

# Introduction to Programming in Turing

## Calculations & Assignment Operator

# The IPO Model

The most basic model for a computer system is the Input-**Processing**-Output (IPO) Model.

In order to interact with the computer as a programmer, we must develop simple examples of each of these stages, which we will then build upon to solve more and more sophisticated problems.

# Output in Turing

Whenever we refer to a string in Turing (and most other languages), we need to put the characters in quotation marks:

```
put "First Program:"
```

```
put "Hello world!"
```

For numbers, no quotation marks are needed. Turing recognizes that they are numbers.

```
put 35
```

```
put 3.14
```

# Mathematical Operations in Turing

To actually make use of the computer's calculating ability, we need to use some mathematical operators.

Operator	Operation	Code
+	Add	$A + B$
-	Subtract	$A - B$
*	Multiply	$A * B$
/	Divide	$A / B$
**	Exponent	$A ** B$

# Math using Turing

```
put 3 + 5           % output is 8
put 4 - 11          % output is -7
put 2 * 6           % output is 12
put 7 / 2           % output is 3.5
```

Remember that Order of Operations (BEDMAS) applies to what you are doing. You can use brackets to ensure calculations are done in the order you want.

# Example: Using Brackets for Order of Operations

$$\frac{4+6}{3-5}$$

How would the output from these commands be different? Which is correct?

```
put 4 + 6 / 3 - 5
```

```
put 4 + (6 / 3) - 5
```

```
put 4 + 6 / (3 - 5)
```

```
put (4 + 6) / (3 - 5)
```

# External & Internal Variables

When we use the get command to assign a variable, the information is *external* – the user provides the data directly.

It is often useful to have variables that are used *internally* in the program.

As programs become more complicated, it becomes necessary to do extra calculations and store useful information.

# Example – External Variables

(all variables given values using get)

Ask the user to enter 2 numbers and then display the sum.

```
var num1, num2 : int
```

```
put "Number 1: "..  
get num1
```

```
put "Number 2: "..  
get num2
```

```
put "The sum is "..  
put num1 + num2
```



# Example – Internal Variables

(some variables do not use get)

```
var num1, num2 : int
var sum : int

put "Number 1: " ..
get num1
put "Number 2: " ..
get num2

sum := num1 + num2

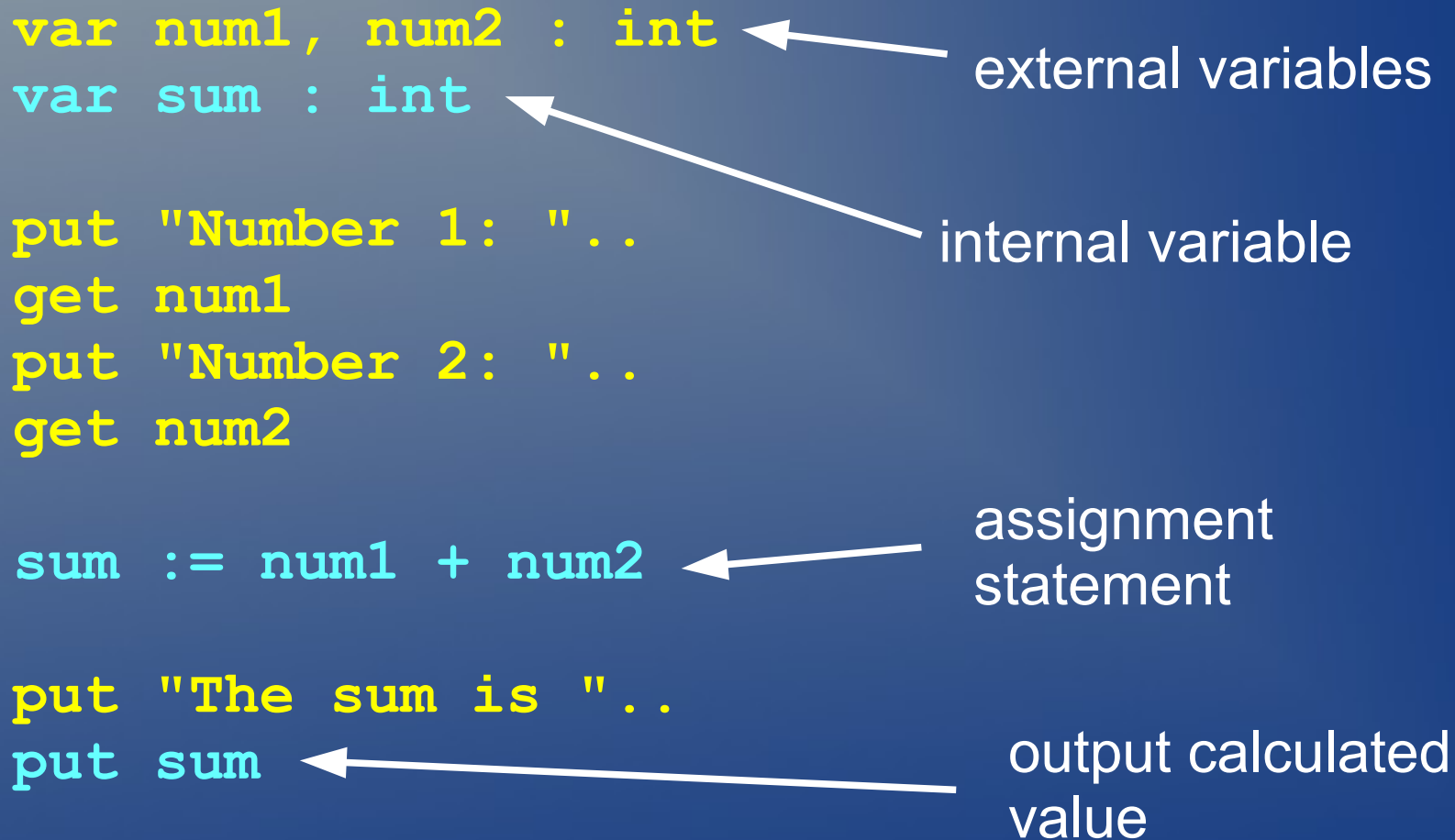
put "The sum is " ..
put sum
```

external variables

internal variable

assignment statement

output calculated value

The diagram consists of a code block on the left and four labels with arrows on the right. The label 'external variables' has an arrow pointing to the 'num1, num2' part of the first line of code. The label 'internal variable' has an arrow pointing to the 'sum' part of the second line of code. The label 'assignment statement' has an arrow pointing to the ':=' operator in the third line of code. The label 'output calculated value' has an arrow pointing to the 'sum' variable in the final line of code.

# Assignment Operator

The assignment operator is the command where we assign a value to a variable.

The value can be a constant or the result of a calculation.

```
var a, b, c : int
```

```
a := 5
```

```
b := 6
```

```
c := a + b - 11
```

# Assignment Operator

## Program

```
var a, b, c : int
```

```
a := 5
```

```
b := 6
```

```
c := a + b - 11
```

```
put a
```

```
put b
```

```
put c
```

## What the Computer Sees

```
var a, b, c : int
```

```
a := 5
```

```
b := 6
```

```
c := 5 + 6 - 11
```

```
put a
```

```
put b
```

```
put c
```

Since the computer knows a is 5 and b is 6, then  $5 + 6 - 11$  is 0. Therefore the final result of this calculation is that c is equal to 0.