагч юм. Товчхондоо:

👉 Та коммандын мөр дээр (cmd) ямар нэгэн программын нэр бичихэд, тэр программаа аль фолдероос хайх вэ? гэдгийг энэ Path хувьсагч шийддэг. Тухайн төхөөрөмж (программ)-ийг cmd, PowerShell, VS Code terminal гэх мэт газруудаас **шууд ажиллуулах боломжтой болгох** гэсэн санаа юм.

**dart pub get -** Энэ нь таны pubspec.yaml дээр үндэслэн .dart\_tool/package\_config.json файл үүсгэж, таны package-ийг бүртгэх болно.

**Num –** Бүхэл болон бутархай тоог хоёуланг нь хадгалах үед ашиглана.

**Dart Хувьсагчийн Дүрэм**

* Зөвхөн үсэг болон тоо агуулж болно
* Нэрийн **эхэнд тоо тавьж болохгүй**.
* Тусгай тэмдэгтүүдээс зөвхөн **\_ (underscore)** ба **$ (dollar)** тэмдэгтүүдийг ашиглаж болно.
* **const** түлхүүр үгийг ашиглан тогтмол зарлаж болно.
* **lowerCamelCase** ашиглан нэрлэнэ.

**Runes -** Юникод тэмдэгтүүдийн утга хадгалдаг.

**Multi-Line-String –** Олон мөртэй текстийг хадгалах үед та гурван хашилт ашигладаг.   
 ***Жишээ нь:*** ***String multiLineText = ''' Энэ бол олон мөртэй текст гурван нэгэн хашилттай. Би энд бичиж байна.'''***

**Raw string** – Тусгай тэмдэгтүүдийг **ажиллуулахгүйн** тулд r үсгийг ашигладаг.

***Жишээ нь:*** ***String withRawString = r"Үнэ нь \t $price"***

***Үр дүн: Raw: Үнэ нь \t $price хэвлэгдэнэ***

**Type Conversion –** Өгөгдлийн төрлийг **parse** ашигланхөрвүүлж болдог.

1. String-ийг Int рүү Хувиргах

***Жишээ нь:***   
***String strvalue = "1";***

***int intvalue = int.parse(strvalue)***

1. String-ийг Double рүү Хувиргах

***Жишээ нь:  
String strvalue = "1.1";***

***double doublevalue = double.parse(strvalue);***

1. Int-ийг String рүү Хувиргах

***Жишээ нь:  
int one = 1;***

***String oneInString = one.toString()***

1. Double-ийг Int рүү Хувиргах

***Жишээ нь:***

***double num1 = 10.01;***

***int num2 = num1.toInt();  
тоймлолгүйгээр шууд бүхэл утгыг авна***

**Set –** Энэ нь дотроо давхцахгүй өгөгдлийг агуулдаг.

**Map -** Map бол key-value хослолыгхадгалдаг объект ба тухайн элементэд key – г бичин ханддаг.

**Var** – түлхүүр үгийг ашиглавал өгөгдлийн төрөл автоматаар олддог.

**Runes** – Ашиглан String-ийн Юникод утгуудыг олж мэдэх боломжтой.

***Жишээ нь:*** ***String value = "ASvhjvhu";***

***print(value.runes);   
Үр дүн:*** ***(65, 83, 118, 104, 106, 118, 104, 117)***

**runtimeType** – Dart -д хувьсагчийн төрөл ямар байгааг шалгахын тулд ашигладаг.

**Optionally Typed Language -** Төрлийн сонголт нь статик болон динамик төрлийн хооронд ялгаа үүсгэдэг.

* **Статик төрөл (Statically Typed):** Программын бичих үед хувьсагчийн төрлийг тодорхойлдог. Компилятор кодыг хурдан шалгаж, алдааг илрүүлдэг.

***Жишээ нь:  
var myVariable = 50; // `int` төрөл***

***myVariable = "Hello"; // Энэ нь алдаа үүсгэнэ***

* **Динамик төрөл (Dynamically Typed):** Програмыг ажиллуулах үед хувьсагчийн төрлийг тодорхойлдог.

***Жишээ нь:***

**dynamic myVariable = 50; // `dynamic` төрөл**

**myVariable = "Hello"; // асуудалгүйгээр текстийг өгч болно**

**Операторууд**

1. Тоо бодох операторууд (Arithmetic Operators)
   * **~/** - Бүхлээр хуваах бол ашиглана.
2. Инкремент ба Декремент операторууд (Increment and Decrement Operators)
   * **Var++ - Дараа нь нэмэгдүүлдэг  
     ++Var - Өмнө нь нэмэгдүүлдэг**
3. Тушаал өгөх операторууд (Assignment Operators)
   * **+= Хувьсагчдын утгыг нэмэгдүүлээд онооно.**
4. Логик операторууд (Logical Operators)
   * &&, ||, ! үйлдлүүд
5. Харьцуулах операторууд (Relational Operators)
   * != 2 зүйлийн утгыг харьцуулахдаа ашигладаг.
6. Төрлийг шалгах операторууд (Type Test Operators)
   * Is, is! объект нь тодорхой төрлийнх мөн эсэхийг гэдгийг шалгадаг.

**Input –** Хэрэглэгчээс оролтын утгыг авахын тулд **dart:io** багцын импортыг ашиглах хэрэгтэй. **stdin.readLineSync()** ашиглан оролтын утгыг авна.  
**String –** ийн хувьд **stdin.readLineSync()** шууд ашиглаж болно.  
**Тоо** – ны хувьд **double.parse(stdin.readLineSync()!),int? number = int.parse(stdin.readLineSync()!);** тус тус ашиглана**.  
*Тайлбар:***DartPad дээр хэрэглэгчийн оруулгыг авах боломжгүй. Хэрэглэгчийн оруулга бүхий программыг ажиллуулахын тулд таны компьютерт Dart-г ажиллуулах орчин байх шаардлагатай.

***dart --enable-asserts .\prctice\_2.dart*** – Assert буюу тухайн кодыг шалгах үед ашиглагдана.

**Функц –** Функцийг хэрэгжүүлэхдээ нэрийг нь **lowerCamelCase** – р бичдэг.

**Онцгой төрөл –** Авах параметрийн утга заавал оруулах шаардлагатай биш байж болно.

***Жишээ нь:******void printInfo(String name, String gender, [String title = "sir/ma'am"])  
Дуудахдаа: printInfo("John", "Male");  
printInfo("Kavya", "Female", "Ms.");****тус тус дуудаж болно.*

**Required** – Авах утгаа заавал оноох хувьсагчтай хамт тодорхойлох ёстой.

***Жишээ нь:void printInfo1({required String name, required String gender})  
Дуудахдаа: printInfo(gender: "Male", name: "John");***

**Anonymous function –** Дотоод функц байдлаар тодорхойлон main дотор зарлаж ашиглах боломжтой.

***Жишээ нь:var cube = (int number) {***

***return number \* number \* number;***

***};  
cube(2);  
Үр дүн:8;***

**Arrow Function –** Функцийн их хаалт байхгүйгээр үйлдлээ хийгээд => ашиглан шууд утгаа буцаадаг функцийг хэлнэ.

***Жишээ нь: int add(int n1, int n2) => n1 + n2;  
add(12, 23);***

**Global scope –** Нийт программын хүрээнд ашиглах боломжтой хувьсагчийг хэлнэ.

**Math Random –** Math санг ашиглан random integer болон double boolean утгыг үүсгэх боломжтой.Доорх жишээ нь ерөнхий жишээ юм.

***Жишээ№1:0 – 9 ийн хооронд random тоо үүсгэх 10 өөрөө орохгүй***

***Random random = new Random();***

***int randomNumber = random.nextInt(10); // from 0 to 9 included***

***print("Generated Random Number Between 0 to 9: $randomNumber");***

***Жишээ№2: 10 – 20 ийн хооронд random тоо үүсгэх***

***int min = 10;***

***int max = 20;***

***int randomnum = min + Random().nextInt((max + 1) - min);***

***Жишээ№3: Тодорхой тооны элементтэй list үүсгэх гэхдээ index ашиглахгүй.***

***List<int> randomList = List.generate(10, (\_) => Random().nextInt(100)+1);***

***Жишээ№4: Тодорхой тооны элемент тэй list үүсгэх гэхдээ index ашиглана.Index дээр тулгуурлан тооцоолол хийгдэнэ.***

***List<int> indexedList = List.generate(10, (index) => index \* 2);***

**List -** Холимог төрөлтэй болон программд ашиглагддаг төрлүүдээр list – ийг үүсгэх боломжтой.Мөн list нь дараах боломжуудтай.

* **first**: эхний элементийг авах
* **last**: сүүлийн элементийг авах.
* **isEmpty**: List – д утга байхгүй тохиолдолд **true** эсрэг тохиолдолд **false** буцаана.
* **isNotEmpty**: List – д утга байвал **true** эсрэг тохиолдолд **false** буцаана.
* **length**: List – д байгаа элементийн тоо.
* **reversed**: List – ийн бүх утгыг хойноос урагш байрыг сольно.
* **single**: Зөвхөн нэг элементийг буцаана.
* **add** – List – д нэг элемент нэмнэ.
* **addAll()** – List – д List нэмэх үед ашиглана.
* **Insert()** – Зөвхөн нэг утгыг хаана оруулах index – ийн хамт зааж өгдөг.
* **InsertAll()** – List – ийг хаана оруулах index – ийн хамт зааж өгдөг.
* **Remove() –** Устгах элемент ийн утгыг оруулж өгдөг.
* **RemoveAt() –** Устгах элемент ийн индексийг оруулж өгдөг.
* **RemoveRange() –** Устгах элементүүдийн заагласан индексийг оруулж өгдөг.
* **RemoveLast() –** Сүүлийн элементийг устгадаг.
* **List.foreach((too) => print(too)) –** Тухайн list – ийн бүх элемент ээр тодорхой нэг үйлдэл хийхийн тулд ашигладаг утга буцаадаггүй.
* **List.map((too) => too \* 2) –** Тухайн list – ийн бүх элемент ээр тодорхой нэг үйлдэл хийгээд шинэ list – ийг буцаадаг.  
  ***Жишээ нь: элемент бүрийг 2р үржүүлэх  
  List<int> list = [10, 20, 30, 40, 50];  
  var douledList = list.map((n) => n \* 2);***
* **MergeList -**2 list – ийг нэгтгэхдээ шинэ list – нд хадгалах ёстой ба **[…names, …ages]** 2 list – ээ тодорхойлно.  
  ***Жишээ нь:Нэгтгэх  
   List<String> names = ["Raj", "John", "Rocky"];  
   List<String> names2 = ["Mike", "Subash", "Mark"];  
   List<String> allNames = [...names, ...names2];***
* **Condition in List –** Нөхцөлтэйгөөр тухайн утгыг list – д байх эсэхийг тодорхойлж болно.
* **Where in List –** Тухайн list – ийн элемент бүрийн хувьд нөхцөл шалгах үед ашиглагдана.  
  ***Жишээ нь:Тэгш тоотой list үүсгэх  
  List<int> numbers = [2,4,6,8,10,11,12,13,14];  
  List<int> even = numbers.where((number)=> number.isEven).toList();***

**Set –** Тухайн цуглуулгад давхардаагүй өгөгдлийг хадгалдаг.

* **Contains –** Тухайн нэг утга цуглуулга дотор байгаа эсэхийг шалгадаг Boolean утга буцаадаг.
* **Add –** Оруулах утгаа параметрт нь дамжуулдаг.
* **Remove –** Устгах утгаа параметрт нь дамжуулдаг.
* **addAll –** Өөр шинэ цуглуулга нэмэхэд ашигладаг.
* **Clear –** Цуглуулгад байгаа бүх элемент үүдийг устгана.
* **Difference –** Эхний цуглуулгаас үүнд хандах зурвас илгээн параметрдээ дараагийн харьцуулах цуглуулгыг дамжуулаад шинэ set цуглуулга үүсгэдэг.Мөн final түлхүүрийг ашиглан тухайн цуглуулгыг хадгалах боломжтой.
* **elementAt() –** Тухайн индекс дээрх утгыг параметрээр авдаг.
* **Intersection –** 2 цуглуулгаас давхацсан өгөгдөл болон цуглуулгыг буцаадаг.

**Map –** Түлхүүр болон утгын хамт хосолсон утгыг хадгалдаг.Тухайн нэг элемент ийн утгад хандахдаа түлхүүрийн утгыг **[индекс]** байдлаар тодорхойлж өгдөг.

* **Keys –** Тухайн цуглуулгын түлхүүрүүдийг харах тохиолдолд ашиглана
* **Values -** Тухайн цуглуулгын утгуудыг харах тохиолдолд ашиглана
* **Length –** Тухайн цуглуулгын нийт хэмжээг хослолын нийт тоог буцаадаг.
* **NewItem - countryCapital['Japan'] = 'Tokio'** түлхүүрээрээ дамжуулан утга оноох байдлаар шинээр элемент нэмж болдог.Мөн тухайн түлхүүрт харгалзах утгыг шинэчлэх боломжтой байна.
* **keys.toList() –** Түлхүүрийн утгуудыг list руу хувиргах боломжтой.
* **values.toList() –**Утгуудыг list руу хувиргах боломжтой.
* **containsKey(‘key’) –** Түлхүүрийн утга байгаа эсэхийг шалгадаг. Boolean утга буцаадаг.
* **containsValue(‘value”) –** Тухайн утга байгаа эсэхийг шалгадаг. Boolean утга буцаадаг.
* **Clear –** Цуглуулгад байгаа бүх элемент үүдийг устгана.
* **Remove –** Түлхүүрийн утгыг өгөн тухайн хослолыг устгах боломжтой.
* **Loop –** for давталтын үед MapEntry түлхүүрийг ашиглан дараах байдлаар map цуглуулгаар гүйх боломжтой.  
  ***Жишээ1:******for(MapEntry mapEntry in map.entries)  
  Жишээ2: book.forEach((key,value)=> print('Key is $key and value is $value'));***
* **RemoveWhere -** *mathMarks.removeWhere((key, value) => value < 32);*

**File –** Файлтай ажиллахдаа File объектыг үүсгэн тухайн унших гэж буй файлыг байгуулагчд нь оруулж өгдөг.Тухайн файлтай холбоотой бүх зүйлсийг мэдэх боломжтой.

* **Read -** file.readAsStringSync();
* **Write - file.writeAsStringSync("Update new line!!");** Шинээр тухайн файлд бичдэг өмнөх текстүүдийг устгадаг.Харин mode – ийг append хийж өгснөөр өмнөх текстүүд хадгалагдана. **file.writeAsStringSync("\n this is new content.", mode: FileMode.append);CSV** файлд текст нэмэхдээ бичих үйлдэлдээ харгалзан утгуудаа онооно.
* **File.existsSync()** – Үүгээр тухайн файл байгаа эсэхийг шалгадаг.

**OOP –** Классыг хэрэгжүүлэхдээ тухайн классын гишүүн өгөгдөл **null** байж болно гэдэг утгаар төрлийн ард заавал **?** тэмдэг тавьдаг байх.

* Байгуулагчийг тодорхойлоогүй тохиолдолд гишүүн өгөгдлүүдэд обьектоор нь хандан утгыг нь оноож өгнө.
* Мөн байгуулагчийн авах параметрийн хувьд default утгыг оноогоод заавал авах ёсгүй байдлаар мөн тодорхойлж болдог.
* **Chair({this.name, this.color});** энэ нь тухайн объектыг үүсгэх үед гишүүн өгөгдлийг харгалзан авах утгын хамт байхаар тохируулдаг **Chair(name: "Chair1", color: "Red").**
* Дээрх жишээний хувьд анхны утга тодорхой бол объект үүсгэхдээ параметрүүдийг оруулах шаардлагагүй болно.**Table({this.name = "Table1", this.color = "White"});Table table = Table();**
* **Default constructor –** Тухайн классын аль нэг гишүүн өгөгдөлд утга оноосон байхаар тодорхойлж болно.

**Flutter project –** Шинээр project үүсгэхдээ тухайн фолдертоо орон cmd нээн ***flutter create project\_name*** коммандыг бичин шинэ төслөө үүсгэдэг.

**Тохируулга хийх - flutter doctor** энэ комманд таны системийг шалгаж, Flutter-ийн суулгалтын статусыг тайлагнана. Хэрэв танд ямар нэгэн хамааралтай зүйл дутуу байвал мэдэгдэнэ.

**Шинэ Төсөл Үүсгэх - flutter create my\_app** шинэ Flutter төсөл эхлүүлэхийн тулд дараах коммандыг ашиглана.

**Байгууллагын домэйн нэртэй - flutter create --org com.yourdomain appname** хэрэв өөрийн байгууллагын домэйн нэрийг төслөөсөө оруулахыг хүсвэл ингэж бичиж болно.

**Апп ажиллуулах** – **flutter run** flutter апп-аа анхдагч төхөөрөмж дээр ажиллуулна.

**Төхөөрөмжүүдийг Шалгах - flutter devices** Холбогдсон бүх төхөөрөмжүүдийг шалгахын тулд дараах коммандыг ашиглана.

**Android – д APK барих - flutter build apk**

**iOS-д IPA Барих - flutter build ios**

**Бүх Пакетуудыг Татах - flutter pub get** төслийнхөө бүх хамааралтай пакетуудыг татах.

**Android App Bundle Үүсгэх - flutter build appbundle** Google Play-д апп байрлуулахад зориулсан App Bundle үүсгэхийн тулд ашиглана.

**Хамгийн их хэрэглэгддэг Flutter виджетүүд**

* **Text:** Дэлгэц дээр текст харуулахад ашиглана.
* **Container**: Тэгш өнцөгт хайрцаг харуулахад ашиглана. Container виджет ашиглан **padding, margin, border, background color** гэх мэт стиль нэмэх боломжтой.
* **Row:** Виджетүүдийг хэвтээ байдлаар байрлуулна.
* **Column:** Виджетүүдийг босоо байдлаар байрлуулна.
* **Stack:** Виджетүүдийг давхарлан харуулахад ашиглана.

**Container –** Widget болон бусад өгөгдлүүдийг агуулах хайрцаг.**EdgeInsets.all(10)** дотор байгаа зүйл нийт талаасаа 10 хэмжээний зай авна.**offset** байршилийг, **blurRadius** нь бүдгэрүүлэх хэмжээг тус тус тодорхойлдог.**borderRadius.circular(10)** дугуйруулах хэмжээг илтгэнэ.**gradient: LinearGradient** нь өнгө холих боломжийг олгоно.

***Жишээ 1:*** ***decoration: BoxDecoration(***

***boxShadow: [***

***BoxShadow(***

***color: Colors.grey,***

***blurRadius: 5,***

***offset: Offset(100,100),***

***),***

***],***

***borderRadius: BorderRadius.circular(50),***

***gradient: LinearGradient(***

***colors: [***

***Colors.blue,***

***Colors.green,***

***],***

***),***

***),***

***Жишээ 2: Зураг оруулах үед ашиглана.***

***child: Image.network(***

***'https://avatars.githubusercontent.com/u/33576285?v=4',***

***),***

**Row –** Мөр байдлаар ашиглана.Horizontally

**Column –** Багана байдлаар ашиглана. vertically

**mainAxisAlignment –** Children – г хэвтээ чиглэлд хэрхэн байрлуулахыг заана.

**crossAxisAlignment -** Children – г босоо чиглэлд хэрхэн байрлуулахыг заана.

**Children -** Row дотор байрлах widget-үүдийг жагсаана.

|  |  |
| --- | --- |
|  |  |

|  |  |
| --- | --- |
|  |  |

**mainAxisAlignment.start –** Children - г зүүн талаас байрлуулна

**MainAxisAlignment.end -** Children - г баруун талаас байрлуулна

**MainAxisAlignment.center -** Children - г голд байрлуулна

**MainAxisAlignment.spaceBetween –** Children хооронд зайг тэнцүү өгнө, харин эхлэл ба төгсгөлд зай байхгүй.

**MainAxisAlignment.spaceAround -** Children хооронд зай тэнцүү, мөн эхлэл ба төгсгөлд тал орчим зай өгнө.

**MainAxisAlignment.spaceEvenly -** Children өмнө, хооронд, дараа нь тэнцүү зай өгнө.

**CrossAxisAlignment.stretch –** Children - г босоо чиглэлд дүүргэж сунгана (өргөн нь биш өндрөөр).

**CrossAxisAlignment.baseline -** Children - г baseline буюу текстийн суурийн шугамаар зэрэгцүүлэн байрлуулна (ихэвчлэн текст дээр хэрэгтэй).

**SingleChildScrollView -** Row дотор олон widget байвал, тэдгээрийг дэлгэцээр хөндлөн гүйлгэж харах боломжийг олгодог. **scrollDirection: Axis.horizontal**

**Expanded** – Тухайн **widget-д боломжит бүх зайг эзлүүлэх** зориулалттай. Ялангуяа Row дотор ашиглахад, тухайн widget бусад элементтэй зайг тэнцүү хувааж авдаг.**overflow** үүсэхгүй гэсэн үг.

**Flexible** – Expanded -тэй төстэй ч, **хэмжээг нь илүү уян хатан** (flexible) болгодог. Тухайн widget нь **дотроо багтах хэмжээтэй байна**, гэхдээ хэт их тэлэхгүй.

**CircleAvatar** - Flutter-ийн UI элемент бөгөөд **дугуй хэлбэртэй зураг эсвэл текст харуулахад** ашиглагддаг.**raduis, backroundImage, backroundColor, child** зэрэг элементүүдийг агуулдаг.

**Row** – TextField ашиглан хэрэглэгчийн оролтын утга авах боломжийг олгоно.

**Жишээ нь: const Row(**

**children: <Widget>[**

**Text('Username: '),**

**Expanded(child: TextField()),**

**],**

**),**

**Stack –** Дэлгэцэнд харагдах элементүүдийг давхарлан харуулдаг.

**flutter run -t lib/main\_feature1.dart –** Файлыг сонгож ажиллуулах боломжтой.

**Employee(this.name, this.age, [this.subject = "N/A", this.salary=0]); -** Энэ байгуулагч нь тухайн объектыг үүсгэхэд сүүлийн 2 шинжийн утгыг заавал объект үүсгэхдээ дамжуулах шаардлагагүйг илэрхийлдэг.**Employee employee = Employee("John", 30);**

**Chair({this.name, this.color}); -** Байгуулагч үүсгэхдээ заавал нэрийн хамт утгуудыг харглазан өгдөг.**Chair chair = Chair(name: "Chair1", color: "Red");** Мөн утгыг оноохдоо дарааллын хувьд ялгаатай байдлаар мөн өгч болдог. **Chair(color: "Ногоон", name: "Савх2");** заавал дамжуулах утга өгөхгүй байгуулагч үүсгэж болно гэхдээ тухайн шинжийн утга null болно.Мөн анхны утга оноосон бол мөн байгуулагчид шинжийн утгыг дамжуулах шаардлагагүй болно.

**Person({required this.name, this.age}); -** name шинжид заавал утга оноох ёстойг илтгэнэ. **Person person1 = Person(name: "Алимаа");**

**Default Constructor -** Default конструктор нь параметрүүдгүй байна. Default конструкторыг классын нэр болон хаалт () ашиглан зарладаг.   
**Laptop() {**

**print("This is a default constructor");**

**}**

**Student(this.name, this.age, this.rollNumber); -** Энэ байгуулагчын хувьд параметрүүдийн авах утгын дараалал яг оноосон дагуу байх ёстой.

**Named Constructor –** Ижил нэртэй олон байгуулагч үүсгэхийн тулд ашигладаг.**Class.namedConstructor()** **Class.namedConstructor2()** гэсэн байдлаар тодорхойлдог.

**fromJsonString constructor –** Объектыг байгуулахдаа json төрлийн string болон map төрдийн collection – г дамжуулан түүнийхээ тусламжтайгаар байгуулагчыг үүсгэж болдог.

***Жишээ 1:***

***Person.fromJsonString(String jsonString) {***

***Map<String, dynamic> json = jsonDecode(jsonString);***

***name = json['name'];***

***age = json['age'];***

***}  
String jsonString1 = '{"name": "Bishworaj", "age": 25}';  
Person p1 = Person.fromJsonString(jsonString1);***

***Жишээ 2 :***

***Person.fromJson(Map<String, dynamic> json) {***

***name = json['name']; age = json['age'];***

***}***

***Map<String, dynamic> personMap = {***

***'name': 'Сараа',***

***'age': 28,***

***};***

***Person anotherPersonFromMap = Person.fromJson(personMap);***

**Const Contructor -** Тогтмол Конструктор нь утгыг нь өөрчлөх боломжгүй объект үүсгэхэд ашиглагддаг. Ингэхдээ классын бүх шинж final байх, ямар нэгэн биегүй байна, const түлхүүрийг ашиглан анхны утгыг оноодог. Dart нь ижил утгатай бүх тогтмол объектод санах ойд нэг л хувийг үүсгэж, бүх хувьсагчийг тэр хувь руу заадаг.

***Жишээ :***

***const Point(this.x, this.y);***

***Point p1 = const Point(1, 2);***

***print("The p1 hash code is: ${p1.hashCode}");***

***Point p2 = const Point(1, 2);***

***print("The p2 hash code is: ${p2.hashCode}");***

***Point p3 = Point(2, 2);***

***print("The p3 hash code is: ${p3.hashCode}");***

***Point p4 = Point(2, 2);***

***print("The p4 hash code is: ${p4.hashCode}");***

***Тайлбар : p1, p2 нь const байдлаар үүсгэгдсэн ба шинжийн утгууд ижил учир 1 hashcode – той харин p3, p4 ижил утгагтай ч hashcode өөр өөр байна.Hashcode – нь тухайн объектыг илэрхийлэхийн тулд ашиглагддаг тоон утга.***

**Encapsulation -** шинж чанарыг доогуур зураас (\_) ашиглан **private** гэж зарлах.**var \_name; int? \_id;** байдлаар тодорхойлж өгдөг ба шууд хандах боломжгүй болдог.Хандахын тулд заавал get set функцууд ашиглана.

***Жишээ :***

***String get name => \_name;***

***int get age => \_age;***

***String get gender => \_gender;***

***// Map Getter***

***Map<String, dynamic> get map {***

***return {"name": \_name, "age": \_age, "gender": \_gender};***

***}***

***Map төрлийн get – ийг мөн хэрэгжүүлж болдог.***

**Final –** түлхүүр нь тухайн объектыг үүсгэсний дараа утгыг нь өөрчлөх боломжгүй гэдгийг тодорхойлдог.

**Inheritance –** Хүүхэд классын объекыг үүсгэхэд дагаад эцэг классын байгуулагч дуудагддаг.

**1.Child non parameter –** Мөн эцэг классын байгуулагч нь параметртэй бол хүүхэд класс нь түүндээ заавал дамжуулдаг.**super** түлхүүр үгийн тусламжтайгаар эцэг классын шинж чанаруудын утгыг тохируулахад ашигладаг.**MacBook(String name, String color) : super(name, color)**

**2.Child class parameter –** Хүүхэд класс өөрийн шинжтэй бол түүндээ утгыг оноохоос гадна эцэг классын байгуулагчыг дуудахдаа дараах байдлаар хэрэгжүүлдэг.**Student(String name, int age, this.rollNumber) : super(name, age)**

**3.Named parameter –** Параметрын нэрийн хамт шинжийн утгыг тодорхойлдог бол дараах байдлаар тодорхойлно.  
**Laptop({String name, String color})   
MacBook({String name, String color}) : super(name: name, color: color)  
var macbook = MacBook(name: "MacBook Pro", color: "Silver");**

**Polymorphism –** Эцэг классын үйлдлийг хүүхэд класс өөрийхөөрөө хийх.Хэрвээ хүүхэд классд тухайн үйлдлийг хэрэгжүүлэх бол @override түлхүүрийг ашигладаг.Мөн хүүхэд классд дахин тодорхойлоогүй бол эцэг классын үйлдлийг дууддаг.  
**Static –** Классыг хэрэгжүүлэхэд ганц л удаа анхны утгаа авдаг.Үүссэн объект бүрийн хувьд нэг ижил утгыг ашиглах хэрэгцээ гарсан тохиолдолд ашигладаг.Объектыг үүсгэхгүйгээр шууд хандах боломжтой.

**Enum –** Тогтмол утгуудыг агуулна. Enum нь конструктортой байж (заавал const байх ёстой), гишүүдийг үүсгэх үед анхны утгыг оноох боломжтой. Enum-ийн гишүүн бүрийг үүсгэх үед энэ конструктор дуудагдаж, дамжуулсан String утгыг text шинж чанарт оноодог.

***Жишээ :***

***enum CompanyType {***

***soleProprietorship("Sole Proprietorship"),***

***partnership("Partnership"),***

***corporation("Corporation"),***

***limitedLiabilityCompany("Limited Liability Company");***

***// Members***

***final String text;***

***const CompanyType(this.text);***

***}***

***void main() {***

***CompanyType soleProprietorship = CompanyType.soleProprietorship;***

***print(soleProprietorship.text);***

***}***

**Abstract class –** Объект үүсдэггүй өөртөө шинж болон үйлдлүүдийг агуулсан классыг хэлнэ.Үйлдлийг тодорхойлохдоо яг яаж юу хийхийг нь тодорхойлдоггүй зарладаг.Удамшсан классуудад тухайн үйлдлийг @override хийж дахин тодорхойлдог.extends түлхүүрийг ашиглан хэрэгжүүлдэг.

**Interface –** Интрерфэйсд тодорхойлогдсон бүх шинж чанар аргуудыг тодорхойлох ёстой.Хэрэгжүүлэхдээ implements түлхүүр үгийг ашигладаг.Тухайн класс ямар чадвартай ямар үйлдэлтэй байхыг тодорхойлдог.Бүх классын хувьд interface байж болдог ба interface – ийн нөхцөл буюу дан үйлдлүүдийг агуулсан байх ёстой.Гэхдээ мөн эсрэгээрээ abstract классын interface хэлбэрээр ашиглах тохиолдолд шинжтэй байж болно.Олон классуудаас interface – ийг хэрэгжүүлж мөн болдог.

**Mixin –** mixin түлхүүр үг, дараа нь нэр гэх байдлаар тодорхойлдог.Үүнтэй ажиллахдаа mixin, on, with гэсэн 3 түлхүүр үгийг ашигладаг.Объект үүсдэггүй олон классын хооронд кодыг хуваалцахад ашигладаг. конструкторгүй бөгөөд өргөжүүлж болдоггүй.Нэг классд олон mixin-ийг ашиглах боломжтой.

**Жишээ нь:**

**mixin CanFly {**

**void fly() {**

**print('I can fly');**

**}**

**}**

**mixin CanWalk {**

**void walk() {**

**print('I can walk');**

**}**

**}**

**class Bird with CanFly, CanWalk {**

**}**

**class Human with CanWalk {**

**}**

**void main() {**

**var bird = Bird();**

**bird.fly();**

**bird.walk();**

**var human = Human();**

**human.walk();**

**}**

**ON –** Тухайн mixin – д өөр классын шинж агуулагдах шаардлагатай бол on түлхүүрийг ашиглаад хэрэгжүүлдэг.Тухайн mixin дотроо on хийсэн классын үйлдлийг @override хийж хэрэгжүүлнэ.

**Factory constructor –** Класс эсвэл дэд классын байгуулагчыг мөн private байгуулагчыг мөн буцаах боломжтой.Үүн дотор this түлхүүрийг ашигладаггүй.Мөн классын дотоод гишүүдэд хандах боломжгүй ба нэртэй болон нэргүй байж болдог.

***Жишээ нь:***

***// enum ShapeType***

***enum ShapeType { circle, rectangle }***

***// abstract class Shape***

***abstract class Shape {***

***// factory constructor***

***factory Shape(ShapeType type) {***

***switch (type) {***

***case ShapeType.circle:***

***return Circle();***

***case ShapeType.rectangle:***

***return Rectangle();***

***default:***

***throw 'Invalid shape type';***

***}***

***}***

***// method***

***void draw();***

***}***

***class Circle implements Shape {***

***// implement draw method***

***@override***

***void draw() {***

***print('Drawing circle');***

***}***

***}***

***class Rectangle implements Shape {***

***// implement draw method***

***@override***

***void draw() {***

***print('Drawing rectangle');***

***}***

***}***

***void main() {***

***// create Shape object***

***Shape shape = Shape(ShapeType.circle);***

***Shape shape2 = Shape(ShapeType.rectangle);***

***shape.draw();***

***shape2.draw();***

***}***

**Тайлбар: Ямар дүрсээс хамаараад тухайн дүрсийн байгуулагчыг буцаадаг байхаар хэрэгжүүлэх дасгал илүү тохиромжтой.**

**Нэмэлт:**  
**Person.\_internal(this.name); энэ нь нэрлэгдсэн private төрлийн байгуулагч**

**Generic –** Өөр өөр төрлийн өгөгдөлтэй ажиллах чадвартай класс эсвэл функцыг үүсгэх арга юм.

***Жишээ нь:*** ***// Using Generics***

***class Data<T> {***

***T data;***

***Data(this.data);***

***}***

***void main() {***

***// create an object of type int and double***

***Data<int> intData = Data<int>(10);***

***Data<double> doubleData = Data<double>(10.5);***

***// print the data***

***print("IntData: ${intData.data}");***

***print("DoubleData: ${doubleData.data}");***

***}***

**Тайлбар:** Классын төрөл нь ямарч байж болхоор тодорхойлж ашиглахын тулд generic – г ашигладаг.Төрлийн нэрийн оронд дурын нэр өгч болдог.

***Жишээ :***

***T genericMethod<T, U>(T value1, U value2) {***

***return value1;***

***}***

***void main() {***

***// call the generic method***

***print(genericMethod<int, String>(10, "Hello"));***

***print(genericMethod<String, int>("Hello", 10));***

***}***

**Тайлбар:** Функцын хувьд мөн авах буцаах утгын өгөгдлийн төрөл тодорхойгүй байдлаар ашиглаж болно.

**Өгөгдлийн төрөл хязгаарлах –** Тухайн классын хувьд нийт авах утгыг хязгаарлахын тулд ашигладаг.

***Жишээ :// Define generic class with bounded type***

***class Data<T extends num> {***

***T data;***

***Data(this.data);***

***}***

***void main() {***

***// create an object of type int and double***

***Data<int> intData = Data<int>(10);***

***Data<double> doubleData = Data<double>(10.5);***

***// print the data***

***print("IntData: ${intData.data}");***

***print("DoubleData: ${doubleData.data}");***

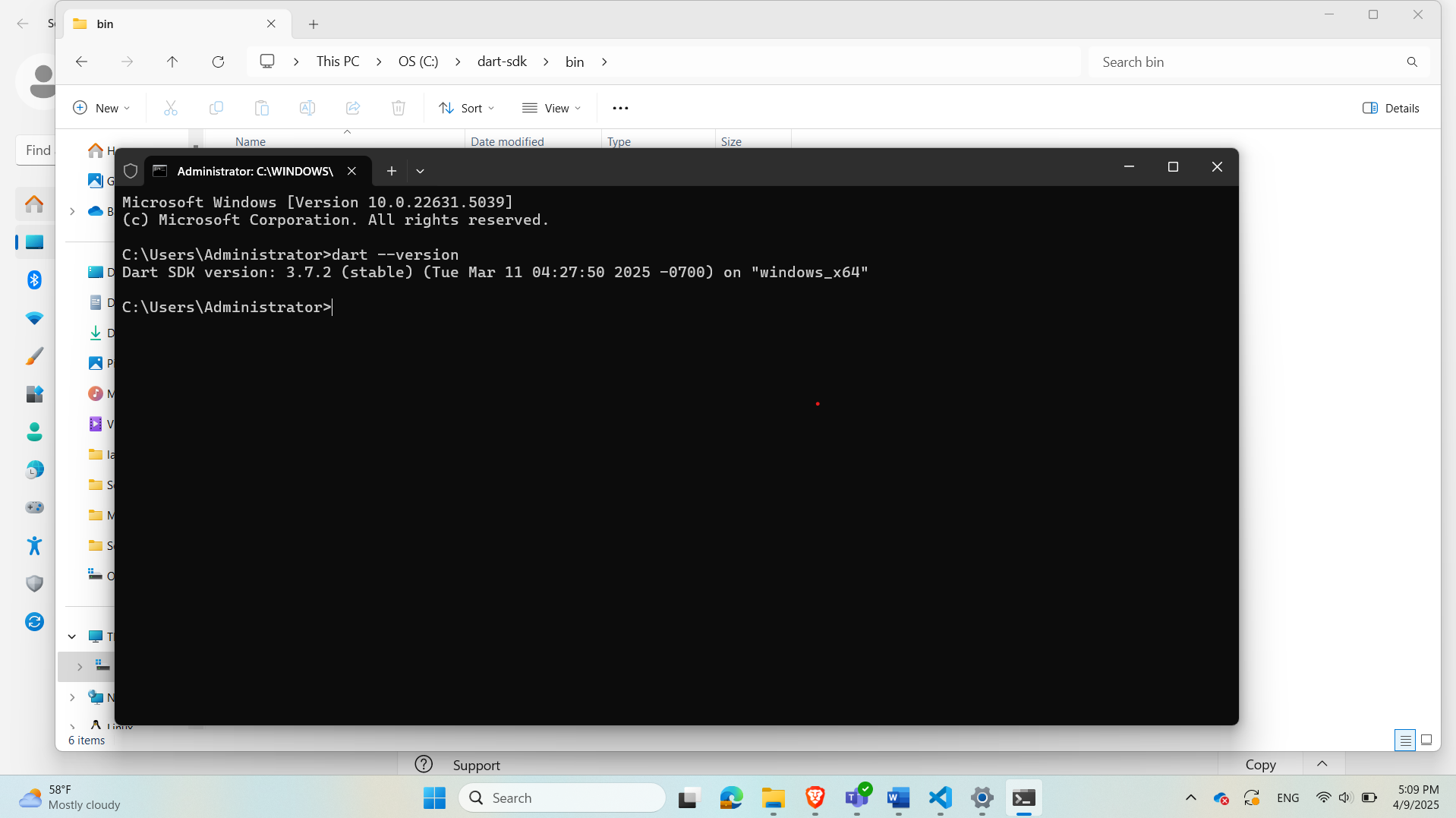
***// Not Possible***

***// Data<String> stringData = Data<String>("Hello");***

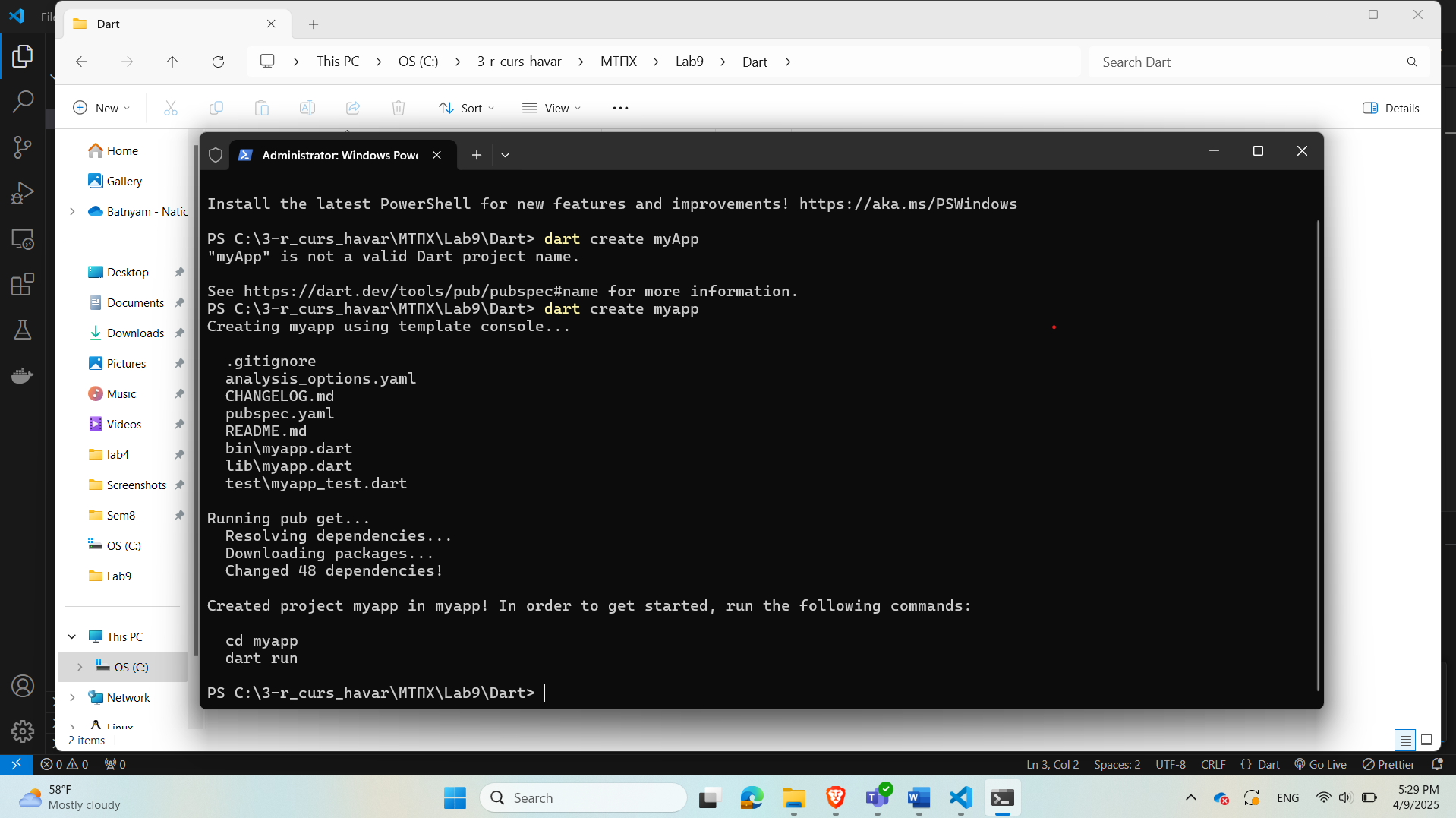
***}***

**Нэмэлт: num гэдэг нь Dart хэлний суурь абстракт класс бөгөөд бүх тоон төрлүүдийн эцэг анги юм.**

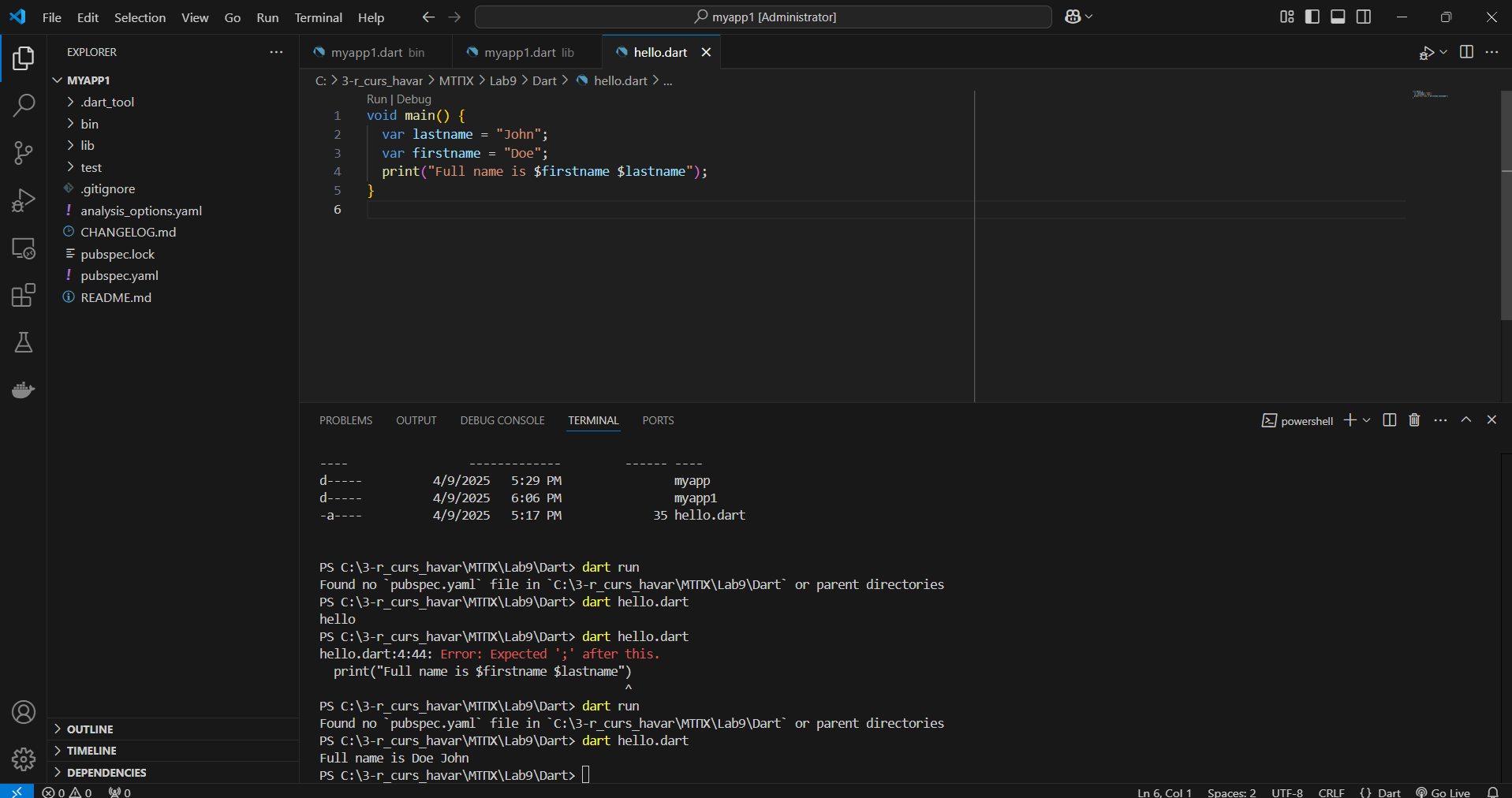
3.Хэрэгжүүлэлт



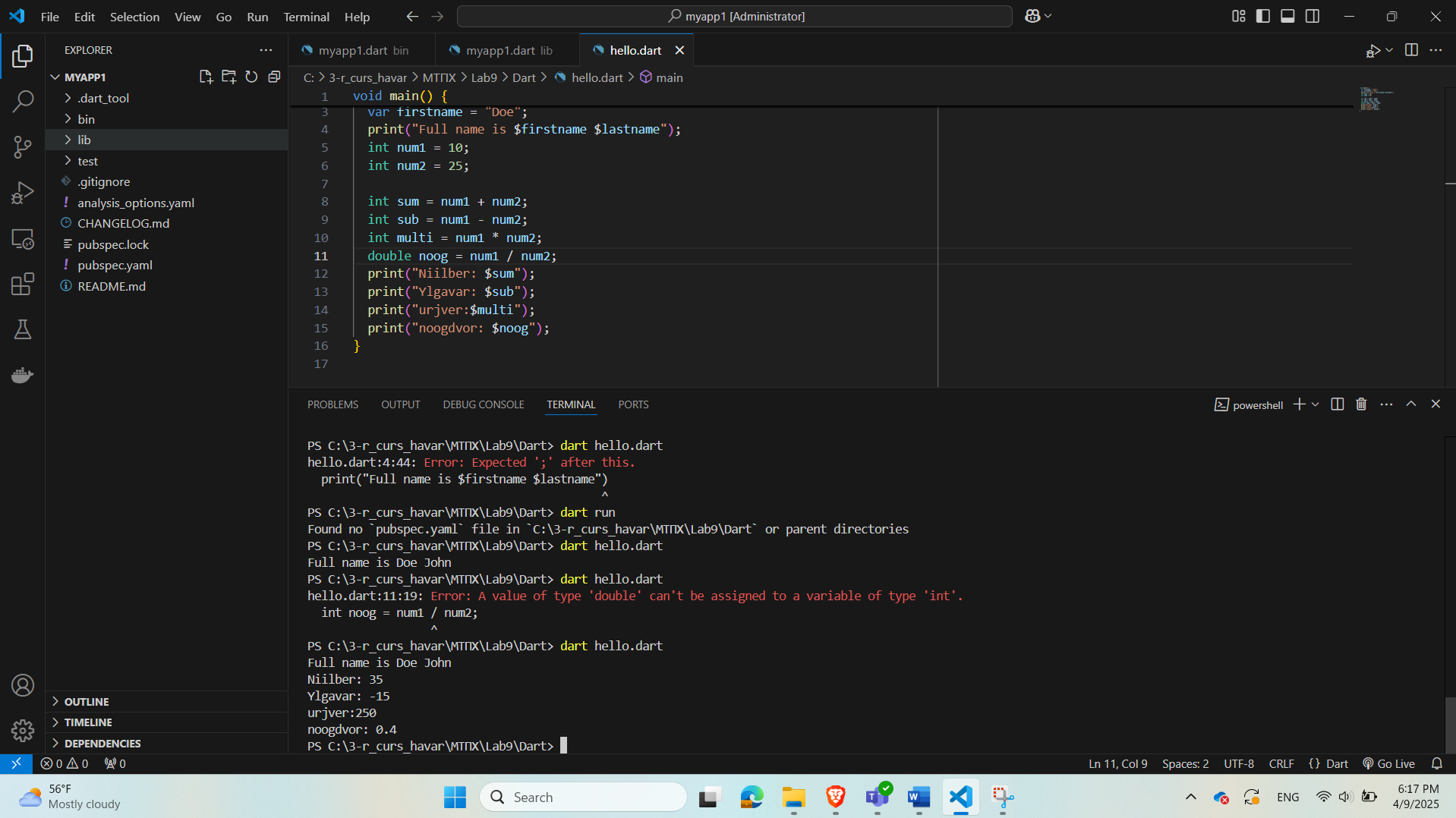
***Зураг 1*** *Dart амжилттай суусан*

**

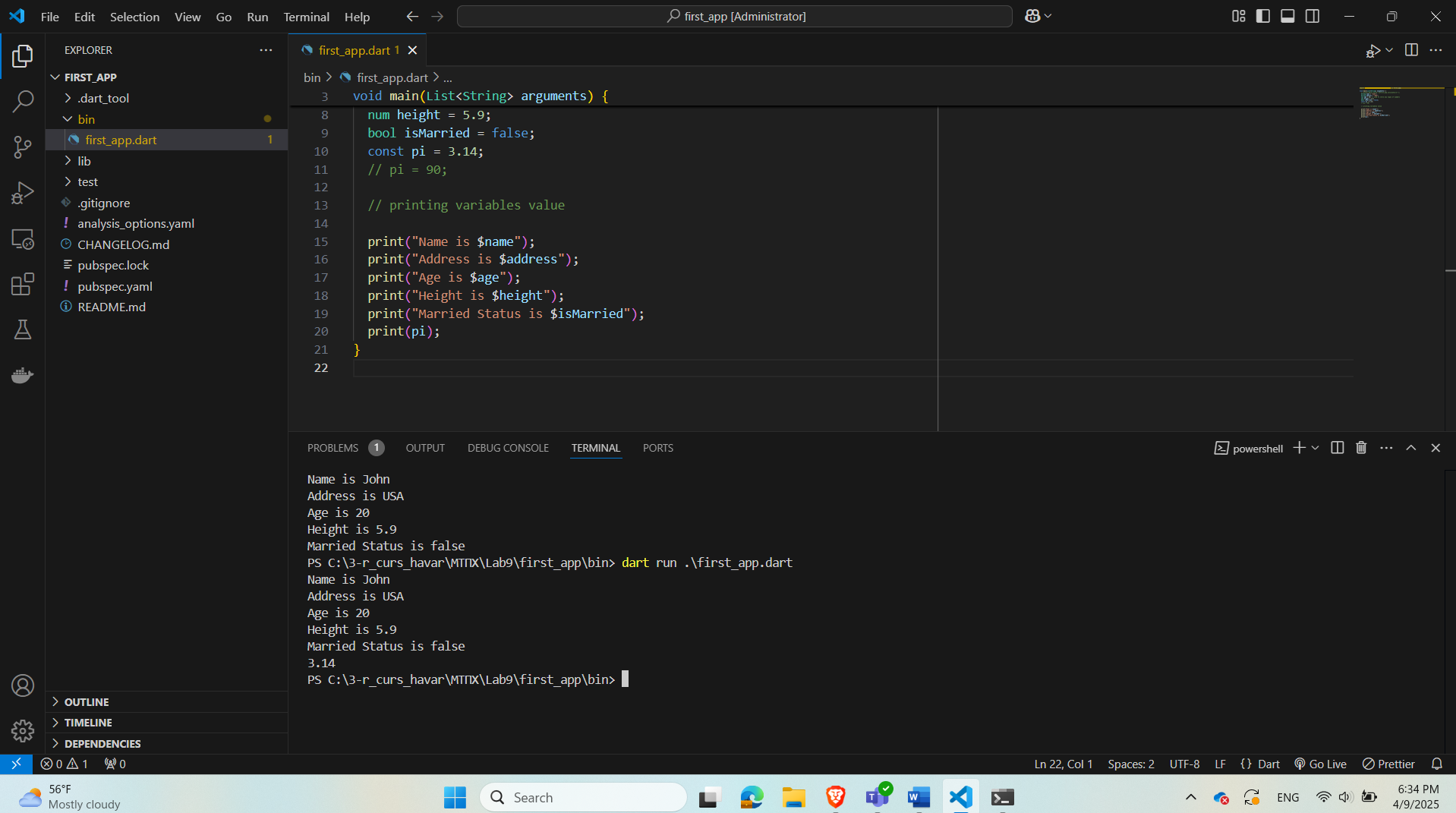
***Зураг2*** *Эхний ажиллаа амжилттай үүсгэсэн*

**

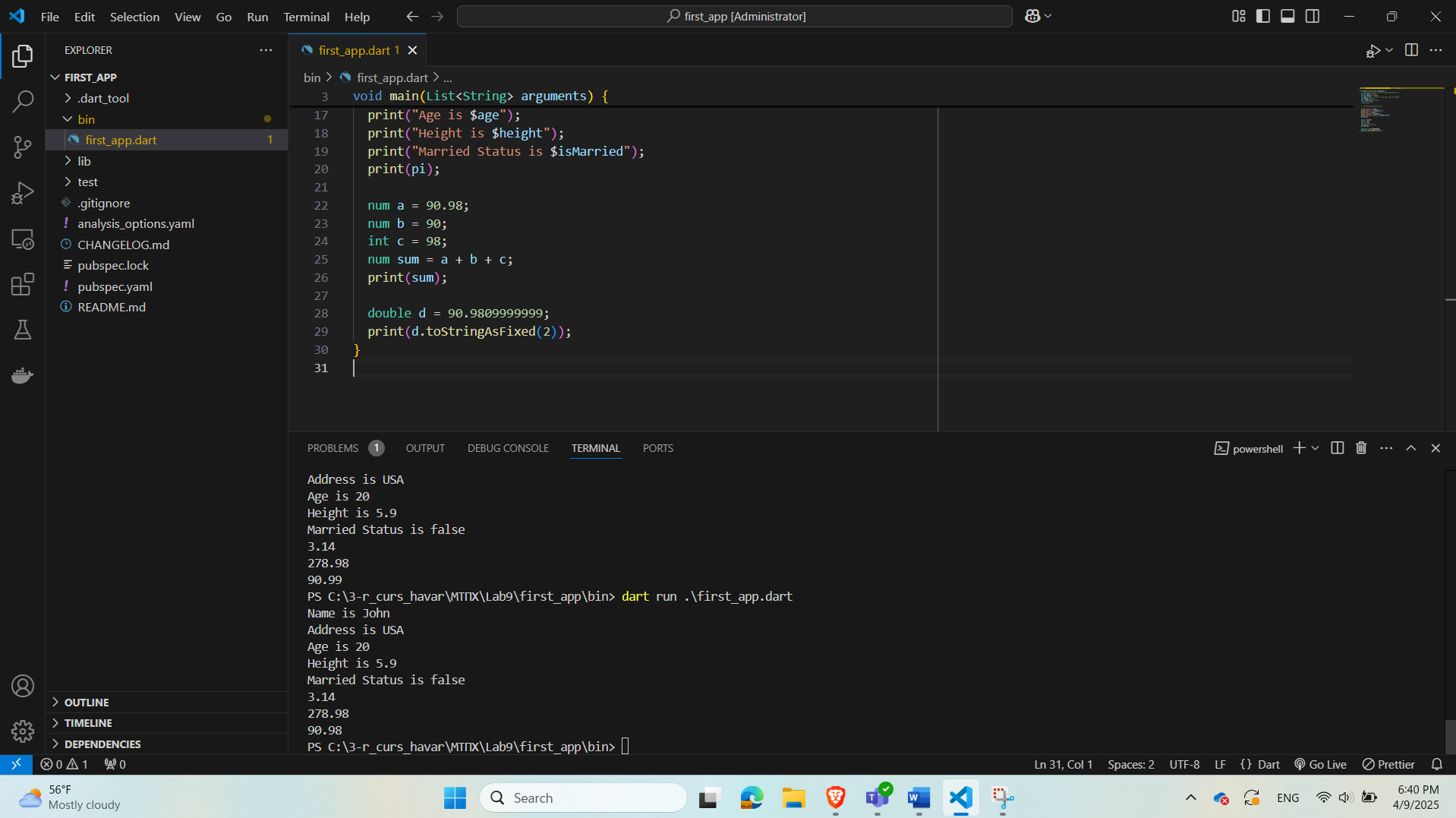
***Зураг3*** *Энгийн хэвлэх жишээг ажиллуулсан*

**

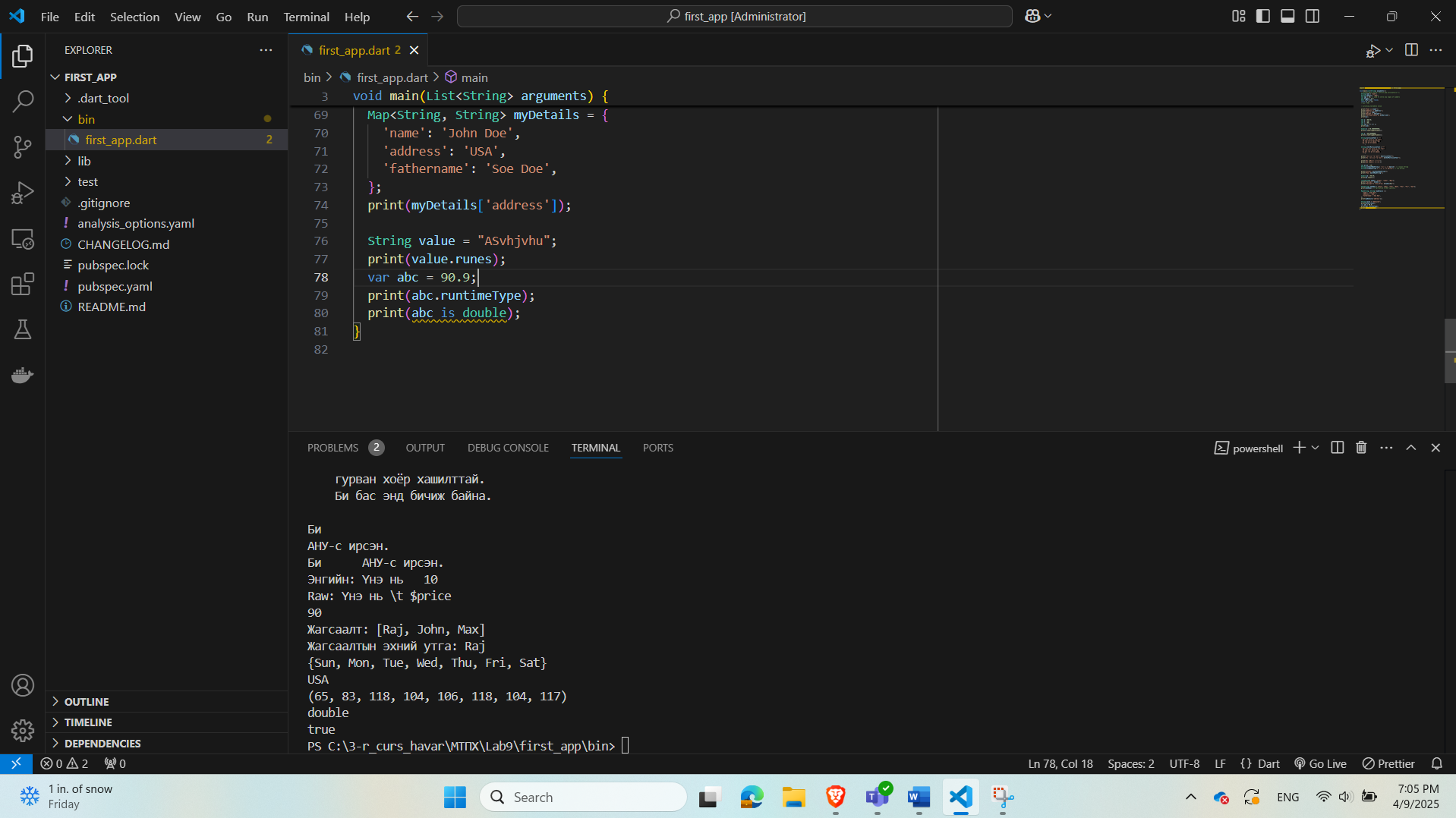
***Зураг4*** *Тоон дээр үйлдэл хийх туршилт*

**

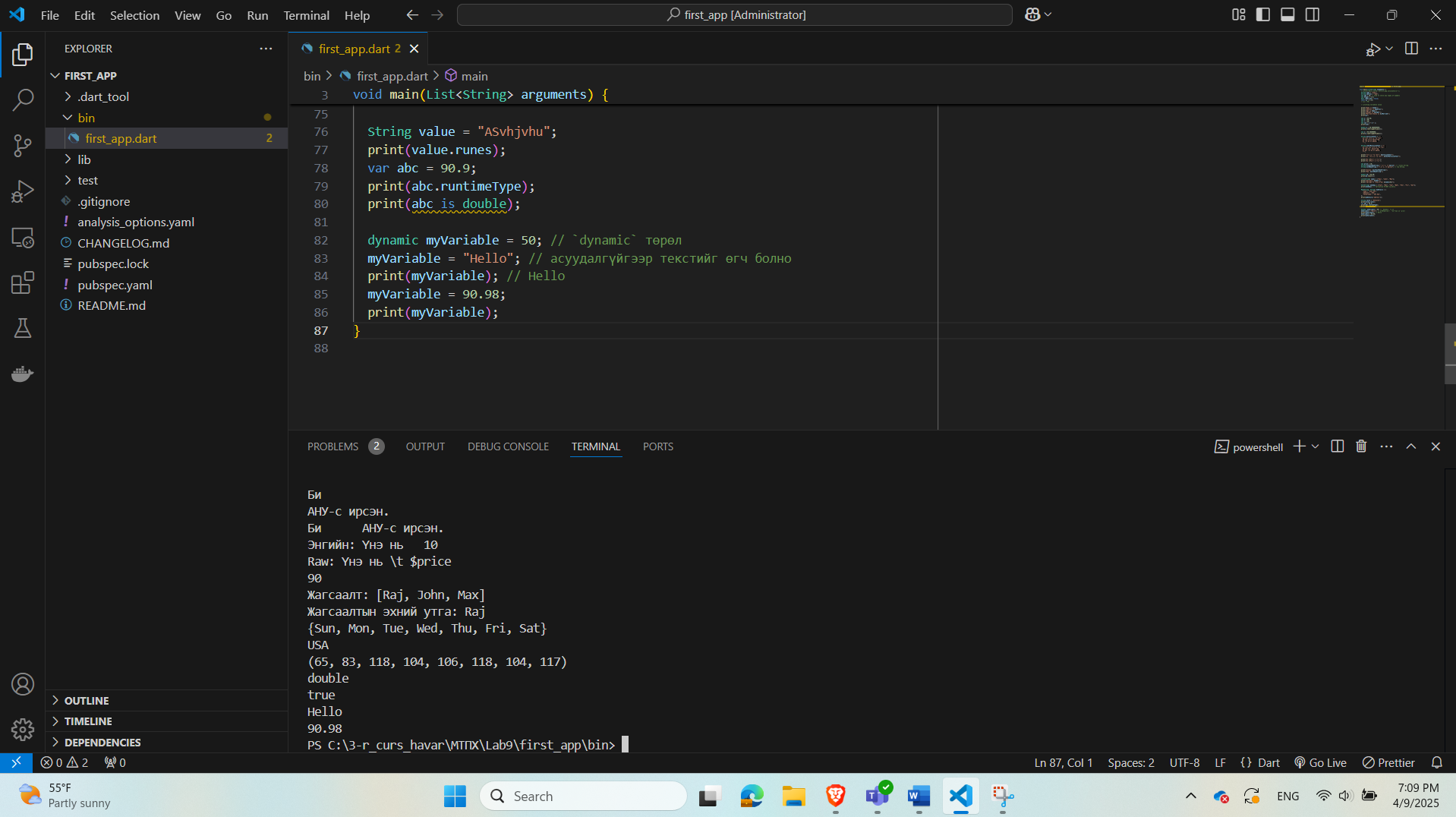
***Зураг5*** *Тогтмол утгыг өөрчлөх боломжгүй*

**

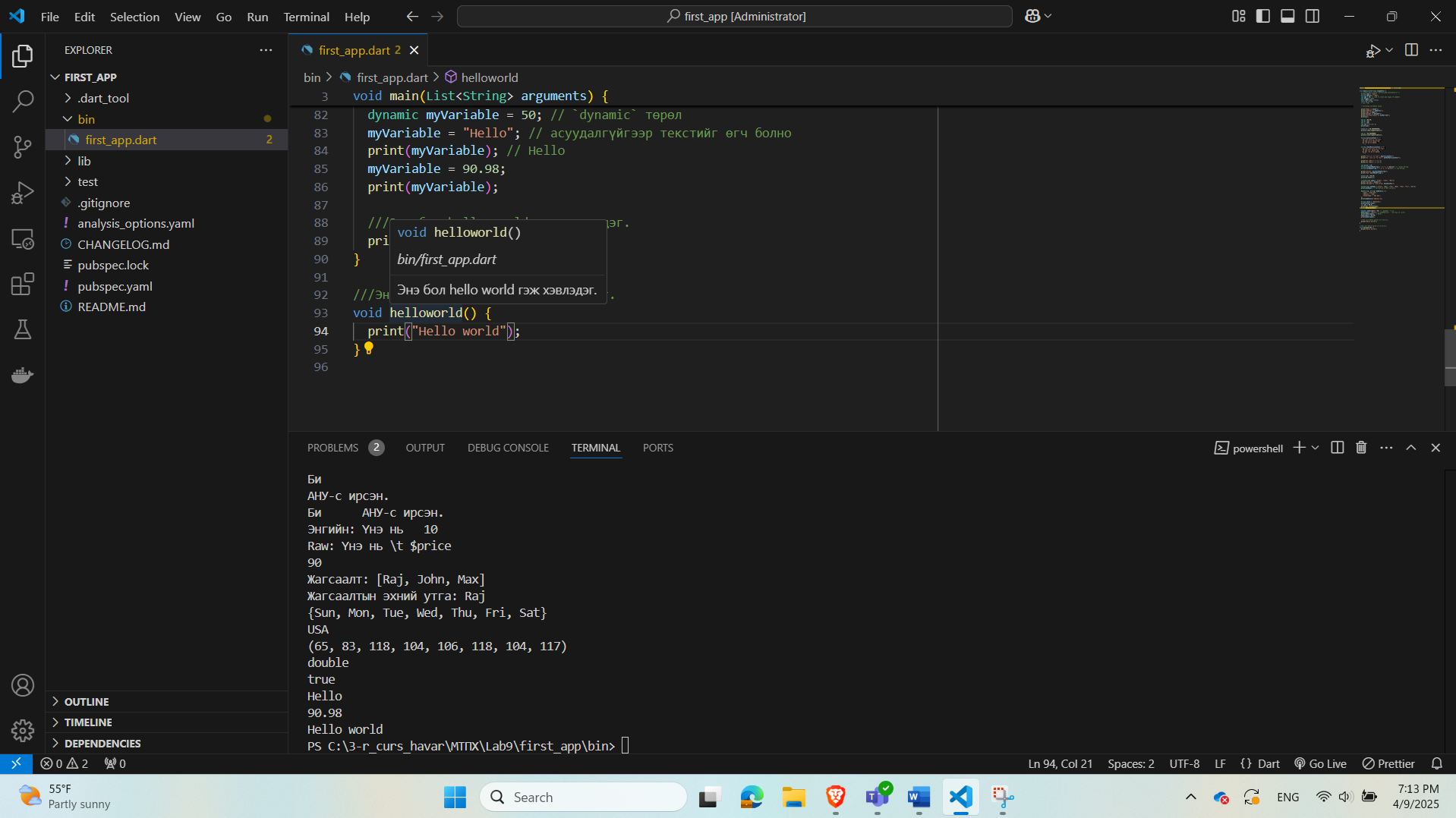
***Зураг6*** *toStringAsFixed() ашиглан num болон double өгөгдлийн төрлийг бүхэлдэж болдог*

**

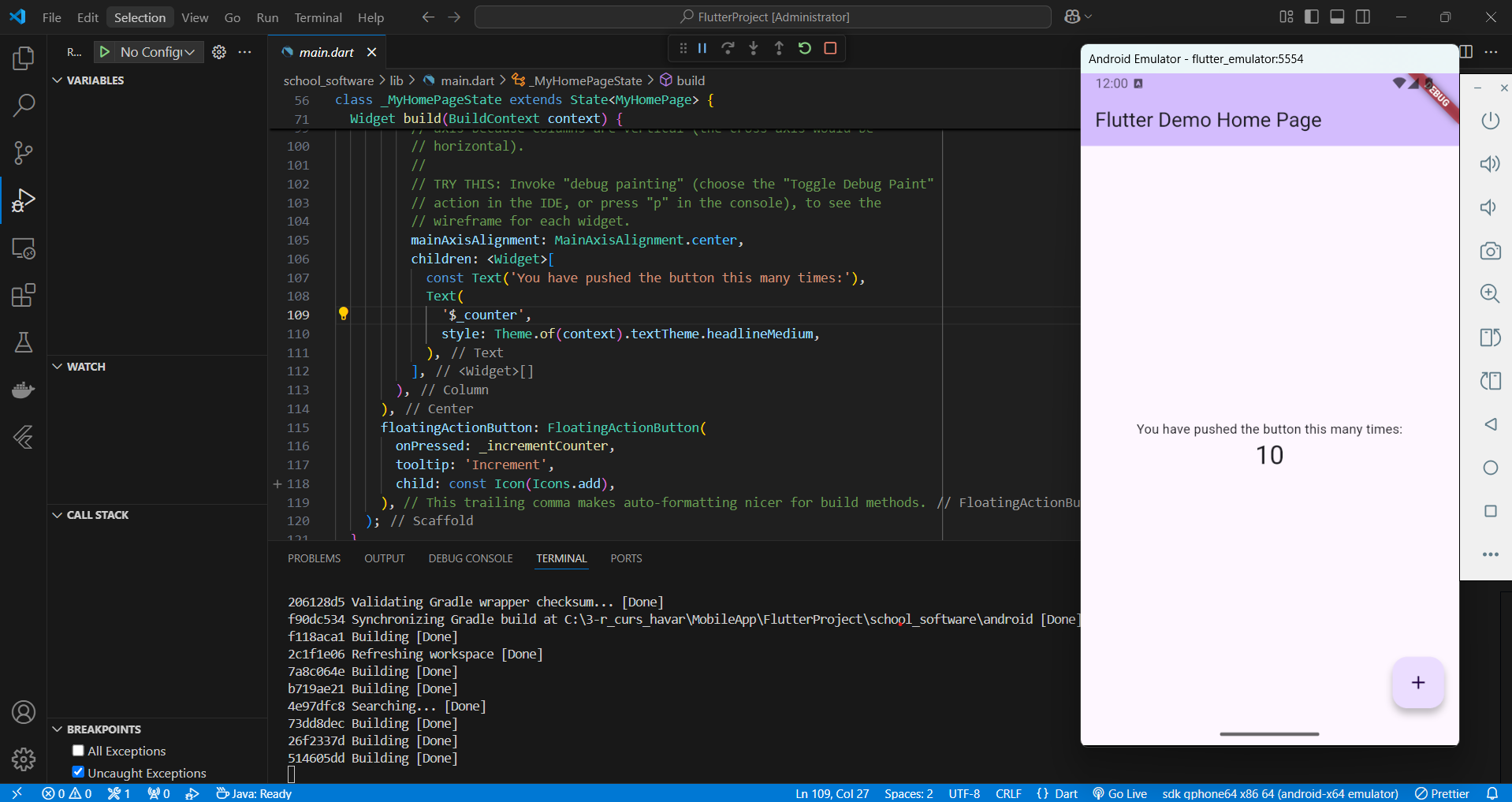
***Зураг7*** *runtimeType ашигласан жишээ*

**

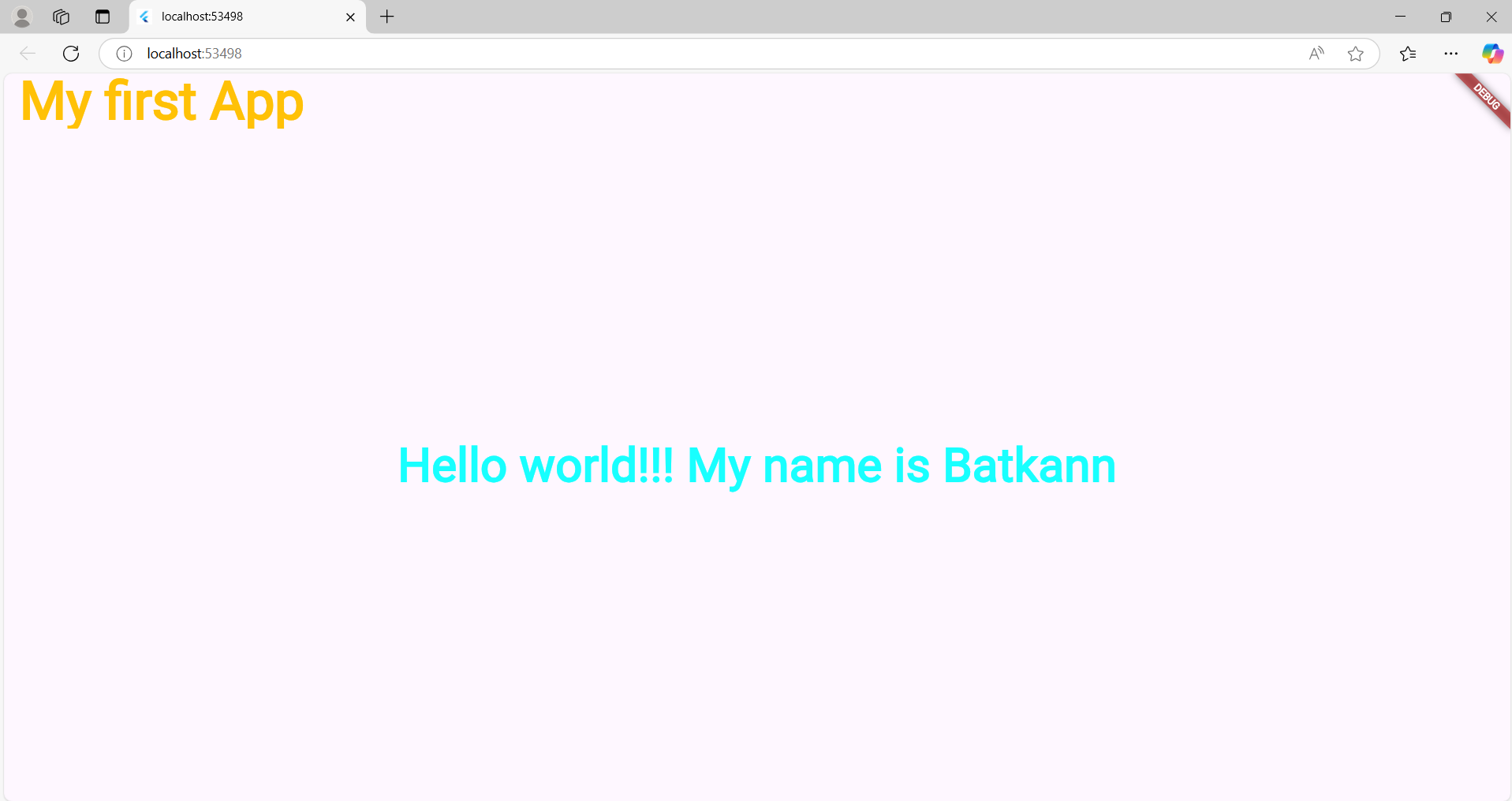
***Зураг8*** *Динамик өгөгдлийн төрөл ашигласан*

**

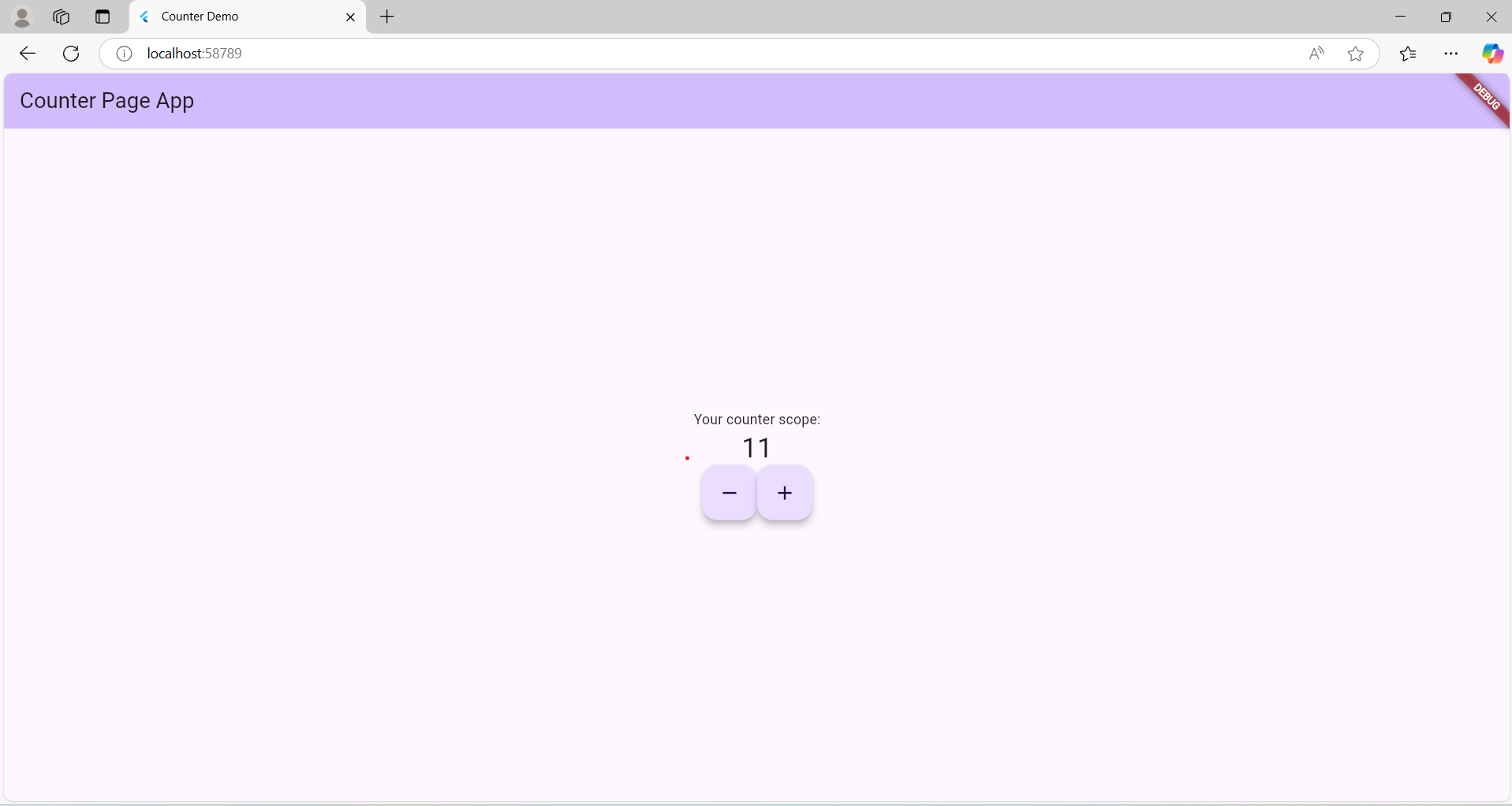
***Зураг9*** *Documentation comment хэрэгжүүлсэн*

**

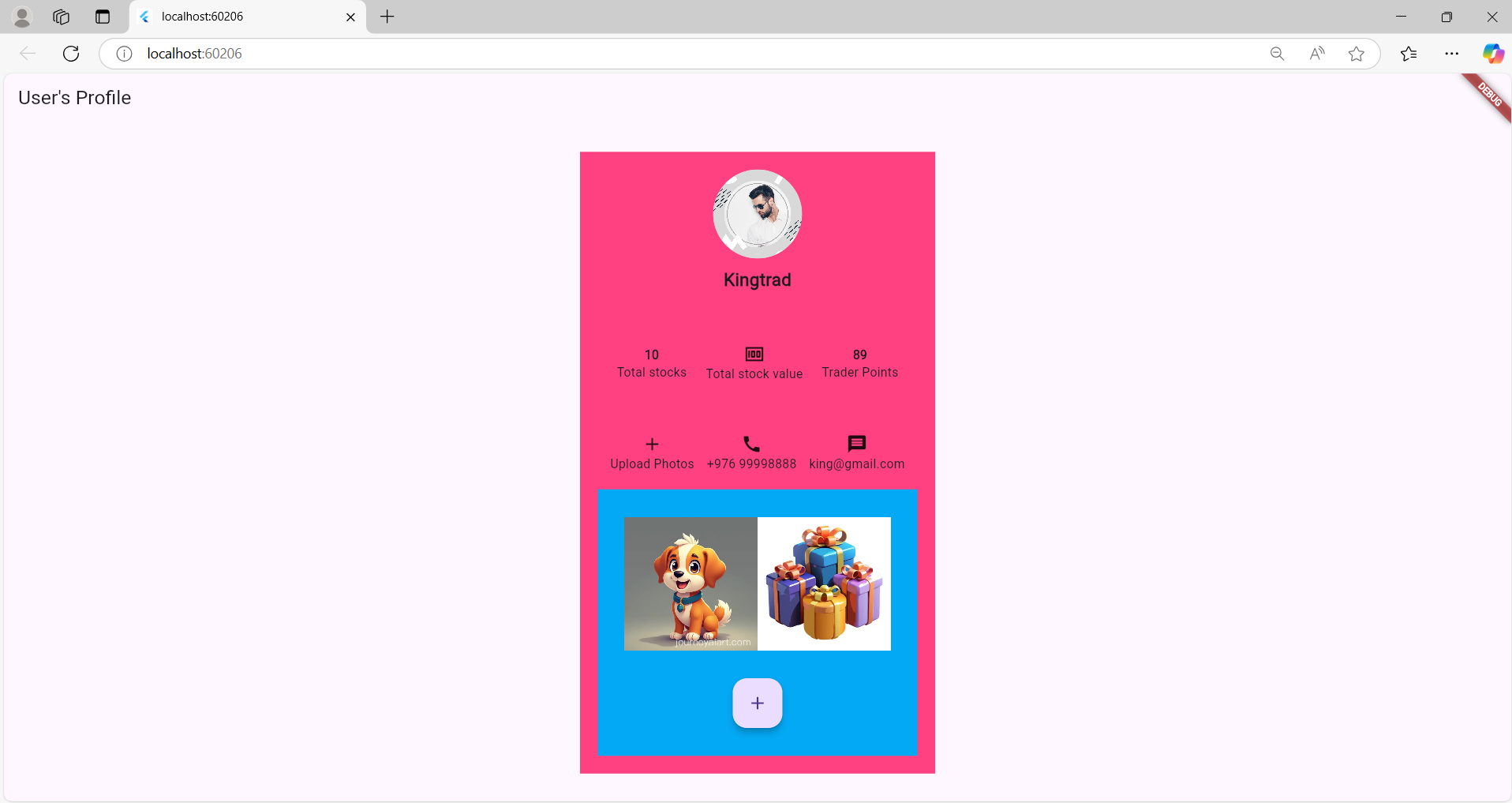
***Зураг10*** *Анхны flutter апп хэрэгжүүлсэн*

**

***Зураг11*** *Анхны flutter апп хэрэгжүүлсэн*

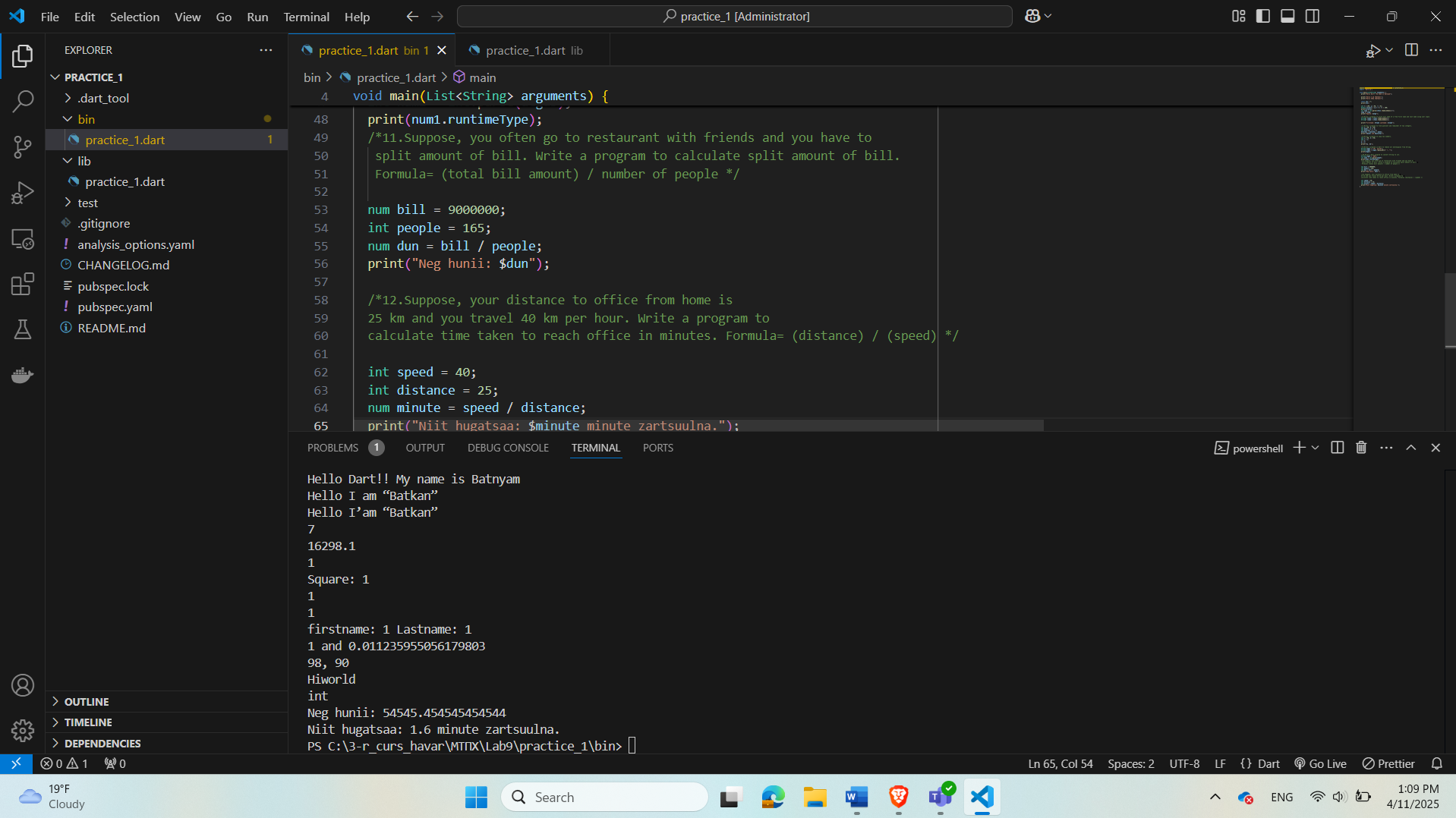
**

***Зураг12*** *Counter апп хэрэгжүүлсэн*

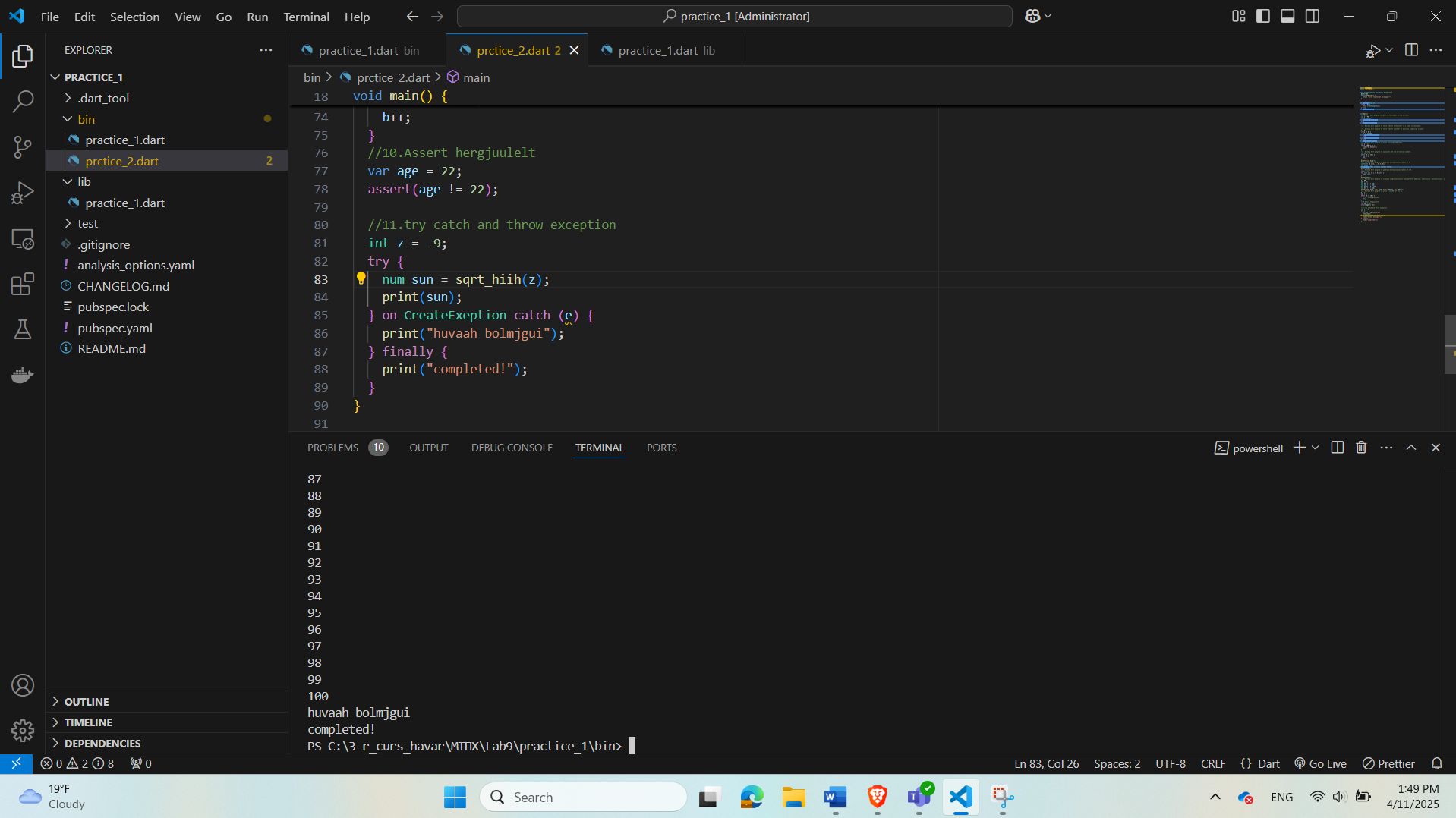
**

***Зураг 13*** *Profile дасгалыг хийж гүйцэтгэсэн*

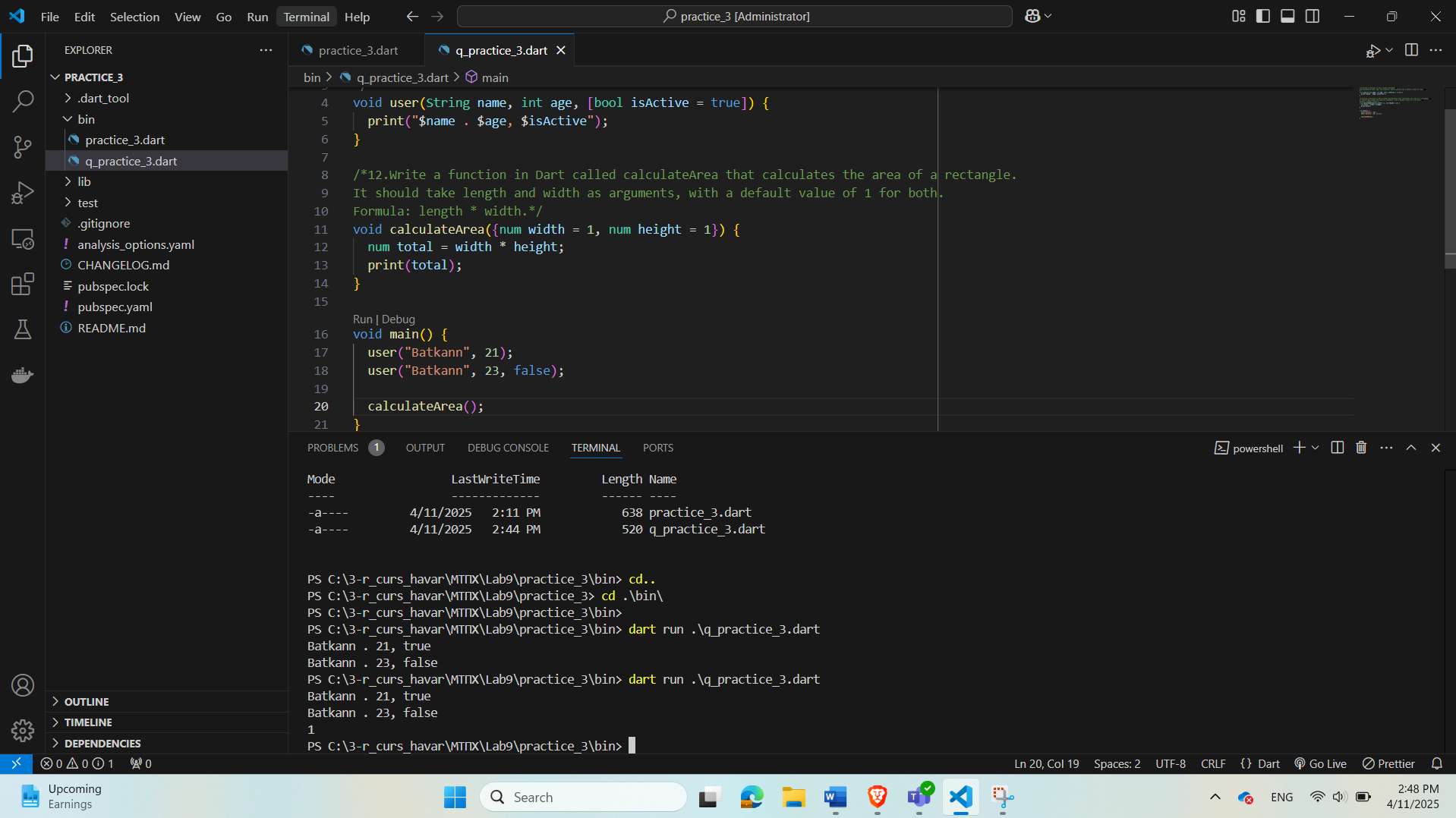
4. Бүлгийн дасгал

**

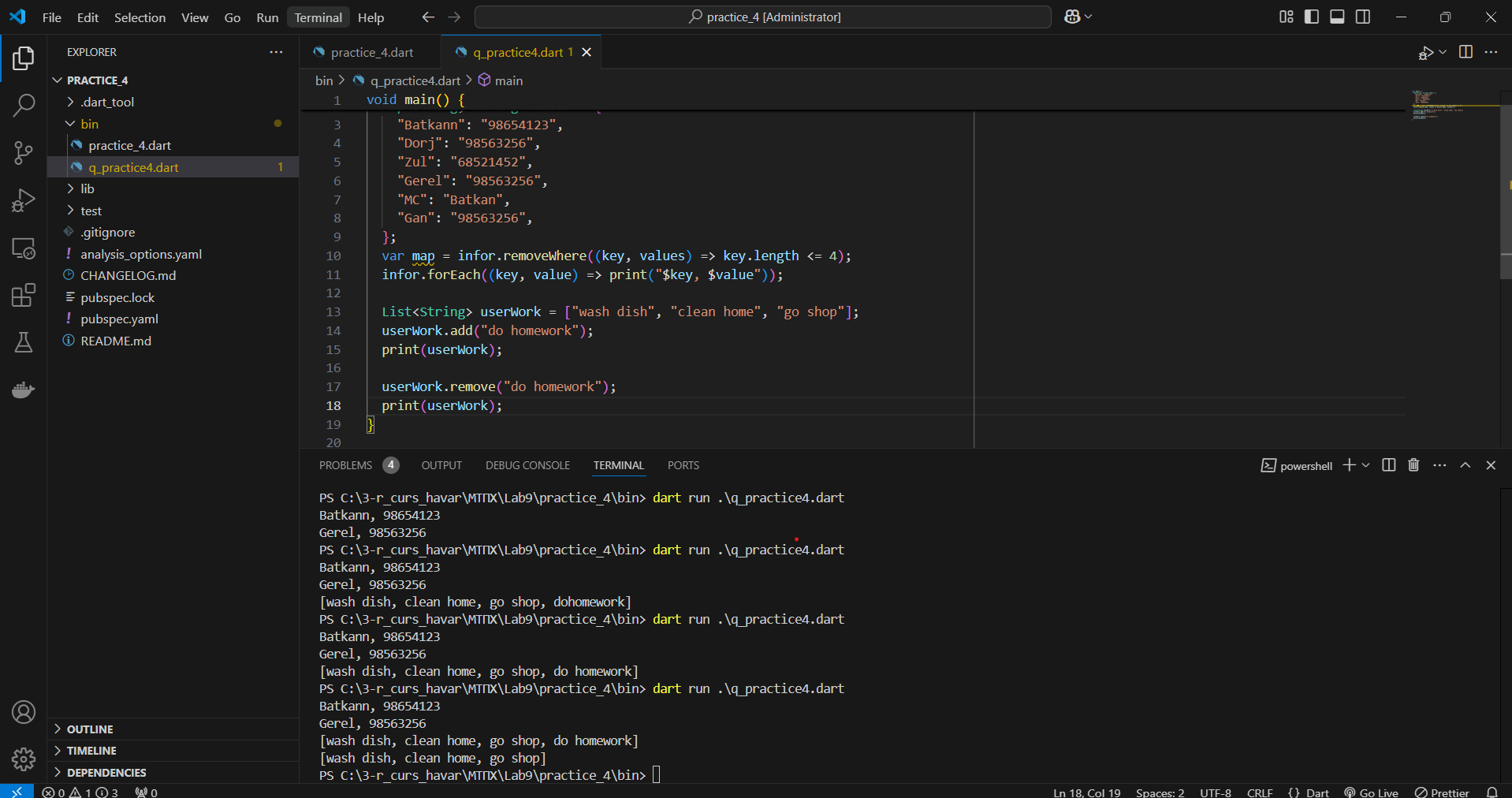
***Зураг 1*** *Эхний бүлгийн дасгалуудыг хийж гүйцэтгэсэн*



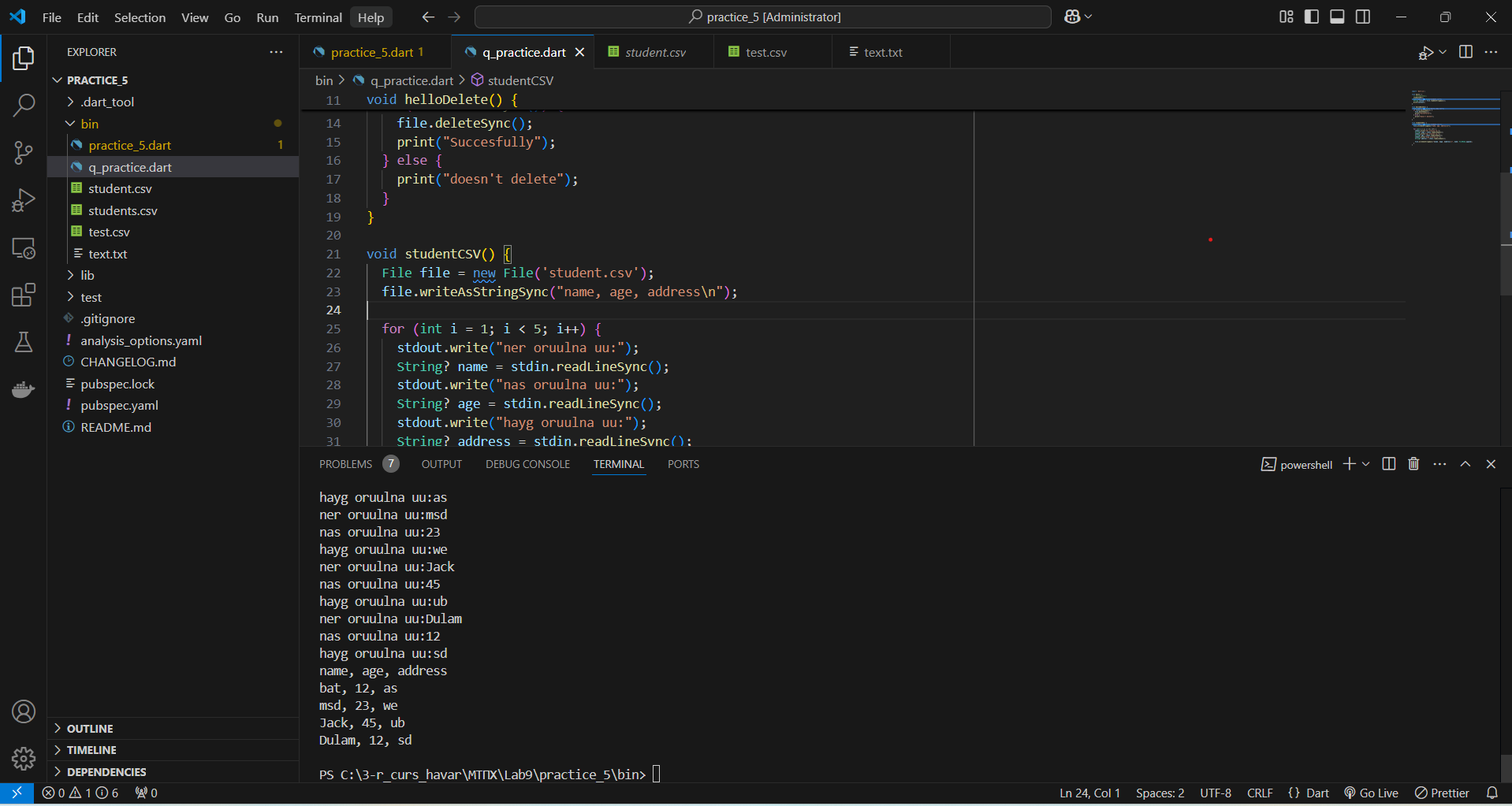
***Зураг 2*** *2дах бүлгийн дасгалуудыг хийж гүйцэтгэсэн*



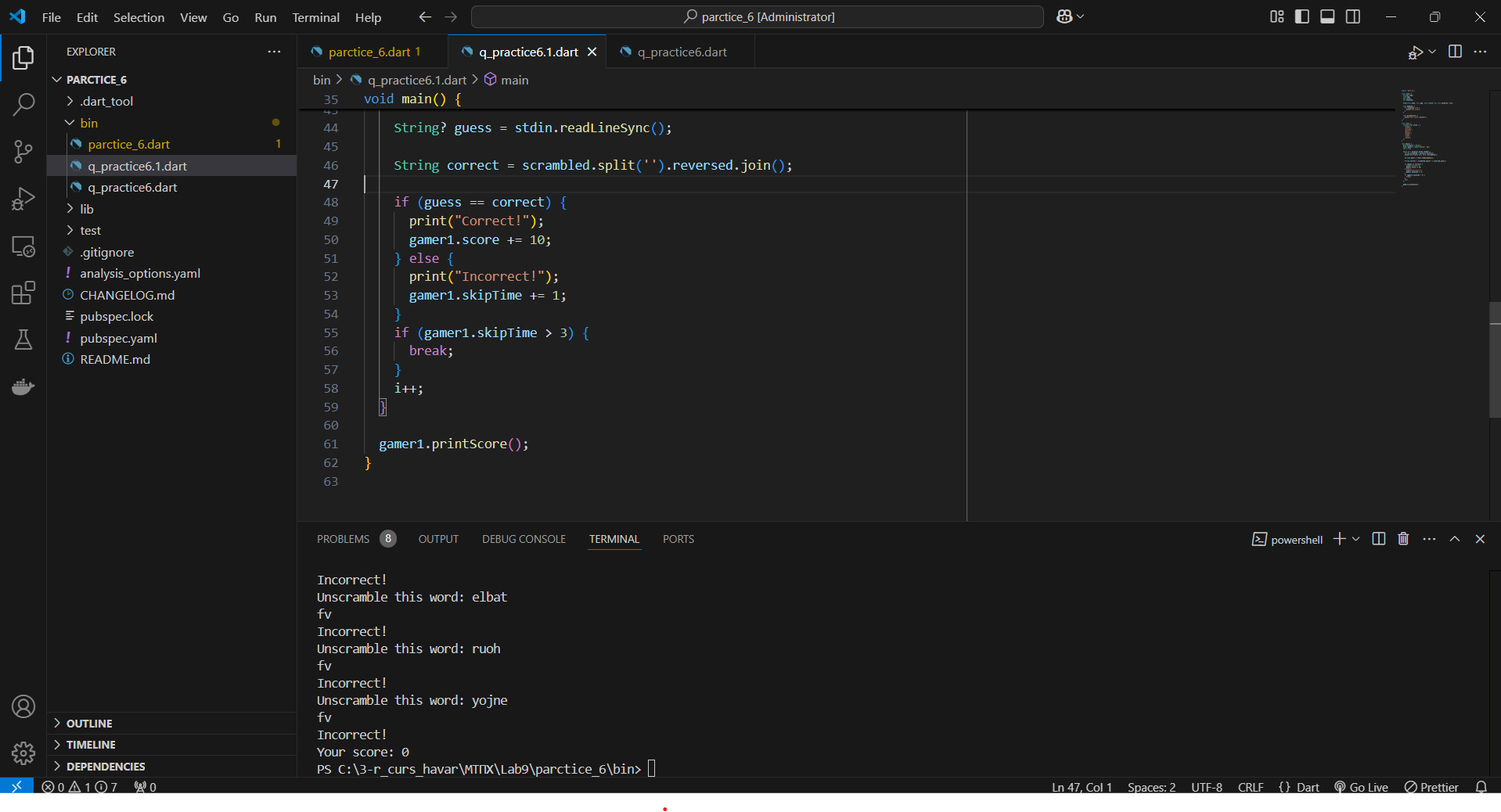
***Зураг 3*** *3дах бүлгийн дасгалуудыг хийж гүйцэтгэсэн*

**

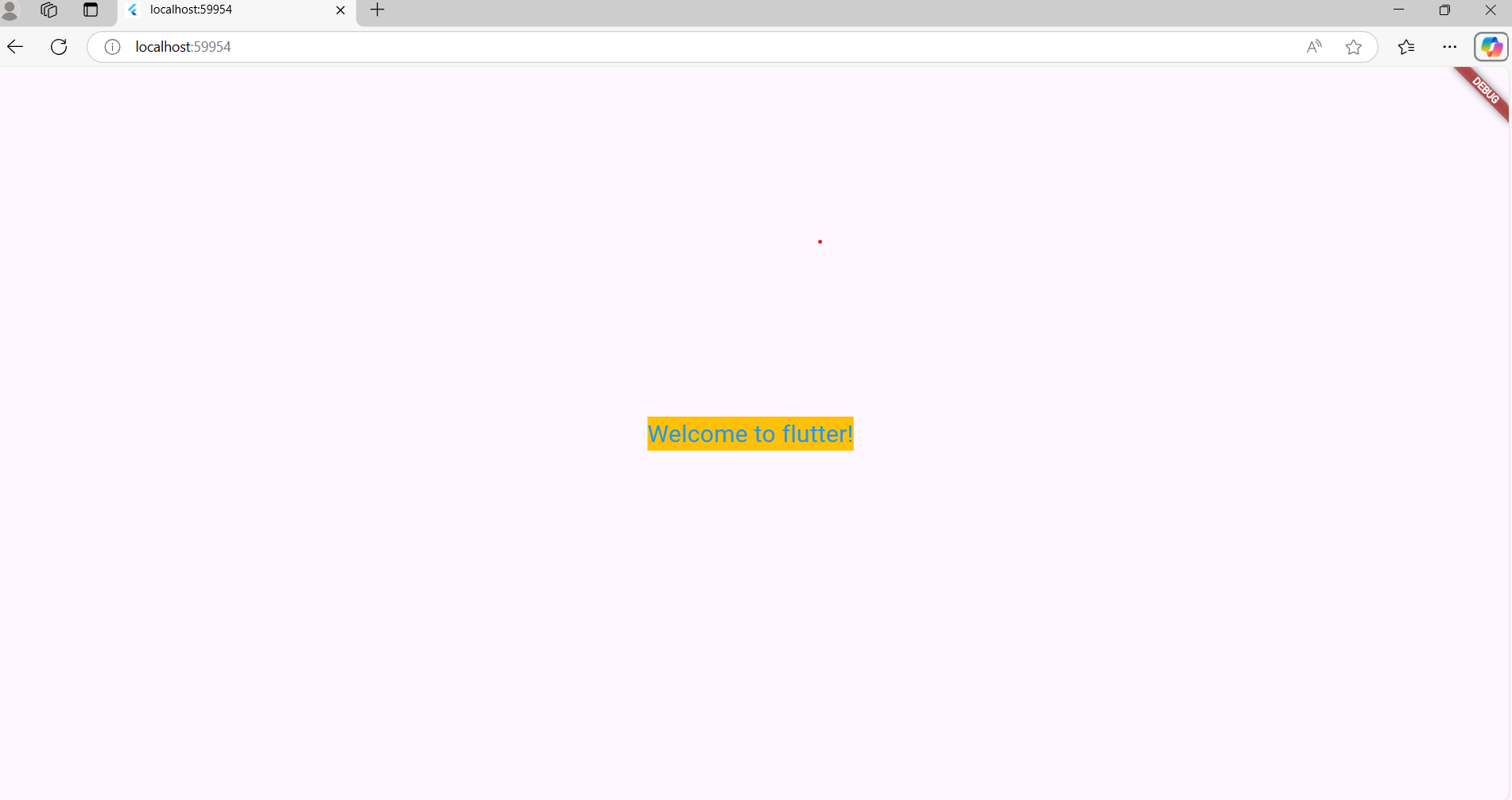
***Зураг 4*** *4дах бүлгийн дасгалуудыг хийж гүйцэтгэсэн*

**

***Зураг 5*** *5дах бүлгийн дасгалуудыг хийж гүйцэтгэсэн*

**

***Зураг 6*** *6дах бүлгийн дасгалуудыг хийж гүйцэтгэсэн*

**

***Зураг7*** *Practice1 хэрэгжүүлсэн*

5.Дүгнэлт

Энэ лабораторийн ажлаар dart программчлалын хэл болон flutter фрэймворктэй танилцан түүний дасгал ажлуудыг хийж гүйцэтгэн үндсэн ойлголтыг эзэмшлээ.

6. Ашигласан материал

(https://developer.android.com/courses/pathways/android-basics-compose-unit-3-pathway-1, n.d.)

(https://developer.android.com/courses/pathways/android-basics-compose-unit-3-pathway-2?\_gl=1\*4nx32f\*\_up\*MQ.., n.d.)

(https://developer.android.com/courses/pathways/android-basics-compose-unit-3-pathway-3, n.d.)