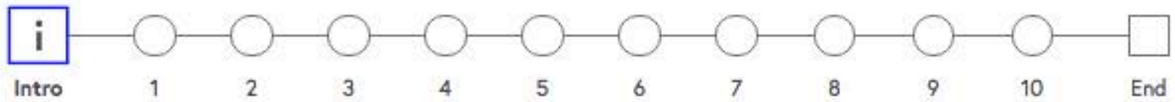
# How well do you know Erlang?




This final test covers some of the more advanced concepts we've been looking at in Week 3 of the course.

TEST RULES AND GRADING

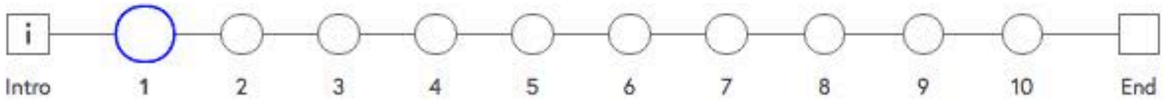
- You may take 3 attempts to answer each question
- Each question has 3 points available
- A point will be deducted for each incorrect attempt
- You can review your total score for the test at the end
- If you want to buy a Certificate of Achievement for this course, you will need to score an average of 70% or above on all tests
- You cannot repeat a test to improve your score
- You can check your average test score for this course so far on your Progress page

Begin test

WHAT DO YOU THINK ABOUT ERLANG?
DISCUSSION

SKIP TEST
GO TO STEP 3.18

# How well do you know Erlang?



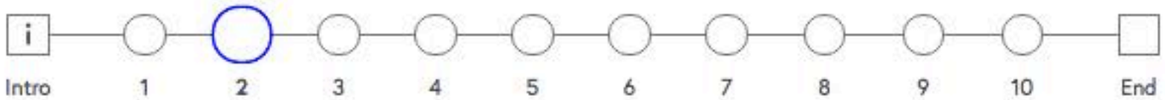

## Question 1

**Which of the following statements about Erlang programming is false?**

- Erlang uses single assignment, so that an instance of a variable cannot be re-assigned once it has a value. It is, however, possible to pattern match against bound variables.
- Evaluation in Erlang is demand-driven: an argument to a function is only evaluated if its value is needed by subsequent computation.
- Functions in Erlang are "first-class citizens": they can be included in data structures, be passed to functions as parameters, returned as results, and compared using equality and ordering.

**Tries left:** 3

# How well do you know Erlang?




## Question 2

**Which of these statements about type checking in Erlang is correct?**

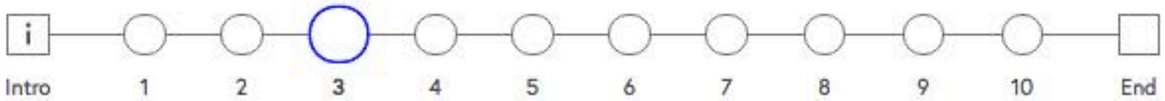
- Most type checking for Erlang is performed at runtime, even in some cases when it is possible to check a condition at compile time.
- Type checking for Erlang could take place entirely at compile time, so that no type errors would be generated from running code.
- No type checking takes place at compile time in Erlang.

**Tries left:** 3

PREVIOUS QUESTION

SKIP QUESTION

# How well do you know Erlang?




## Question 3

**What is the result of evaluating the following expression?**

```
lists:foldr(fun(X,Y) when X>Y -> Y; (X,Y) -> X end, 0, [2,1,-4,2,4]).
```

- -4
- 0
- 4

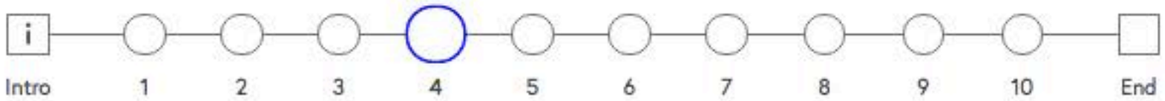
**Tries left:** 3

PREVIOUS QUESTION

SKIP QUESTION

Support

# How well do you know Erlang?




## Question 4

### What is the result of evaluating the following expression?

```
lists:foldr(fun(X,Y) -> [X|Y] end, [2,1,-4,2,4], [2,1,-4,2,4]).
```

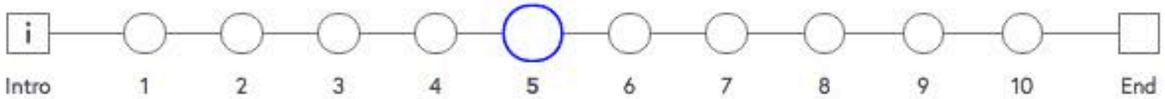
- [4,2,-4,1,2,2,1,-4,2,4]
- [2,1,-4,2,2,1,-4,2,4]
- [2,1,-4,2,4,2,1,-4,2,4]

Tries left: 3

PREVIOUS QUESTION

SKIP QUESTION

# How well do you know Erlang?




## Question 5

**Which of the following definitions is a correct implementation of a function to return the Nth element (first argument) of a list (second argument).**

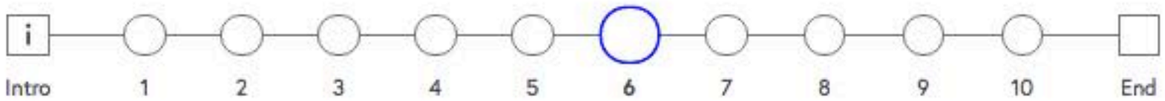
```
nth(0,[X|_]) -> X;
nth(N,[_|Xs]) -> nth(N-1,Xs);
nth(N,Xs) -> 0.
```

```
nth(0,[X|_]) -> X;
nth(N,[_|Xs]) -> nth(N-1,Xs).
```

```
nth(N,[_|Xs]) -> nth(N-1,Xs);
nth(0,[X|_]) -> X.
```

Tries left: 3

# How well do you know Erlang?




## Question 6

**Which higher-order function would you use to implement `lists:zipwith` assuming that you could call `lists:zip` in your definition?**

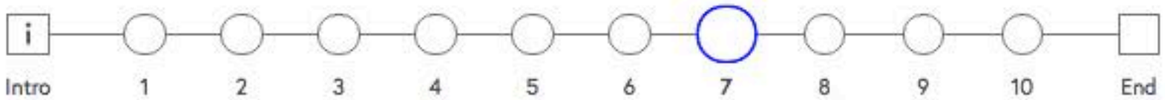
- The filter function, `lists:filter`.
- The function `lists:splitwith`.
- The function `lists:map`.

**Tries left:** 3

PREVIOUS QUESTION

SKIP QUESTION

# How well do you know Erlang?




## Question 7

**Which of these is *not* a feature of Erlang?**

- Explicit memory allocation.
- Compilation to virtual machine code.
- Garbage collection.
- Multi-platform.

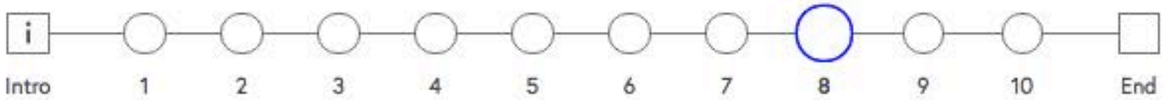
**Tries left:** 3

PREVIOUS QUESTION

SKIP QUESTION

Support

# How well do you know Erlang?




## Question 8

**Which of the following is an Erlang atom?**

- ◯ `case`
- ◯ `'i'm an atom'`
- ◯ `'case'`
- ◯ `"case"`

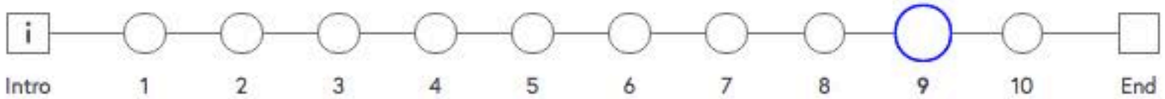
**Tries left:** 3

PREVIOUS QUESTION

SKIP QUESTION

Support

# How well do you know Erlang?




## Question 9

**Which of the following is not a feature of Erlang?**

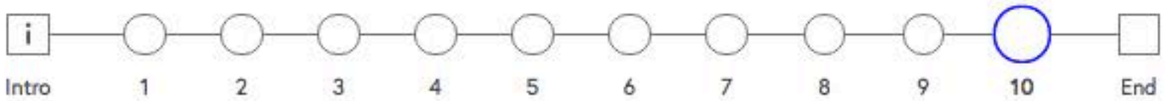
- Passing a function as an argument.
- Building a list of functions.
- Returning a function as the result of a function.
- Passing one argument to a two argument function.

**Tries left:** 3

PREVIOUS QUESTION

SKIP QUESTION

# How well do you know Erlang?




## Question 10

**Which of the following statements is true?**

- Erlang functions cannot be compared for equality.
- The operators == and =:= give the same results when applied to numbers.
- Evaluating `lists:map == lists:filter` will return `false`.
- Atoms can be compared for equality and ordering.

**Tries left:** 3

PREVIOUS QUESTION

SUMMARY