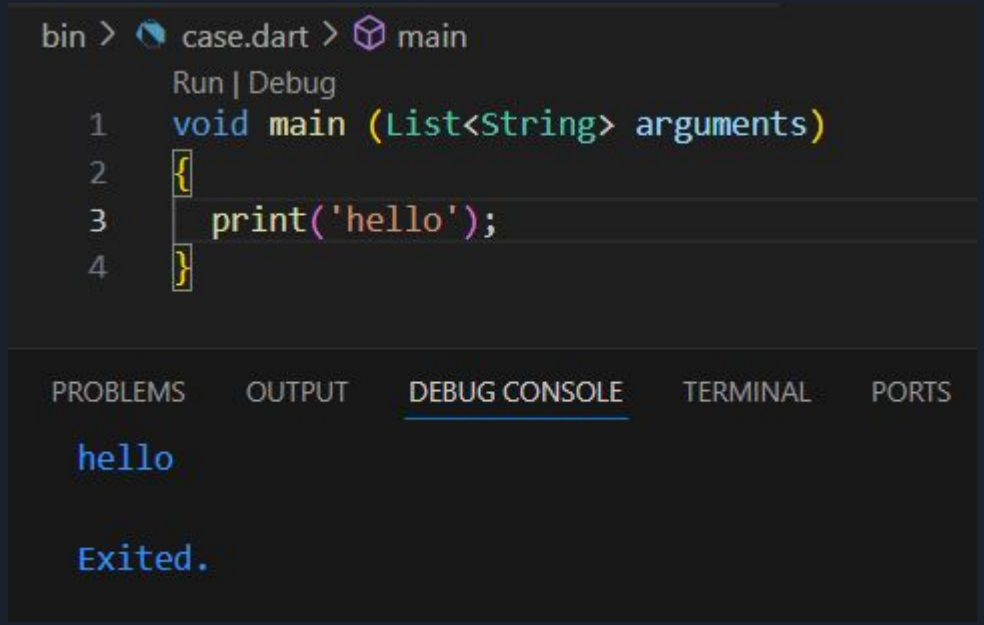
Dart Programming Language Essentials

MODULE 3 CHAPTER 1 - DART

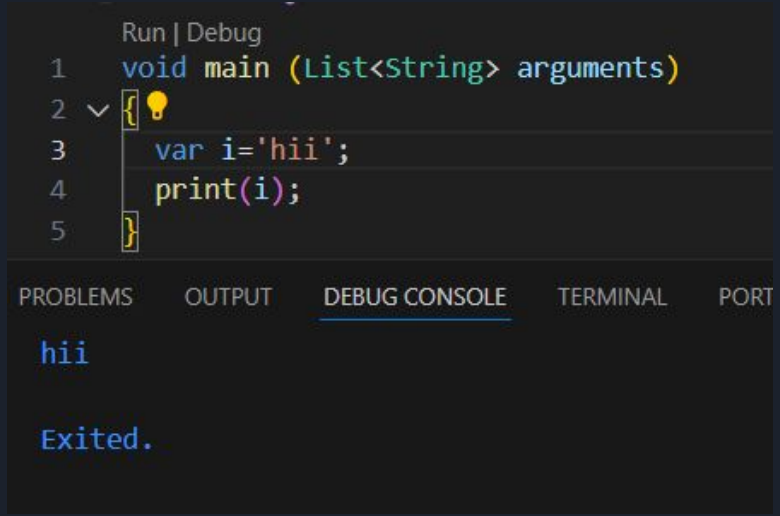
Basic Structure and Entry Point

`main()` **Function:** Every Dart program starts execution at the `main()` function.



Variable Declaration Keywords

Dart offers several keywords to declare variables,



Core Data Types

Category	Type	Purpose
Numbers	int	Whole numbers (non-decimal).
	double	Decimal/floating-point numbers.
Text	String	Sequence of characters (text). Immutable.
Boolean	bool	Logical value: true or false.

```
Run | Debug
1  void main (List<String> arguments)
2  {
3    int age = 30;
4    double price = 19.99;
5    String name = 'abcd ';
6    bool isActive = true;
7    print(age);
8    print(price);
9    print(name);
10   print(isActive);
11 }
```

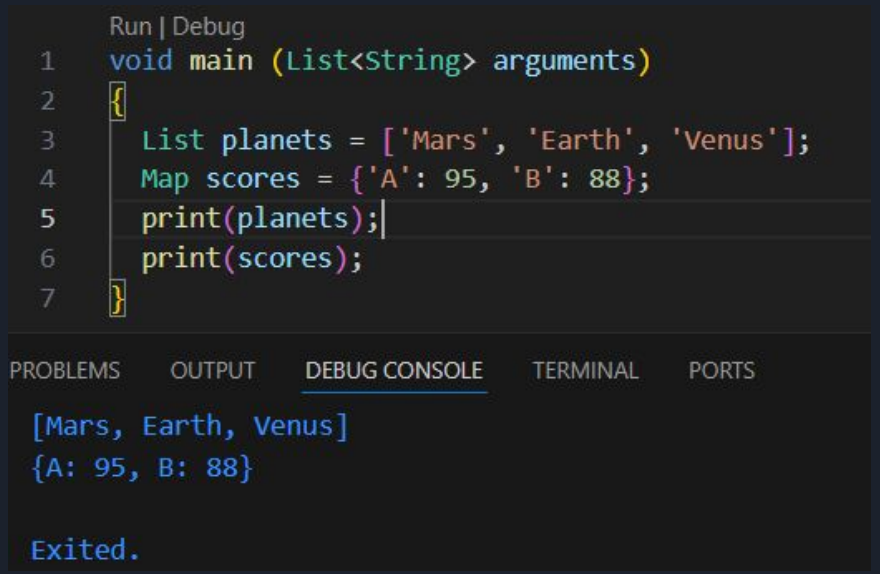
```
ROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL
30
19.99
Dash
true
```

List

List abcd = ['a', 'b', 'c'];

Map

Map abcd = { };



DART OPERATORS TABLE

CATEGORY	OPERATOR	DESCRIPTION	EXAMPLE	OUTPUT / MEANING
ARITHMETIC	+	Addition	10 + 5	15
	-	Subtraction	10 - 5	5
	*	Multiplication	10 * 5	50
	/	Division	10 / 5	2.0
	~/	Integer Division	10 ~/ 3	3
	%	Modulus (Remainder)	10 % 3	1
RELATIONAL	>	Greater than	10 > 5	true
	<	Less than	10 < 5	false
	>=	Greater than or equal to	10 >= 10	true
	<=	Less than or equal to	10 <= 5	false
	==	Equal to	10 == 10	true
	!=	Not equal to	10 != 5	true
LOGICAL	&&	Logical AND	(5 > 2 && 5 < 10)	true
	||	Logical OR	(5 > 10 || 5 == 5)	true
	!	Logical NOT	!(5 > 2)	false
ASSIGNMENT	=	Assign value	a = 10	a = 10
	+=	Add and assign	a += 5	a = a + 5
	-=	Subtract and assign	a -= 5	a = a - 5
	*=	Multiply and assign	a *= 2	a = a * 2
	/=	Divide and assign	a /= 2	a = a / 2
	~/=	Integer divide and assign	a ~/= 3	a = a ~/ 3
	%=	Modulus and assign	a %= 2	a = a % 2
UNARY	++a	Pre-increment	a = 5; ++a;	a = 6
	a++	Post-increment	a = 5; a++;	a = 6
	--a	Pre-decrement	a = 5; --a;	a = 4
	a--	Post-decrement	a = 5; a--;	a = 4
	-a	Negation	a = -5	Negative of value
	!a	Logical negation	a = true; !a;	false

Control Statements

Conditional Statements (Decision-making)-

If else: Execute Block A if a condition is true, Block B otherwise
Nested If: Check an inner condition only if the outer condition is true.
Switch case: Select one block of code to run based on matching a variable's value.

Iteration Statements (Loops)

While: Repeat a block of code as long as the condition is true. (Pre-check)
Do while: Execute the block *at least once*, then repeat as long as the condition is true. (Post-check)
For: Repeat a block a fixed number of times or iterate over a collection.

Jump Statements (Transfer)

Break: Exit the current loop or `switch` block immediately.
Continue: Skip the rest of the current loop iteration and move to the next one.

If ,else if , else

```dart
Run | Debug
void main(List<String> args) {

  var alphabet = 's';

  if (alphabet == 'q')
  {
    print('alphabet is q');
  }

  else if (alphabet == 'r')
  {
    print('alphabet is r');
  }
  else
  {
    print('alphabet is not q or r');
  }

  }

```

PROBLEMS 6 OUTPUT DEBUG CONSOLE TERMINAL

```
alphabet is not q or r

Exited.
```

switch

bin > workshop5.dart > main

```dart
Run | Debug
void main() {
  var grade = 'C';

  switch (gr
    case 'A'
      print('Excellent!');
      break;
    case 'B':
      print('Good!');
      break;
    case 'C':
      print('Fair');
      break;
    case 'D':
      print('Poor!');
      break;
    case 'E':
      print('Fail!');
      break;
    default:
      print('Invalid grade');
  }
}
```

Type: String

PROBLEMS 6 OUTPUT DEBUG CONSOLE TERM

```
Fair

Exited.
```

Nested if

```dart
Run | Debug
void main(List <String> arguments)
{
  int num = 15;
  if (num>10)
  {
    if(num%2==0)
    {
      print('$num is greater than 10,it is even number');
    }
    else
    {
      print('$num is greater than 10,it is odd number');
    }
  }
  else if(num<10 && num%2==0)
  {
    print('$num is Less than 10,it is even number');
  }
  else
  {
    print('$num is Less than 10,it is odd number');
  }
}

```

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS Filter (e.g. text, te

```
15 is greater than 10,it is odd number

Exited.
```

while

bin > workshop1.dart > main

Run | Debug

```dart
void main(List<String> args) {
  var number = 0;

  while (number <= 10)
  {
    print(number);
    number++;
  }
}
```

PROBLEMS 6 OUTPUT DEBUG CONSOLE TER

```
0
1
2
3
4
5
6
7
8
9
10

Exited.
```

do while

bin > workshop2.dart > main

Run | Debug

```dart
void main(List<String> args) {
  var number = 0;

  do {
    print(number);
    number++;
  } while (number <= 10);
}
```

PROBLEMS 6 OUTPUT DEBUG CONSOLE TER

```
0
1
2
3
4
5
6
7
8
9
10

Exited.
```

for

bin > workshop3.dart > main

```dart
var stream = StringInputStream;

class StringInputStream {}

Run | Debug
void main(List<String> args)
  {
    var start = 0; var end = 5;
    for (int i = start; i <= end; i++)
    {
      print(i);
    }

}
```

PROBLEMS 6 OUTPUT DEBUG CONSOLE TERMINAL P

```
0
1
2
3
4
5

Exited.
```

For in

```dart
Run | Debug
void main(List<String> args) {
  List items = [0,1,2,3,4,5];
  for (var item in items) {
    print(item);
  }
}
```

PROBLEMS 48 OUTPUT DEBUG CONSOLE TERMI

```
0
1
2
3
4
5

Exited.
```

break

```dart
Run | Debug
void main(List<String> args) {

  for (int i = 0; i <= 15; i++)
  {
    print(i);
    if (i == 4) {
      if (i == 4) {
        print('breaked in $i');
        break;
      }
    }
  }
}
```

PROBLEMS 48 OUTPUT DEBUG CONSOLE TERMI

```
0
1
2
3
4
breaked in 4

Exited.
```

continues

workshop11 > bin > workshop6.dart > main

```dart
Run | Debug
void main(List<String> args) {
  for (int i = 0; i <= 6; i++) {

    if(i == 5)  {
      print('Skipped $i but loop continues');
      continue;
    }
    print(i);
  }
}
```

PROBLEMS 48 OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
0
1
2
3
4
Skipped 5 but loop continues
6

Exited.
```

EXERCISE 1

* Pyramid

bin > dart_application_1.dart > main

Run | Debug

```dart
void main() {
  int n = 5;
  for (int i = 1; i <= n; i++) {
    String stars = '';
    for (int j = 1; j <= n - i; j++) {
      stars += ' ';
    }
    for (int k = 1; k <= i; k++) {
      stars += '* ';
    }
    print(stars);
  }
}
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
    *
   * *
  * * *
 * * * *
* * * * *

Exited.
```

n → 5

for (int i = 1; i <= n; i++) → the loop will run n times

String star = ''; → Creates an empty string

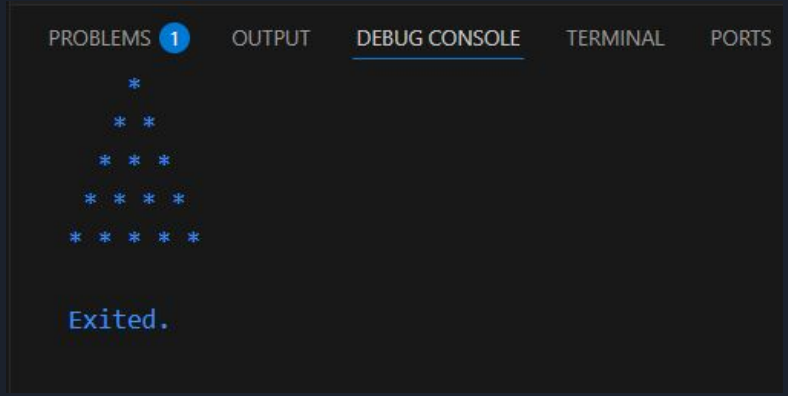
→ Concatenation +=

This loop adds **spaces** before the stars to center-align the pyramid.

This loop adds the **stars (*)** for the current row.

bin > main.dart > ...

```dart
void printpy(int n)
{
  for (int i = 1; i <= n; i++) {
    String star = '';
    for (int j = 1; j <= n - i; j++) {
      star += ' ';
    }
    for (int k = 1; k <= i; k++) {
      star += '* ';
    }
    print(star);
  }
}
Run | Debug
void main()
{
  printpy(5);
}
```



EXERCISE 2

Arithmetic Operations

bin > dart_application_2.dart > main

Run | Debug

```dart
void main(List <String> args)
{

  int a=10;
  int b=3;
  var sum=a+b;
  var difference=a-b;
  var product=a*b;
  var quotient=a/b;
  var remainder=a%b;
  print('SUM : $sum');
  print('DIFFERENCE : $difference');
  print('PRODUCT : $product');
  print('QUOTIENT : $quotient');
  print('REMAINDER : $remainder');
}
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
SUM : 13
DIFFERENCE : 7
PRODUCT : 30
QUOTIENT : 3.3333333333333335
REMAINDER : 1

Exited.
```

Thank You !