**Lab 4**

LIST

// 1) How do you declare an empty list in Dart?

|  |
| --- |
| void main(){  List lst = List.filled(4, null);  } |

// 2) How do you access elements in a Dart list?

|  |
| --- |
| void main(){  List lst = List.filled(4, null);  print(lst);  } |

// 3) Explain how to add an element to the end of a Dart list

|  |
| --- |
| void main() {  List tst = [ 'One', 'Two' , 'Three' ] ;  print(tst);  print(tst[0]);  tst.add( 'Four' );  print(tst);  } |

// 4) How do you remove an element from a specific index in a Dart list?

|  |
| --- |
| void main() {  List tst = [ 'One', 'Two' , 'Three' ] ;  print(tst);  tst.removeAt(1);  print(tst);  } |

// 5) What is the length property of a Dart list?

|  |
| --- |
| void main() {  List tst = [ 'One', 'Two' , 'Three' ] ;  print(tst);  print(tst.length);  } |

MAP

// 1) How do you declare an empty list in Dart?

|  |
| --- |
| import 'dart:io';  void main() {  dynamic mapType = {};  print(mapType);  } |

// 2) Provide an example of initializing a map with key-value pairs

|  |
| --- |
| import 'dart:io';  void main() {  dynamic mapType = new Map();  mapType['First'] = 'University';  mapType['Second'] = 'of';  mapType['Third'] = 'Lahore';  print(mapType);  } |

// 3) How do you access elements in a Dart list?

|  |
| --- |
| import 'dart:io';  void main() {  dynamic mapType = new Map();  mapType['First'] = 'University';  mapType['Second'] = 'of';  mapType['Third'] = 'Lahore';  print(mapType);  } |

// 4) Explain how to add a new key-value pair to a Dart map.

|  |
| --- |
| import 'dart:io';  void main() {  dynamic mapType = new Map();  mapType['First'] = 'University';  mapType['Second'] = 'of';  mapType['Third'] = 'Lahore';  print(mapType);  mapType['New'] = 'Addition';  print(mapType);  } |

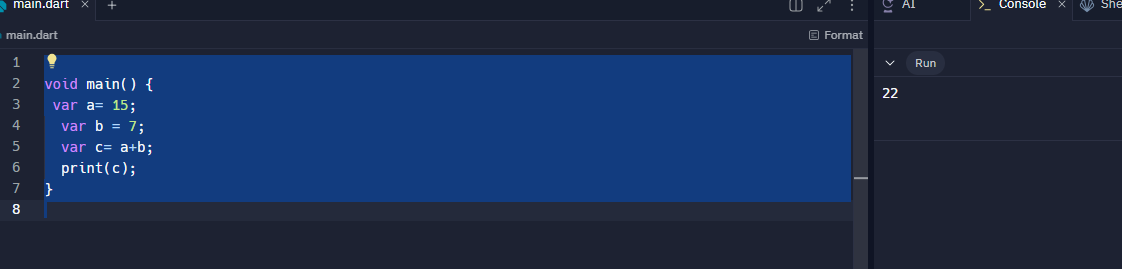
\\ 5) What is the length property of a Dart map?

|  |
| --- |
| import 'dart:io';  void main() {  dynamic mapType = new Map();  mapType['First'] = 'University';  mapType['Second'] = 'of';  mapType['Third'] = 'Lahore';  print(mapType);  print(mapType.length);  } |

**Variables**

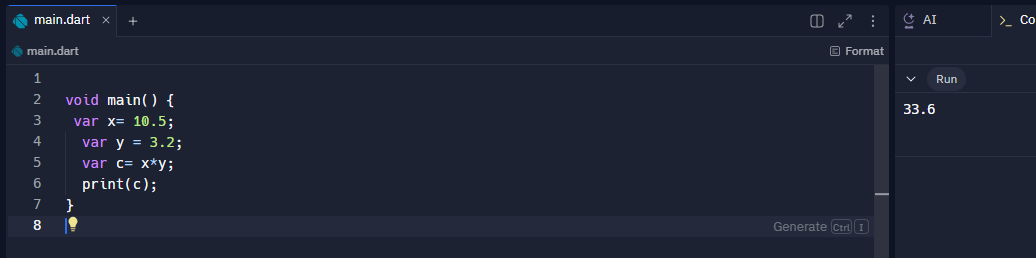
**Q1) Declare two variables, a and b, with values 15 and 7 respectively. Print their sum.**

|  |
| --- |
| void main() {  var a= 15;  var b = 7;  var c= a+b;  print(c);  } |



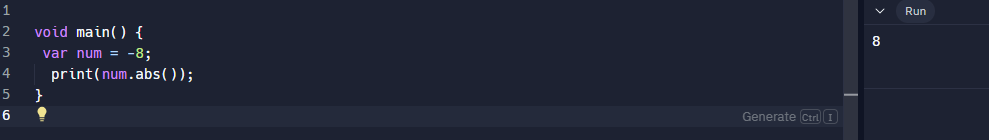
**2) Declare two variables, x and y, with values 10.5 and 3.2 respectively. Print their product.**

|  |
| --- |
| void main() {  var x= 10.5;  var y = 3.2;  var c= x\*y;  print(c);  } |



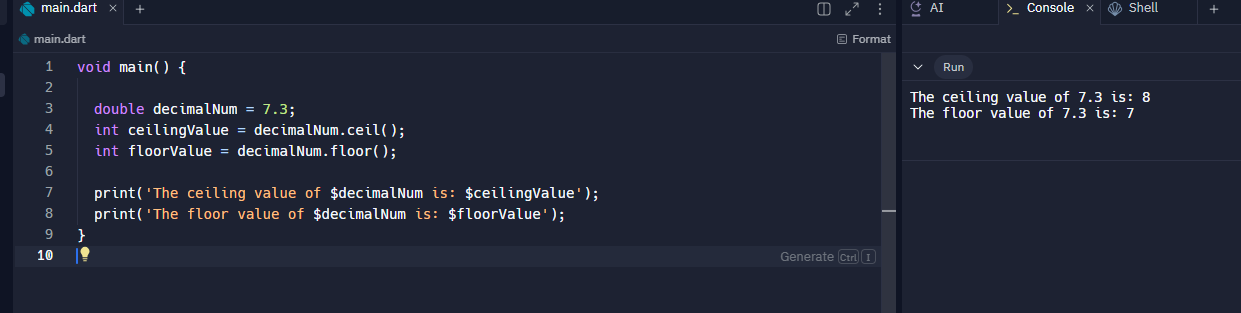
**3) Declare a variable num with a value of -8. Print its absolute value**

|  |
| --- |
| void main() {  var num = -8;  print(num.abs());  } |



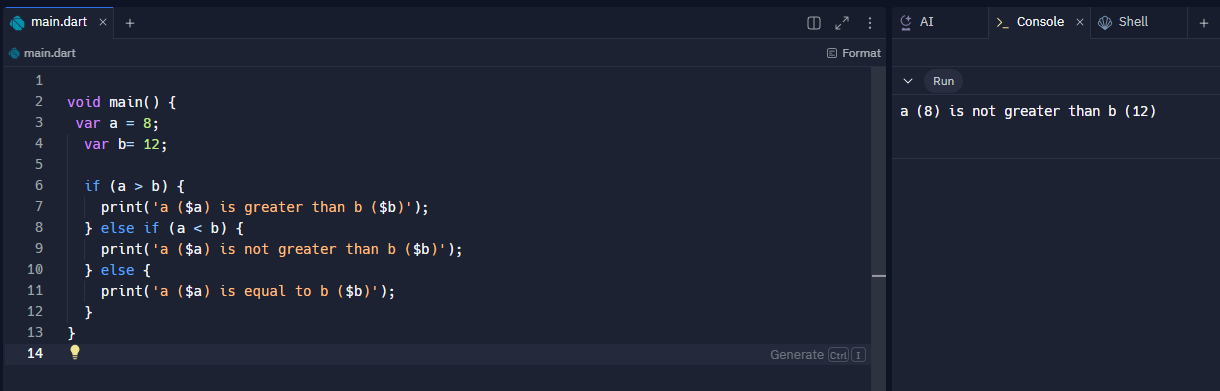
**4) Declare a variable decimalNum with a value of 7.3. Print its ceiling and floor values.**

|  |
| --- |
| void main() {    double decimalNum = 7.3;  int ceilingValue = decimalNum.ceil();  int floorValue = decimalNum.floor();  print('The ceiling value of $decimalNum is: $ceilingValue');  print('The floor value of $decimalNum is: $floorValue');  } |



**5) Compare the values of a and b. Print whether a is greater than b.**

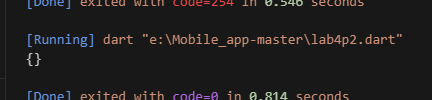
|  |
| --- |
| void main() {  var a = 8;  var b= 12;  if (a > b) {  print('a ($a) is greater than b ($b)');  } else if (a < b) {  print('a ($a) is not greater than b ($b)');  } else {  print('a ($a) is equal to b ($b)');  }  } |



**SETS**

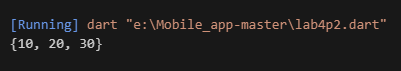
// 1) Declare an empty set named mySet in Dart.

|  |
| --- |
| import 'dart:io';  void main(){  Set<String> newSet = { };  print(newSet);  } |



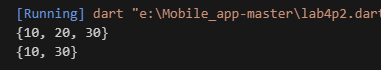
// 2) Add the numbers 10, 20, and 30 to the set.

|  |
| --- |
| import 'dart:io';  void main(){    Set<int> newSet = { 10,20 ,30 };    print(newSet);  } |



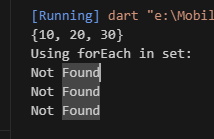
3) Remove the number 20 from the set.

|  |
| --- |
| import 'dart:io';  void main(){    Set<int> newSet = { 10, 20 ,30 };    print(newSet);    newSet.remove(20);    print(newSet);  } |



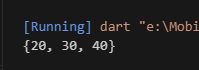
4) Check if the set contains the number 15.

|  |
| --- |
| import 'dart:io';  void main(){      Set<int> newSet = { 10, 20 ,30 };    print(newSet);     print ("Using forEach in set:");     newSet.forEach((element) {    if (element == 15 ) {        print("Found");      } else {        print("Not Found");      }    });  } |



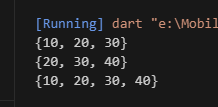
// 5) Create another set named otherSet with numbers 20, 30, and 40.

|  |
| --- |
| import 'dart:io';  void main(){    Set<int> otherSet = { 20 ,30,40 };    print(otherSet);  } |



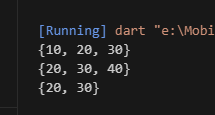
6) Find the union of mySet and otherSet.

|  |
| --- |
| import 'dart:io';  void main(){    Set<int> mySet = { 10, 20 ,30 };    print(mySet);    Set<int> otherSet = { 20 ,30,40 };    print(otherSet);    Set<int> unionSet = mySet.union(otherSet);    print(unionSet);  } |



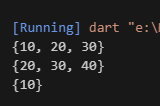
7) Find the intersection of mySet and otherSet.

|  |
| --- |
| import 'dart:io';  void main(){    Set<int> mySet = { 10, 20 ,30 };    print(mySet);    Set<int> otherSet = { 20 ,30,40 };    print(otherSet);    Set<int> intersectSet = mySet.intersection(otherSet);    print(intersectSet);  } |



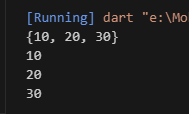
8) Find the difference between mySet and otherSet.

|  |
| --- |
| import 'dart:io';  void main(){    Set<int> mySet = { 10, 20 ,30 };    print(mySet);    Set<int> otherSet = { 20 ,30,40 };    print(otherSet);    Set<int> diffSet = mySet.difference(otherSet);    print(diffSet);  } |



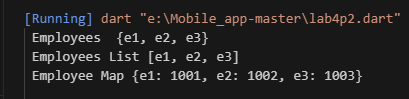
// 9) Print each element in mySet using a forEach() method.

|  |
| --- |
| import 'dart:io';  void main(){    Set<int> mySet = { 10, 20 ,30 };    print(mySet);    mySet.forEach((value) {      print(value);    });  } |



10) Consider a set of unique employee names in a company. Create a Dart program that performs the following tasks: a. Initialize a set named employeeSet with at least three unique employee names. b. Convert the set to a list named employeeList. c. Convert the set to a map named employeeMap where the employee names are used as keys, and their corresponding employee IDs (IDs can be arbitrary) are used as values. d. Print the original set, the converted list, and the converted map.

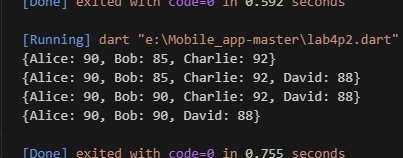
|  |
| --- |
| import 'dart:io';  void main(){    Set<String> employee = { 'e1', 'e2', 'e3' };    print(" Employees  $employee");    List<String> empList = employee.toList();    print(' Employees List $empList');    Map<String, int> empMap = {};    int id =1001;   employee.forEach((emp) {     empMap[emp]= id;      id++;   });    print(" Employee Map $empMap");  } |



MAP

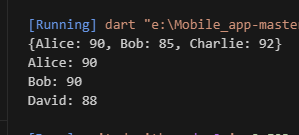
1) Given the map: var studentGrades = {'Alice': 90, 'Bob': 85, 'Charlie': 95}; Perform the following operations: a) Add a new student 'David' with a grade of 88. b) Update Bob's grade to 90. c) Remove the student 'Charlie' from the map.

|  |
| --- |
| void main(){    var studentGrades = {'Alice': 90 , 'Bob': 85, 'Charlie': 92};    print(studentGrades);    studentGrades['David']= 88;    print(studentGrades);    studentGrades['Bob']= 90;    print(studentGrades);    studentGrades.remove('Charlie');    print(studentGrades);  } |



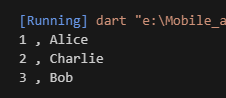
2) Iterate through the map created in question 1 and print each key-value pair.

|  |
| --- |
| void main(){    var studentGrades = {'Alice': 90 , 'Bob': 85, 'Charlie': 92};    print(studentGrades);    studentGrades['David']= 88;    studentGrades['Bob']= 90;    studentGrades.remove('Charlie');      studentGrades.forEach((key, value) {      print('$key: $value');    });  } |



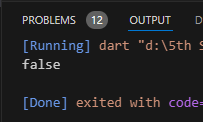
3) Create a map literal with keys as integers and values as strings. Add three key-value pairs of your choice.

|  |
| --- |
| void main(){    var newMap = {1: 'Alice', 2: 'Charlie', 3: 'Bob'};    newMap.forEach((key , value ){      print('$key , $value');    });  } |



4) Compare two map literals for equality. The first map contains: 'a' -> 1, 'b' -> 2. The second map contains the same key-value pairs but in a different order.

|  |
| --- |
| void main(){    var Map1 = {'num1': 1 , 'num2': 2};    var Map2 = {'num1': 2 , 'num2': 1};    print(Map1 == Map2);  } |



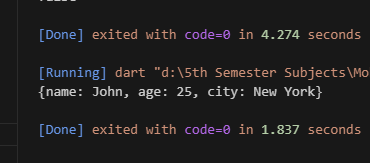
5) Initialize an empty map literal and add the following key-value pairs:

a) name' -> 'John'

b) 'age' -> 25

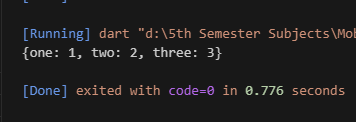
c) 'city' -> 'New York'

|  |
| --- |
| void main(){      var newMap = {};      newMap['name'] = 'John';      newMap['age'] = 25;      newMap['city'] = 'New York';      print(newMap);    } |



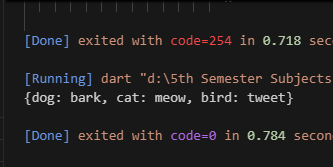
6) Use the from constructor to create a map from the given list of key-value pairs: ['one', 1], ['two', 2], ['three', 3]

|  |
| --- |
| void main() {    var keyValuePairs = [      ['one', 1],      ['two', 2],      ['three', 3]    ];    var map = Map.fromIterable(      keyValuePairs,      key: (pair) => pair[0],      value: (pair) => pair[1]    );    print(map);  } |



7) Use the of constructor to create a map from the given list of key-value pairs: ['dog', 'bark'], ['cat', 'meow'], ['bird', 'tweet']

|  |
| --- |
| void main() {    var map = Map.of({      'dog': 'bark',      'cat': 'meow',      'bird': 'tweet',    });    print(map);  } |



8) Create an unmodifiable map using the unmodifiable constructor with the following key-value pairs: a) 'January' -> 1 b) 'February' -> 2 c) 'March' -> 3

|  |
| --- |
| void main() {    var unmodifiableMap = Map.unmodifiable({      'January': 1,      'February': 2,      'March': 3,    });    print(unmodifiableMap);  } |

