Utilize Variables, Functions, and Methods in Dart using Cloud IDE

Estimated time: **30** minutes

Welcome to this hands-on lab where you will practice using variables, functions, and methods in Dart within a Cloud IDE. This lab will reinforce your understanding of Dart's fundamental concepts and allow you to apply them in a practical setting.

Objectives

After completing this lab, you will be able to:

- Declare and initialize variables in Dart
- Use different data types and understand their significance
- Define and call functions in Dart
- Understand the role of methods in classes and how to use them
- Work with return values and parameters in functions
- Understand type inference, constants, and final variables

About Cloud IDE

Running the lab

This lab is designed to be completed using a Cloud IDE environment. You will be writing and executing Dart code directly within the IDE.

About Skills Network Cloud IDE

Skills Network Cloud IDE (based on Theia and Docker) provides an environment for hands on labs for course and project-related labs. It is an open-source Integrated Development Environment (IDE).

Important notice about this lab environment

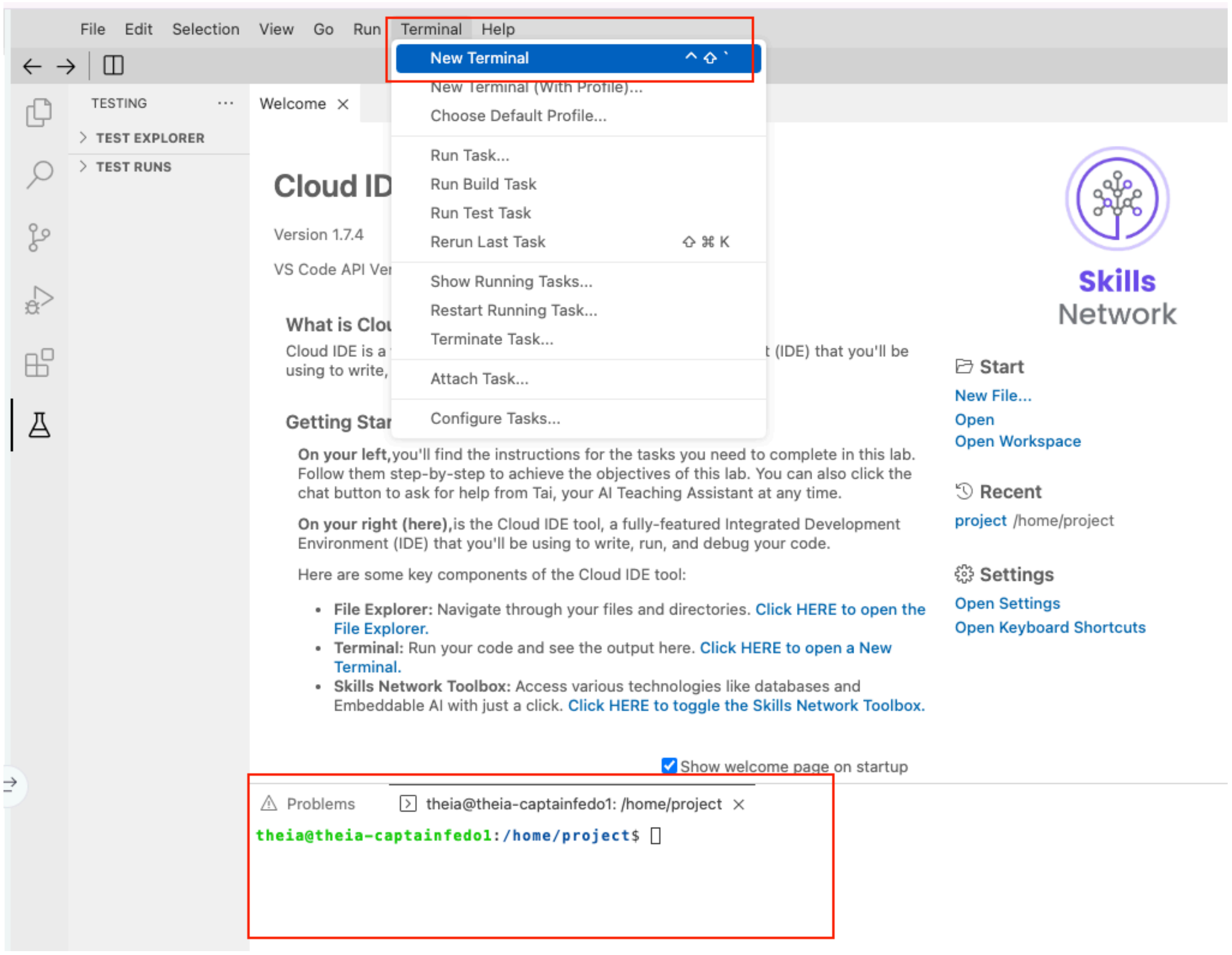
Please be aware that sessions for this lab environment are not persisted. Every time you connect to this lab, a new environment is created for you. Any data you may have saved in the earlier session would get lost. Plan to complete these labs in a single session, to avoid losing your data.

Key terms

- **Variable**: A named storage that can hold a value.
- **Function**: A block of code that performs a specific task and can return a value.
- **Method**: A function defined within a class that operates on objects of that class.
- **Data type**: Defines the type of data that a variable can hold (e.g., `int`, `double`, `String`, etc.).
- **Type inference**: The ability of Dart to automatically determine the type of a variable.
- **Constants**: Variables whose values are fixed during the runtime.
- **Final variables**: Variables that can be assigned only once but are initialized when accessed.

Step 1 - Confirm the development environment

1. Open your terminal or command prompt. Select `Terminal -> New Terminal`.



Cloud ID

Version 1.7.4

VS Code API Ver

What is Clou

Cloud IDE is a … t (IDE) that you'll be using to write,

Getting Star

On your left, you'll find the instructions for the tasks you need to complete in this lab. Follow them step-by-step to achieve the objectives of this lab. You can also click the chat button to ask for help from Tai, your AI Teaching Assistant at any time.

On your right (here), is the Cloud IDE tool, a fully-featured Integrated Development Environment (IDE) that you'll be using to write, run, and debug your code.

Here are some key components of the Cloud IDE tool:

- **File Explorer:** Navigate through your files and directories. Click HERE to open the File Explorer.
- **Terminal:** Run your code and see the output here. Click HERE to open a New Terminal.
- **Skills Network Toolbox:** Access various technologies like databases and Embeddable AI with just a click. Click HERE to toggle the Skills Network Toolbox.

```
theia@theia-captainfedo1:/home/project$
```

2. Run the following command to verify Dart is installed:

```
dart --version
```

You should see a version in the output:

```
theia@theia-captainfedo1:/home/project$ dart --version
Dart SDK version: 3.5.1 (stable) (Tue Aug 13 21:02:17 2024 +0000) on "linux_x64"
```

3. Create a new Dart file named `main.dart`.

```
touch main.dart
```

Step 2: Declare and initialize variables

1. In your `main.dart` file, declare and initialize a few variables with different data types:

```
void main() {
  int age = 25;
  double height = 5.9;
  String name = "John Doe";
  bool isStudent = true;
  print('Name: $name');
  print('Age: $age');
  print('Height: $height');
  print('Is a Student: $isStudent');
}
```

   - **int age = 25;**: Declares an integer variable `age` and initializes it with the value `25`.
   - **double height = 5.9;**: Declares a double variable `height` for decimal numbers and initializes it with `5.9`.
   - **String name = "John Doe";**: Declares a string variable `name` and initializes it with `"John Doe"`.
   - **bool isStudent = true;**: Declares a boolean variable `isStudent` and initializes it with `true`.

2. Run your code to see the output:

```
dart main.dart
```

   The output look as follows:

```
$ dart main.dart
Name: John Doe
Age: 25
Height: 5.9
Is a Student: true
```

Step 3: Explore different data types

1. Replace the previous code with the below code to explore more data types in Dart.

```
void main() {
  List<String> fruits = ['Apple', 'Banana', 'Cherry'];
  Map<String, int> scores = {'Alice': 90, 'Bob': 85, 'Charlie': 95};
  String? nullableString = null;
  print('Fruits: $fruits');
  print('Score of Alice: ${scores['Alice']}');
  print('Nullable String: $nullableString');
}
```

- **List<String> fruits**: Declares a list of strings named `fruits`.
- **Map<String, int> scores**: Declares a map that associates strings with integers.
- **String? nullableString = null;**: Declares a nullable string that is initially set to `null`.

2. **Run the code to understand how Dart handles different data types:**

```
dart main.dart
```

The output should look like the following:

```
Fruits: [Apple, Banana, Cherry]
Score of Alice: 90
Nullable String: null
```

Step 4: Understand type inference and annotations

1. Dart can infer the type of a variable from its value. Let's see this in action:

```
void main() {
    var city = 'New York';  // Type inferred as String
    String country = 'USA';  // Type annotation
    print('City: $city');
    print('Country: $country');
}
```

- **var city = ‘New York’;**: Declares a variable with inferred type `String`.
- **String country = ‘USA’;**: Explicitly declares a variable of type `String`.

2. **Run the code and observe how Dart infers types:**

```
dart main.dart
```

The output should look like this:

```
City: New York
Country: USA
```

Step 5: Use constants and final variables

1. Learn the difference between `const` and `final` by declaring some constants and final variables:

```
void main() {
    const double pi = 3.14;
    final DateTime currentTime = DateTime.now();
    print('Pi: $pi');
    print('Current Time: $currentTime');
}
```

   - **const double pi = 3.14;**: Declares a constant `pi` that is set at compile-time.
   - **final DateTime currentTime = DateTime.now();**: Declares a final variable `currentTime` that is set at runtime but cannot be changed afterward.

2. **Run the code to see how constants and final variables are initialized:**

```
dart main.dart
```

The output should look like this:

```
Pi: 3.14
Current Time: [Current time will vary based on when the code is run]
```

Step 6: Define and call functions

1. Define a simple function that prints a greeting:

```
void greet(String name) {
  print('Hello, $name!');
}
void main() {
  greet('Alice');
}
```

   - **void greet(String name);**: Defines a function named `greet` that takes a `String` parameter `name` and returns no value (`void`).

   - **greet('Alice');**: Calls the `greet` function with the argument `'Alice'`.

2. **Run the code to see the greeting in the console:**

```
dart main.dart
```

Output:

```
Hello, Alice!
```

Step 7: Work with methods in classes

1. Create a class named `Person` with methods:

```
class Person {
  String name;
  int age;
  Person(this.name, this.age);
  void displayInfo() {
    print('Name: $name, Age: $age');
  }
}
void main() {
  Person person = Person('Bob', 30);
  person.displayInfo();
}
```

   - **class Person**: Defines a class named `Person` with attributes `name` and `age`.
   - **Person(this.name, this.age);**: Constructor for initializing `Person` objects with `name` and `age`.
   - **void displayInfo();**: A method that prints the `name` and `age` of the person.

2. **Run your code to see the output:**

```
dart main.dart
```

Output:

```
Name: Bob, Age: 30
```

Conclusion and next steps

Congratulations on completing this lab! You have now practiced using variables, functions, methods, and various data types in Dart. This foundational knowledge will be critical as you continue to explore more advanced features of the Dart programming language.

Next Steps

- Experiment with more complex functions and methods.
- Explore advanced data types like sets and iterators.
- Try working with asynchronous functions and handling exceptions in Dart.

Author(s)

Skills Network