Challenge 1: Basic Operations

Solve the following using the operator methods in one line of code.

If you start with 2 fish, and they breed twice, producing 71 offspring the first time, and 233 offspring the second time, and then 13 fish are swallowed by a hungry moray eel, how many fish do you have left? How many aquariums do you need if you can put 30 fish per aquarium?

* **Hint:** You can chain method calls.
* **Hint:** You can call the methods on numbers, and Kotlin will convert them to objects for you.
* **Bonus question:** What is special about all the numbers of fish?

### Challenge 2: Variables

##### Create a String variable rainbowColor, set its color value, then change it.

##### Create a variable blackColor whose value cannot be changed once assigned. Try changing it anyway.

### Challenge 3: Nullability

##### Try to set rainbowColor to null. Declare two variables, greenColor and blueColor. Use two different ways of setting them to null.

### Challenge 4: Nullability/Lists

##### Create a list with two elements that are null; do it in two different ways.

##### Next, create a list where the list is null.

### Challenge 5: Null Checks

##### Create a nullable integer variable called nullTest, and set it to null. Use a null-check that increases the value by one if it's not null, otherwise returns 0, and prints the result.

**Hint:** Use the Elvis operator.

Read the code below, try to follow what it does, and then choose the correct answer:

**var** welcomeMessage = "Hello and welcome to Kotlin"

when (welcomeMessage.length) {

0 -> println("Nothing to say?")

**in** 1..50 -> println("Perfect")

**else** -> println("Too long!")

}

### QUIZ QUESTION

What will the code above print?

##### 

##### Challenge 6

##### Create three String variables for trout, haddock, and snapper.

##### Use a String template to print whether you do or don't like to eat these kinds of fish.

### Challenge 7

when statements in Kotlin are like case or switch statements in other languages.

Create a when statement with three comparisons:

##### If the length of the fishName is 0, print an error message.

##### If the length is in the range of 3...12, print "Good fish name".

##### If it's anything else, print "OK fish name".

Challenge 8

Read the code below, and then select the correct array initialization that will display the corresponding output.

val array = // initalize array here

val sizes = arrayOf("byte", "kilobyte", "megabyte", "gigabyte",

"terabyte", "petabyte", "exabyte")

**for** ((i, value) **in** array.withIndex()) {

println("1 ${sizes[i]} = ${value.toLong()} bytes")

}

Output:

1 byte = 1 bytes

1 kilobyte = 1000 bytes

1 megabyte = 1000000 bytes

1 gigabyte = 1000000000 bytes

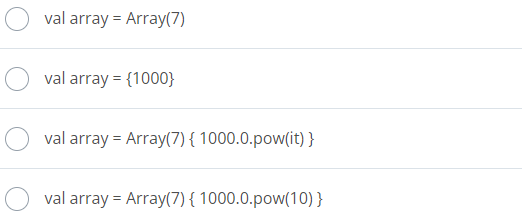
1 terabyte = 1000000000000 bytes

1 petabyte = 1000000000000000 bytes

1 exabyte = 1000000000000000000 bytes

**QUIZ QUESTION**

Which of the statements below is the correct way to initialize the array above to give the correct output.



Challenge 9

Looping over arrays and lists is a fundamental technique that has a lot of flexibility in Kotlin. Let's practice.

#### Basic example

##### Create an integer array of numbers called numbers, from 11 to 15.

##### Create an empty mutable list for Strings.

##### Write a for loop that loops over the array and adds the string representation of each number to the list.

#### Challenge example

##### How can you use a for loop to create (a list of) the numbers between 0 and 100 that are divisible by 7?

### Challenge 10

##### Create a new Kotlin file.

##### Copy and paste the main() function from Hello World into the file.

##### Create a new function, dayOfWeek().

##### In the body of the function, print "What day is it today?"

##### Call dayOfWeek() from main().

##### Run your program.

(More) In the body of the dayOfWeek() function, answer the question by printing the current day of the week.

#### Hints

##### You can use Java libraries (java.util) from Kotlin. For example, to get the day of the week:

* Calendar.getInstance().get(Calendar.DAY\_OF\_WEEK)

##### Type in the code, then press Option + Enter in Mac, or Alt + Enter in Windows, over the red Calendar class to import the library.

##### Use a when statement to print a string depending on the day. Sunday is the first day of the week.

### Challenge 11

Create a program with a function that returns a fortune cookie message that you can print.

##### Create a main() function.

##### From the main() function, call a function, getFortuneCookie(), that returns a String.

##### Create a getFortuneCookie() function that takes no arguments and returns a String.

1. In the body of getFortuneCookie(), create a list of fortunes. Here are some ideas:

##### "You will have a great day!"

##### "Things will go well for you today."

##### "Enjoy a wonderful day of success."

##### "Be humble and all will turn out well."

##### "Today is a good day for exercising restraint."

##### "Take it easy and enjoy life!"

##### "Treasure your friends because they are your greatest fortune."

1. Below the list, print: "Enter your birthday: "

##### ****Hint:**** Use print(), not println()

##### Create a variable, birthday.

##### Read the user's input form the standard input and assign it to birthday. If there is no valid input, set birthday to 1.

##### ****Hint:**** Use readLine() to read a line of input (completed with Enter) as a String.

##### ****Hint:**** In Kotlin, you can use toIntOrNull() to convert a number as a String to an Integer numeric. If the user enters "", toIntOrNull returns null.

##### ****Hint:**** Check for null using the ? operator and use the ?: operator to handle the null case.

##### Divide the birthday by the number of fortunes, and use the remainder as the index for the fortune to return.

##### Return the fortune.

##### In main(), print: "Your fortune is: ", followed by the fortune string.

### Extra practice:

Use a for loop to run the program 10 times, or until the "Take it easy" fortune has been selected.

### Challenge 12

Improve your whatShouldIDoToday() program with the new knowledge from this segment.

##### Add 3 more situations and activities. For example:

1. mood == "sad" && weather == "rainy" && temperature == 0 -> "stay in bed"
2. temperature > 35 -> "go swimming"

##### Create a single-expression function for each condition and then use it in your when expression.

### Challenge

Instead of passing in the mood, get a mood string from the user.

**Hint:** The !! operator may come handy.

### Loops

This lesson introduced the while and repeat loops. To practice using them, do the following:

##### Change your fortune cookie program to use repeat() instead of a for loop. What happens to the break instruction? Using the error message from the compiler, with what you've learned so far, can you think of why?

##### Change your fortune cookie program to use a while loop, which is the better choice when you are looping until a condition is met.

### Challenge 13

You can do the following filter exercise in REPL.

##### Create a list of spices, as follows:

1. val spices = listOf("curry", "pepper", "cayenne", "ginger", "red curry", "green curry", "red pepper" )

##### Create a filter that gets all the curries and sorts them by string length.

##### ****Hint:**** After you type the dot (.), IntelliJ will give you a list of functions you can apply.

##### Filter the list of spices to return all the spices that start with 'c' and end in 'e'. Do it in two different ways.

##### Take the first three elements of the list and return the ones that start with 'c'.

**Note:** We will be able to do a lot more interesting stuff with filters after you learn about classes and Map.

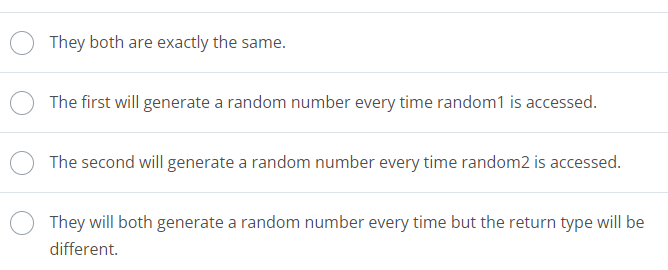
### QUIZ QUESTION

What is the difference between

**val** random1 = random()

**val** random2 = {random()}

Try it out in REPL or a file:



### Challenge 14: Lambdas

##### Create a lambda and assign it to rollDice, which returns a dice roll (number between 1 and 12).

##### Extend the lambda to take an argument indicating the number of sides of the dice used for the roll.

##### If you haven't done so, fix the lambda to return 0 if the number of sides passed in is 0.

##### Create a new variable, rollDice2, for this same lambda using the function type notation.

(more)

##### Why would you want to use the function type notation instead of just the lambda?

##### Create a function gamePlay() that takes a roll of the dice as an argument and prints it out.

##### Pass your rollDice2 function as an argument to gamePlay() to generate a dice roll every time gamePlay() is called.

### Challenge 15

Let's improve our SimpleSpice class so that we can have various spices with different levels of spiciness.

##### Create a new class, Spice.

##### Pass in a mandatory String argument for the name, and a String argument for the level of spiciness where the default value is mild for not spicy.

##### Add a variable, heat, to your class, with a getter that returns a numeric value for each type of spiciness.

##### Instead of the list of spices as Strings you used earlier, create a list of Spice objects and give each object a name and a spiciness level.

##### Add an init block that prints out the values for the object after it has been created. Create a spice.

##### Create a list of spices that are spicy or less than spicy. Hint: Use a filter and the heat property.

##### Because salt is a very common spice, create a helper function called makeSalt().

### Challenge 16

Let's talk about books for a moment, those heavy tomes of paper with printed letters.

##### Create a class, Book, with a title and an author.

##### Add a method, readPage(), that increases the value of a private variable, currentPage, by 1.

##### Create a subclass of Book; name it eBook.

##### eBook also takes in a format, which defaults to "text".

##### In eBooks, counting words makes more sense than pages. Override the readPage() method to increase the word count by 250, the average number of words per page from typewriter days.

### Challenge 17

### Abstract and Interface

Let's go back to your spices. Make Spice an abstract class, and then create some subclasses that are actual spices.

##### It's easiest (organizationally) if you make a new package, Spices, with a file, Spice, that has a main() function.

##### Copy/paste your Spice class code into that new file.

##### Make Spice abstract.

##### Create a subclass, Curry. Curry can have varying levels of spiciness, so we don't want to use the default value, but rather pass in the spiciness value.

##### Spices are processed in different ways before they can be used. Add an abstract method prepareSpice to Spice, and implement it in Curry.

##### Curry is ground into a powder, so let's call a method grind(). However, grinding is something that's not unique to curry, or even to spices, and it's always done in a grinder. So we can create an Interface, Grinder, that implements the grind() method. Do that now.

##### Then add the Grinder interface to the Curry class.

### Delegation

Using the provided code from the lesson for guidance, add a yellow color to Curry.

fun main (args: **Array**<**String**>) {

delegate()

}

fun delegate() {

**val** pleco = **Plecostomus**()

println("Fish has has color ${pleco.color}")

pleco.eat()

}

interface **FishAction** {

fun eat()

}

interface **FishColor** {

**val** color: **String**

}

**object** **GoldColor** : **FishColor** {

**override** **val** color = "gold"

}

**class** **PrintingFishAction**(**val** food: **String**) : **FishAction** {

**override** fun eat() {

println(food)

}

}

**class** **Plecostomus** (fishColor: **FishColor** = **GoldColor**):

**FishAction** by **PrintingFishAction**("eat a lot of algae"),

**FishColor** by fishColor

##### Create an interface, SpiceColor, that has a color property. You can use a String for the color.

##### Create a singleton subclass, YellowSpiceColor, using the object keyword, because all instances of Curry and other spices can use the same YellowSpiceColor instance.

##### Add a color property to Curry of type SpiceColor, and set the default value to YellowSpiceColor.

##### Add SpiceColor as an interface, and let it be by color.

##### Create an instance of Curry, and print its color. However, color is actually a property common to all spices, so you can move it to the parent class.

##### Change your code so that the SpiceColor interface is added to the Spice class and inherited by Curry.

### Challenge 18

##### Create a simple data class, SpiceContainer, that holds one spice.

##### Give SpiceContainer a property, label, that is derived from the name of the spice.

##### Create some containers with spices and print out their labels.

### Challenge 19

You used object in the previous lesson and quiz.

And now that you know about enums, here's the code for Color as an enum:

**enum** **class** **Color**(val rgb: Int) {

RED(0xFF0000), GREEN(0x00FF00), BLUE(0x0000FF);

}

1. In SpiceColor, change the type of color from String to the Color class, and set the appropriate color in YellowSpiceColor.

##### ****Hint:**** The color code for yellow is YELLOW(0xFFFF00)

1. Make Spice a sealed class.

##### What is the effect of doing this?

##### Why is this useful?