# Elixir Learning Path

## Index:

1. JoE -> https://joyofelixir.com/toc.html
2. ES -> https://elixirschool.com/
3. EC -> https://elixircasts.io/
4. Video tutorials -> https://drive.google.com/open?id=1dkHgw4rCJYL_I1kk5eFCnMVG2YLUrdW2

You have to learn the given topics from all 3 of them above to get a better understanding of the topic.

## Tasks:

### Day 1:

#### Learn:

1. Introduction
2. Basic types
3. Basic operators
4. Pattern matching
5. case, cond, and if
6. Binaries, strings, and charlists

#### Submission 1: Algorithms, Pattern Matching

Write a function to convert from normal numbers to Roman Numerals.

The Romans were a clever bunch. They conquered most of Europe and ruled it for hundreds of years. They invented concrete and straight roads and even bikinis. One thing they never discovered though was the number zero. This made writing and dating extensive histories of their exploits slightly more challenging, but the system of numbers they came up with is still in use today. For example the BBC uses Roman numerals to date their programmes.

The Romans wrote numbers using letters - I, V, X, L, C, D, M. (notice these letters have lots of straight lines and are hence easy to hack into stone tablets).

```
 1  => I
10  => X
 7  => VII
```

There is no need to be able to convert numbers larger than about 3000. (The Romans themselves didn't tend to go any higher)

Wikipedia says: Modern Roman numerals ... are written by expressing each digit separately starting with the leftmost digit and skipping any digit with a value of zero.

To see this in practice, consider the example of 1990.

In Roman numerals 1990 is MCMXC:
1000=M 900=CM 90=XC
2008 is written as MMVIII:
2000=MM 8=VIII

See also: http://www.novaroma.org/via_romana/numbers.html

### Submission 2: Strings

Given a DNA strand, return its RNA complement (per RNA transcription).
Both DNA and RNA strands are a sequence of nucleotides.
The four nucleotides found in DNA are adenine (A), cytosine (C), guanine (G) and thymine (T).
The four nucleotides found in RNA are adenine (A), cytosine (C), guanine (G) and uracil (U).
Given a DNA strand, its transcribed RNA strand is formed by replacing each nucleotide with its complement:

```
G -> C
C -> G
T -> A
A -> U
```

### Submission 3: Control flow, String processing

Bob is a lackadaisical teenager. In conversation, his responses are very limited.
Bob answers 'Sure.' if you ask him a question.
He answers 'Whoa, chill out!' if you yell at him.
He answers 'Calm down, I know what I'm doing!' if you yell a question at him.
He says 'Fine. Be that way!' if you address him without actually saying anything.
He answers 'Whatever.' to anything else.
Bob's conversational partner is a purist when it comes to written communication and always follows normal rules regarding sentence punctuation in English.

---

## Day 2:

### Learn:

1. Keyword lists and maps
2. Modules and Functions
3. Recursion
4. Enumerables and streams

### Submission 1: Lists, Maps, Reduce

Given a phrase, count the occurrences of each word in that phrase.

For example for the input "olly olly in come free"

```
olly: 2
in: 1
come: 1
free: 1
```

Words are compared case-insensitively. The keys are lowercase.

### Submission 2: Recursion, Pattern Matching

Recite the lyrics to that beloved classic, that field-trip favorite: 99 Bottles of Beer on the Wall.

Note that not all verses are identical.

```
99 bottles of beer on the wall, 99 bottles of beer.
Take one down and pass it around, 98 bottles of beer on the wall.

98 bottles of beer on the wall, 98 bottles of beer.
Take one down and pass it around, 97 bottles of beer on the wall.

97 bottles of beer on the wall, 97 bottles of beer.
Take one down and pass it around, 96 bottles of beer on the wall.

96 bottles of beer on the wall, 96 bottles of beer.
Take one down and pass it around, 95 bottles of beer on the wall.

95 bottles of beer on the wall, 95 bottles of beer.
Take one down and pass it around, 94 bottles of beer on the wall.
.
.
.
5 bottles of beer on the wall, 5 bottles of beer.
Take one down and pass it around, 4 bottles of beer on the wall.

4 bottles of beer on the wall, 4 bottles of beer.
Take one down and pass it around, 3 bottles of beer on the wall.

3 bottles of beer on the wall, 3 bottles of beer.
Take one down and pass it around, 2 bottles of beer on the wall.

2 bottles of beer on the wall, 2 bottles of beer.
Take one down and pass it around, 1 bottle of beer on the wall.

1 bottle of beer on the wall, 1 bottle of beer.
Take it down and pass it around, no more bottles of beer on the wall.

No more bottles of beer on the wall, no more bottles of beer.
Go to the store and buy some more, 99 bottles of beer on the wall.
```

### For bonus points

These are some additional things you could try:

Remove as much duplication as you possibly can.

Optimize for readability, even if it means introducing duplication.

If you've removed all the duplication, do you have a lot of conditionals? Try replacing the conditionals with polymorphism, if it applies in this language. How readable is it?

Then please share your thoughts in a comment on the submission. Did this experiment make the code better? Worse? Did you learn anything from it?

## Submission 3: Lists, Enumeration, Recursion

Implement basic list operations.

In functional languages list operations like length, map, and reduce are very common. Implement a series of basic list operations, without using existing functions.

## Day 3:

### Learn:

1. Processes
2. IO and the file system
3. alias, require, and import
4. Module attributes
5. Structs

### Submission 1: String Processing

Create an implementation of the rotational cipher, also sometimes called the Caesar cipher.

The Caesar cipher is a simple shift cipher that relies on transposing all the letters in the alphabet using an integer key between 0 and 26. Using a key of 0 or 26 will always yield the same output due to modular arithmetic. The letter is shifted for as many values as the value of the key.

The general notation for rotational ciphers is ROT + <key>. The most commonly used rotational cipher is ROT13.

A ROT13 on the Latin alphabet would be as follows:

Plain:  abcdefghijklmnopqrstuvwxyz
Cipher: nopqrstuvwxyzabcdefghijklm
It is stronger than the Atbash cipher because it has 27 possible keys, and 25 usable keys.

Ciphertext is written out in the same formatting as the input including spaces and punctuation.

Examples
ROT5 omg gives trl
ROT0 c gives c
ROT26 Cool gives Cool
ROT13 The quick brown fox jumps over the lazy dog. gives Gur dhvpx oebja sbk whzcf bire gur ynml qbt.
ROT13 Gur dhvpx oebja sbk whzcf bire gur ynml qbt. gives The quick brown fox jumps over the lazy dog.

### Submission 2: Collections

Implement the keep and discard operation on collections. Given a collection and a predicate on the collection's elements, keep returns a new collection containing those elements where the predicate is true, while discard returns a new collection containing those elements where the predicate is false.

For example, given the collection of numbers:

1, 2, 3, 4, 5
And the predicate:

Is the number even?
Then your keep operation should produce:

2, 4
While your discard operation should produce:

1, 3, 5
Note that the union of keep and discard is all the elements.

The functions may be called keep and discard, or they may need different names in order to not clash with existing functions or concepts in your language.

*apply* will let you pass arguments to a function, as will fun.(args)

## Submission 3: Pattern Matching

Given an age in seconds, calculate how old someone would be on:

Earth: orbital period 365.25 Earth days, or 31557600 seconds
Mercury: orbital period 0.2408467 Earth years
Venus: orbital period 0.61519726 Earth years
Mars: orbital period 1.8808158 Earth years
Jupiter: orbital period 11.862615 Earth years
Saturn: orbital period 29.447498 Earth years
Uranus: orbital period 84.016846 Earth years
Neptune: orbital period 164.79132 Earth years
So if you were told someone was 1,000,000,000 seconds old, you should be able to say that they're 31.69 Earth-years old.

If you're wondering why Pluto didn't make the cut, go watch this youtube video.

## Submission 4: String Processing

Given a string of digits, output all the contiguous substrings of length n in that string in the order that they appear.

For example, the string "49142" has the following 3-digit series:

"491"
"914"
"142"
And the following 4-digit series:

"4914"
"9142"
And if you ask for a 6-digit series from a 5-digit string, you deserve whatever you get.

## Submission 5: Conditionals

Convert a number to a string, the contents of which depend on the number's factors.

If the number has 3 as a factor, output 'Pling'.
If the number has 5 as a factor, output 'Plang'.
If the number has 7 as a factor, output 'Plong'.
If the number does not have 3, 5, or 7 as a factor, just pass the number's digits straight through.

Examples
28's factors are 1, 2, 4, 7, 14, 28.
In raindrop-speak, this would be a simple "Plong".

The 30's factors are 1, 2, 3, 5, 6, 10, 15, 30.
In raindrop-speak, this would be a "PlingPlang".

34 has four factors: 1, 2, 17, and 34.
In raindrop-speak, this would be "34".

## Submission 6: Lists, Enumeration

Given two lists determine if the first list is contained within the second list, if the second list is contained within the first list, if both lists are contained within each other or if none of these are true.

Specifically, a list A is a sublist of list B if by dropping 0 or more elements from the front of B and 0 or more elements from the back of B you get a list that's completely equal to A.

Examples:

A = [1, 2, 3], B = [1, 2, 3, 4, 5], A is a sublist of B
A = [3, 4, 5], B = [1, 2, 3, 4, 5], A is a sublist of B
A = [3, 4], B = [1, 2, 3, 4, 5], A is a sublist of B
A = [1, 2, 3], B = [1, 2, 3], A is equal to B
A = [1, 2, 3, 4, 5], B = [2, 3, 4], A is a superlist of B
A = [1, 2, 4], B = [1, 2, 3, 4, 5], A is not a superlist of, sublist of or equal to B

## Submission 7: Reduce

Given a word, compute the scrabble score for that word.

Letter Values
You'll need these:

| Letter | Value |
|---|---|
| A, E, I, O, U, L, N, R, S, T | 1 |
| D, G | 2 |
| B, C, M, P | 3 |
| F, H, V, W, Y | 4 |
| K | 5 |
| J, X | 8 |
| Q, Z | 10 |

Examples
"cabbage" should be scored as worth 14 points:

3 points for C
1 point for A, twice
3 points for B, twice
2 points for G
1 point for E
And to total:

3 + 2*1 + 2*3 + 2 + 1
= 3 + 2 + 6 + 3
= 5 + 9
= 14

## Day 4:

### Learn:

1. Protocols
2. Comprehensions
3. Sigils
4. try, catch, and rescue
5. Typespecs and behaviours
6. Debugging
7. Erlang libraries

### Submission 1: Algorithms, Recursion, Math

Given a number n, determine what the nth prime is.
By listing the first six prime numbers: 2, 3, 5, 7, 11, and 13, we can see that the 6th prime is 13.
If your language provides methods in the standard library to deal with prime numbers, pretend they don't exist and implement them yourself.

### Submission 2: Reduce, Algorithms

Determine if a word or phrase is an isogram.
An isogram (also known as a "non pattern word") is a word or phrase without a repeating letter, however spaces and hyphens are allowed to appear multiple times.
Examples of isograms:

- lumberjacks
- background
- downstream
- six-year-old

The word isograms, however, is not an isogram, because the s repeats.

### Submission 3: Maps

Given students' names along with the grade that they are in, create a roster for the school.
In the end, you should be able to:

1. Add a student's name to the roster for a grade
    - "Add Jim to grade 2."
    - "OK."
2. Get a list of all students enrolled in a grade
    - "Which students are in grade 2?"
    - "We've only got Jim just now."
3. Get a sorted list of all students in all grades. Grades should sort as 1, 2, 3, etc., and students within a grade should be sorted alphabetically by name.
    - "Who all is enrolled in school right now?"
    - "Grade 1: Anna, Barb, and Charlie. Grade 2: Alex, Peter, and Zoe. Grade 3…"

Note that all our students only have one name. (It's a small town, what do you want?)

Did you get code clean? As you're working in a language with mutable data structures and if your implementation allows outside code to mutate the school's internal DB directly, see if you can prevent this.

### Submission 4: Lists, Recursion

Take a nested list and return a single flattened list with all values except nil/null.
The challenge is to write a function that accepts an arbitrarily-deep nested list-like structure and returns a flattened structure without any nil/null values.
For Example
input: [1,[2,3,null,4],[null],5]
output: [1,2,3,4,5]

### Submission 5:  Algorithms

Given a year, report if it is a leap year.
The tricky thing here is that a leap year in the Gregorian calendar occurs:

```
on every year that is evenly divisible by 4
  except every year that is evenly divisible by 100
    unless the year is also evenly divisible by 400
```

For example, 1997 is not a leap year, but 1996 is. 1900 is not a leap year, but 2000 is.
If your language provides a method in the standard library that does this look-up, pretend it doesn't exist and implement it yourself.
Notes
Though our exercise adopts some very simple rules, there is more to learn!
For a delightful, four minute explanation of the whole leap year phenomenon, go watch this youtube video.

### Submission 6:  Filter, Enumeration

Given a word and a list of possible anagrams, select the correct sublist.

Given "listen" and a list of candidates like "enlists" "google" "inlets" "banana" the program should return a list containing "inlets".

---

## Day 5:

### Learn:

1. Introduction to Mix
2. Agent
3. GenServer
4. Supervisor and Application
5. Dynamic supervisors

### Submission 1:  Concurrency, OTP

Count the frequency of letters in a list of strings using parallel computation.

Parallelism is about doing things in parallel that can also be done sequentially. A common example is counting the frequency of letters. Create a function that returns the total frequency of each letter in a list of strings and that employs parallelism.

### Submission 2:  Calendar, Time

Calculate the moment when someone has lived for 10^9 seconds.
A gigasecond is 10^9 (1,000,000,000) seconds.

## Submission 3:  Algorithms, Structs

Given the position of two queens on a chess board, indicate whether or not they are positioned so that they can attack each other.

In the game of chess, a queen can attack pieces which are on the same row, column, or diagonal.

A chessboard can be represented by an 8 by 8 array.

So if you're told the white queen is at (2, 3) and the black queen at (5, 6), then you'd know you've got a set-up like so:

```
_ _ _ _ _ _ _ _
_ _ _ _ _ _ _ _
_ _ _ W _ _ _ _
_ _ _ _ _ _ _ _
_ _ _ _ _ _ _ _
_ _ _ _ _ _ B _
_ _ _ _ _ _ _ _
_ _ _ _ _ _ _ _
```

You'd also be able to answer whether the queens can attack each other. In this case, that answer would be yes, they can, because both pieces share a diagonal.

## Submission 4:  String Processing, Enumeration

Given a number, determine whether or not it is valid per the Luhn formula.

The Luhn algorithm is a simple checksum formula used to validate a variety of identification numbers, such as credit card numbers and Canadian Social Insurance Numbers.

The task is to check if a given string is valid.

Validating a Number
Strings of length 1 or less are not valid. Spaces are allowed in the input, but they should be stripped before checking. All other non-digit characters are disallowed.

Example 1: valid credit card number
4539 1488 0343 6467
The first step of the Luhn algorithm is to double every second digit, starting from the right. We will be doubling

4_3_ 1_8_ 0_4_ 6_6_
If doubling the number results in a number greater than 9 then subtract 9 from the product. The results of our doubling:

8569 2478 0383 3437
Then sum all of the digits:

8+5+6+9+2+4+7+8+0+3+8+3+3+4+3+7 = 80
If the sum is evenly divisible by 10, then the number is valid. This number is valid!

Example 2: invalid credit card number
8273 1232 7352 0569
Double the second digits, starting from the right

7253 2262 5312 0539
Sum the digits

7+2+5+3+2+2+6+2+5+3+1+2+0+5+3+9 = 57
57 is not evenly divisible by 10, so this number is not valid.

## Submission 5:  Recursion, Pattern Matching, Math

Compute the prime factors of a given natural number.

A prime number is only evenly divisible by itself and 1.

Note that 1 is not a prime number.

Example
What are the prime factors of 60?

Our first divisor is 2. 2 goes into 60, leaving 30.
2 goes into 30, leaving 15.
2 doesn't go cleanly into 15. So let's move on to our next divisor, 3.
3 goes cleanly into 15, leaving 5.
3 does not go cleanly into 5. The next possible factor is 4.
4 does not go cleanly into 5. The next possible factor is 5.
5 does go cleanly into 5.
We're left only with 1, so now, we're done.
Our successful divisors in that computation represent the list of prime factors of 60: 2, 2, 3, and 5.

You can check this yourself:

2 * 2 * 3 * 5
= 4 * 15
= 60
Success!