

Introduction to Programming in Turing

Output of Information

Output in Turing

Joining Multiple Lines

It is possible to join the output from two different commands on a single line. Use the put command, but add .. to the end of the line.

Note: The last put command should not have the .. on the end.



```
put "To be or not to be, " ..  
put "that is the question."
```

Output in Turing

Joining Multiple Lines

The ".." operator can join more than two lines (although this example is a terrible waste). Don't forget that the last line ends without the dots.

```
put "To " ..  
put "be " ..  
put "or " ..  
put "not " ..  
put "to " ..  
put "be "
```

Output in Turing

Joining Multiple Elements

It is also possible to join multiple elements using a comma on a single output line. This is not useful when using only strings, but as we learn more programming, it will become more useful.

```
put "To ", "be ", "or " ..  
put "not ", "to ", "be "
```

Output in Turing

Output Quotation Marks

Quotation marks are used to represent strings in Turing. They are also used to represent an actual quotation in written text.

How can we display quotation marks as part of our Turing output?

~~put ""quote?""~~

Although a reasonable guess, this fails. You can try this yourself.

Output in Turing

Output Quotation Marks

When Turing sees a quotation mark, it interprets it as a string. What we need is a way to tell Turing to interpret it in a second way.

Turing uses another character, the slash `\`, to indicate an alternate, or secondary, meaning.

<code>put</code>	<code>"</code>	<code>a</code>	<code>"</code>
<code>put</code>	<code>"</code>	<code>\"</code>	<code>"</code>
<code>put</code>	<code>"</code>	<code>\\</code>	<code>"</code>

Output in Turing Escape Sequence

The slash `\`, indicates an alternate, or secondary, meaning. This is commonly referred to as an escape sequence (i.e., "escape" from the primary meaning). Here are some examples:

symbol	primary meaning	escape sequence	secondary meaning
"	string output	<code>\"</code>	output quotation mark character
n	output letter 'n'	<code>\n</code>	add new line to output
<code>\</code>	escape sequence	<code>\\</code>	output slash character

Formatted Output in Turing

To have more control over the appearance of output, you may wish to format your output. Here is some unformatted output.

```
put "This" ..  
put "is" ..  
put "a" ..  
put "test!!!"
```

Output:

```
Thisisatest!!!  
12341211234567
```


Formatted Output in Turing

With formatting, it is possible to specify how many spaces will be required for each output. In this case, each output will use at least 5 spaces (notice the last output required more than 5, so it took 7).

```
put "This " : 5 ..
```

```
put "is " : 5 ..
```

```
put "a " : 5..
```

```
put "test!!!" : 5
```

Output:

```
This is      a      test!!!
```

```
1234512345123451234567
```

Formatted Output in Turing

For numbers, it generally works the same way. The difference is that some numbers (real) may have decimals. It is also possible to use formatting to specify the number of decimal places.

`put <number> : <minimum spaces> : <decimal places>`

```
put 1 : 5 : 2
put 22222.2 : 5 : 2
put 3.1415 : 5 : 2
put 4.9999 : 5 : 2
```

In this example, for each output we have specified at least 5 spaces wide, and 2 decimal places for each.

Formatted Output in Turing


put <number> : <minimum spaces> : <decimal places>

```
put 1 : 5 : 2
put 22222.2 : 5 : 2
put 3.1415 : 5 : 2
put 4.9999 : 5 : 2
```

Output:

```
1.00
22222.20
3.14
5.00
12345
```

Most of these numbers only needed 4 spaces (1 leading digit, 1 decimal point, and 2 decimal places). The 5th space was put at the beginning.



One number needed 5 leading digits, 1 decimal point, and 2 decimal places, so it exceeded the specified 5 spaces.