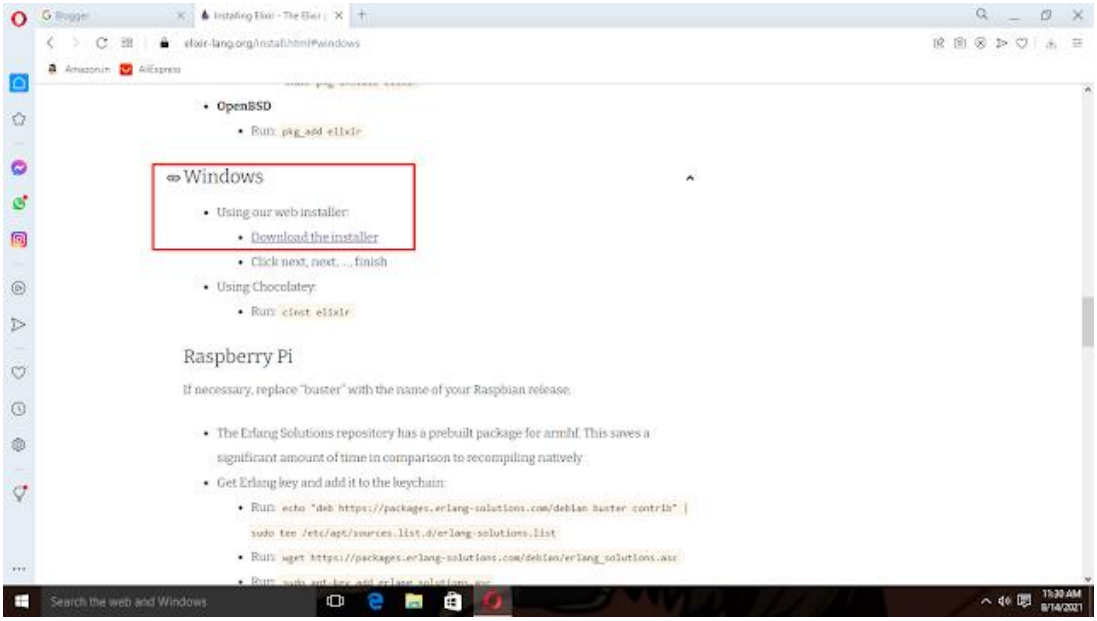
Упражнение 1 - ЕФП

I. Инсталиране - основни стъпки

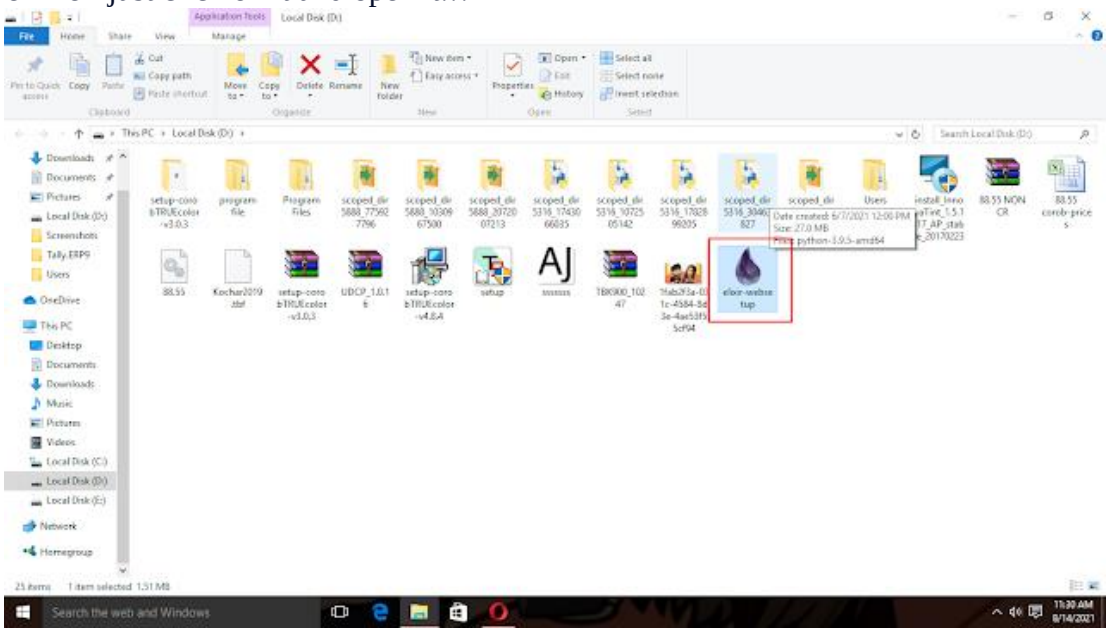
1) https://elixir-lang.org/install.html#windows
2) https://www.youtube.com/watch?v=antnsMgA4Ro&t=100s

How to download and install Elixir on Windows 10

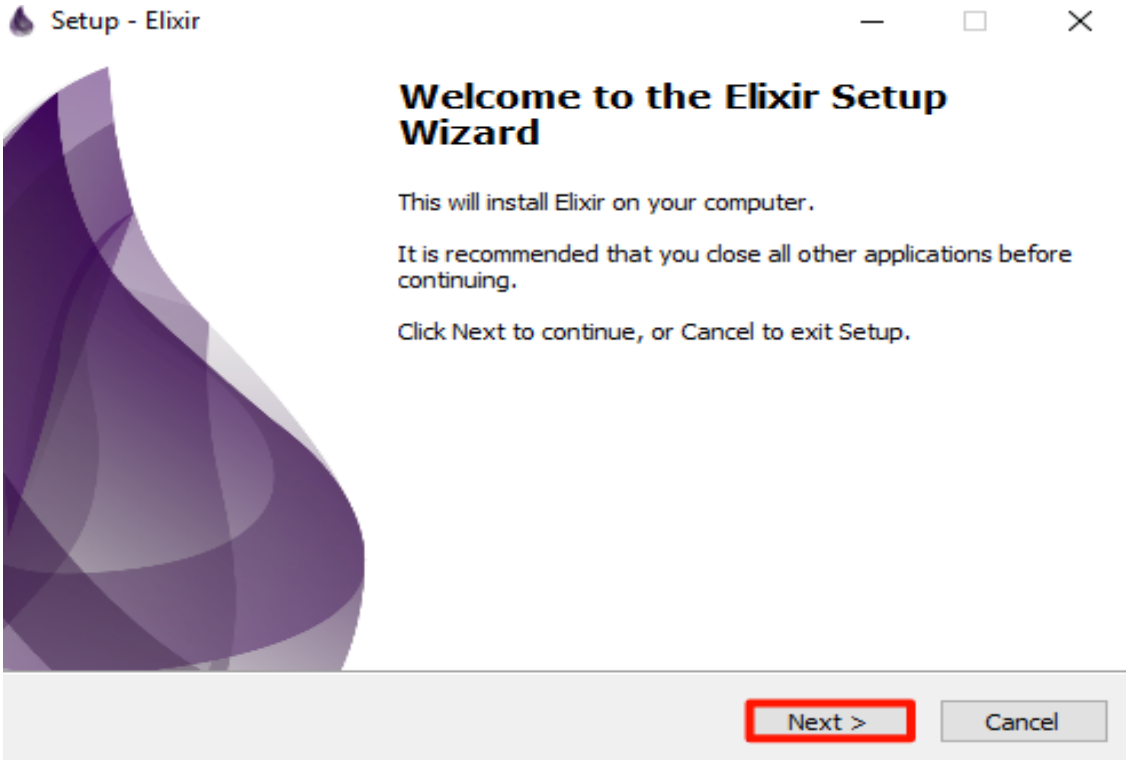
1. First of all, Go to **Elixir downloading page** then scroll down page and click on Elixir Windows Installer.



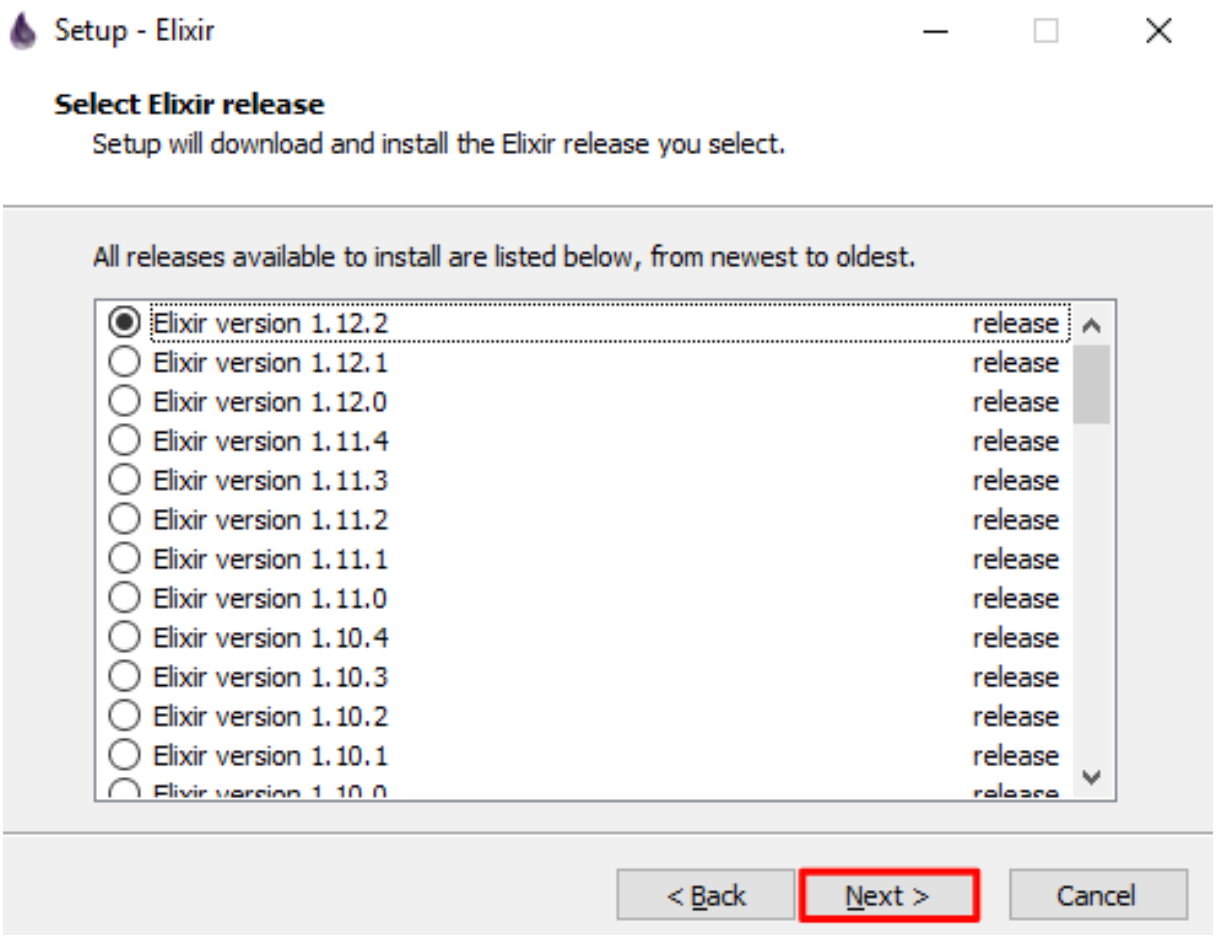
2. After clicking on this, it automatically start downloading on your pc. After downloading Elixir installer file - just click on it and open it.!!



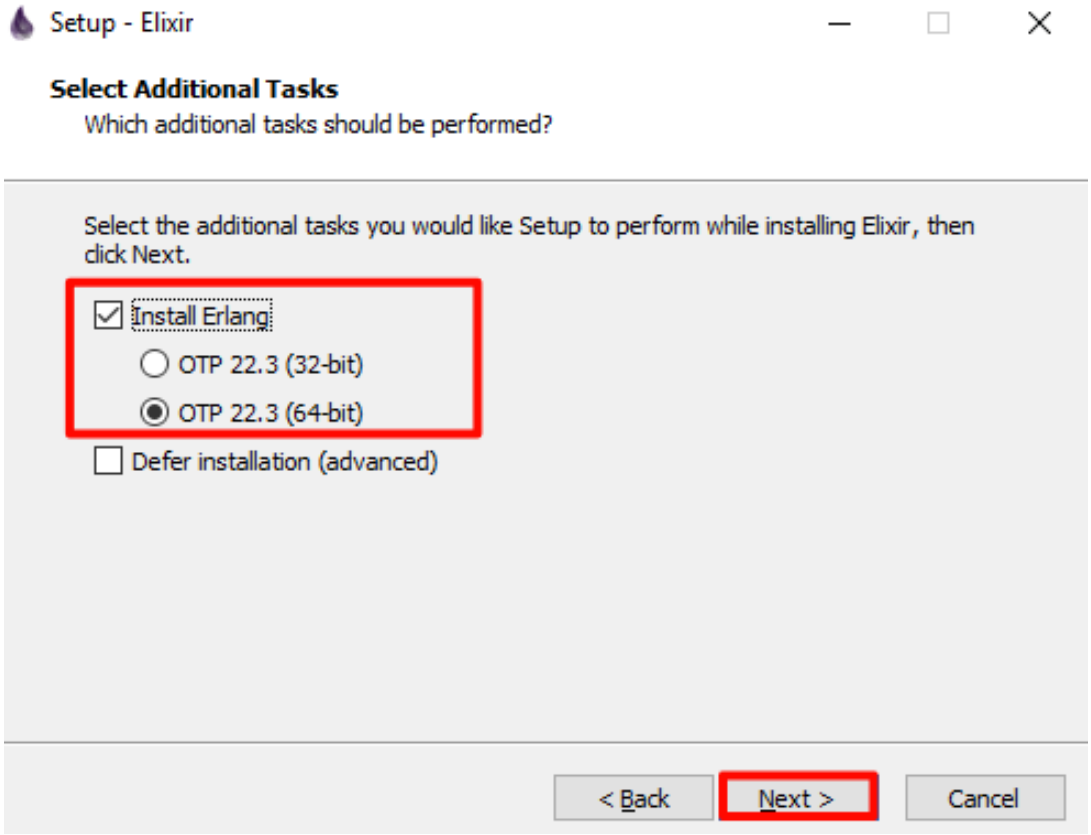
3. Now, the elixir setup wizard will be open on your computer screen - just click on **Next** to continue installation.



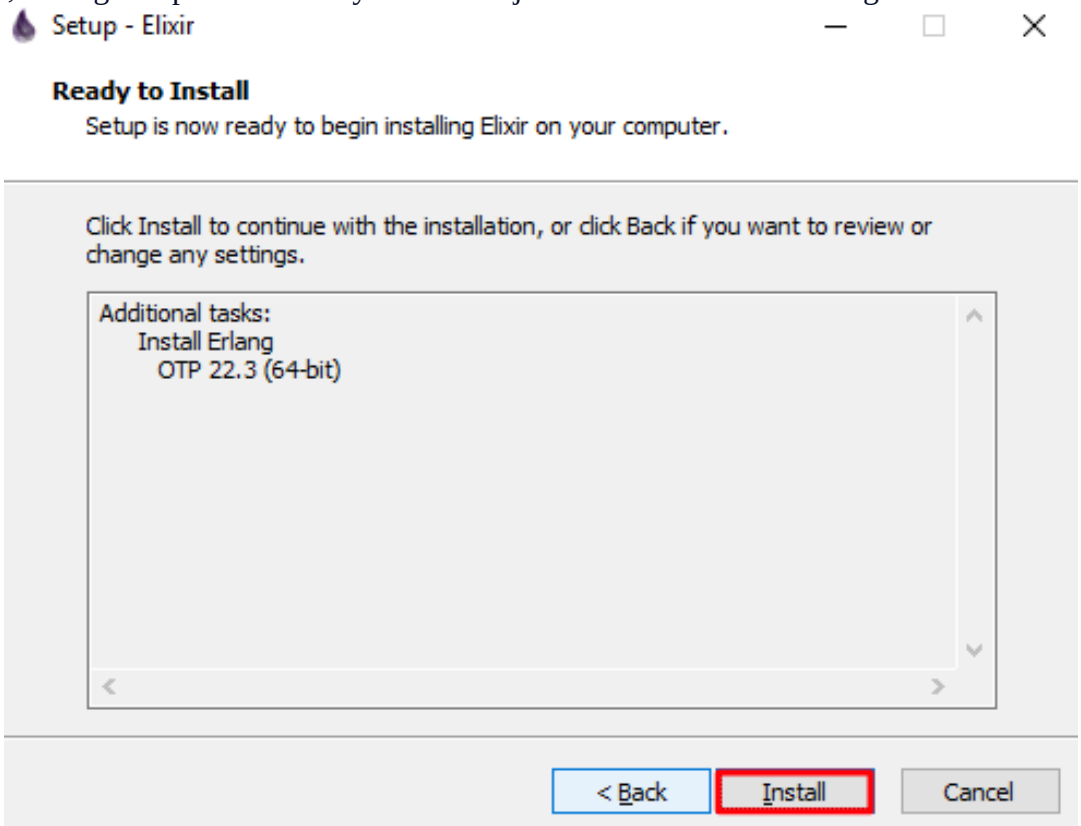
4. Here, All Elixir releases to install are listed - Choose one of them and click on **Next** to continue.



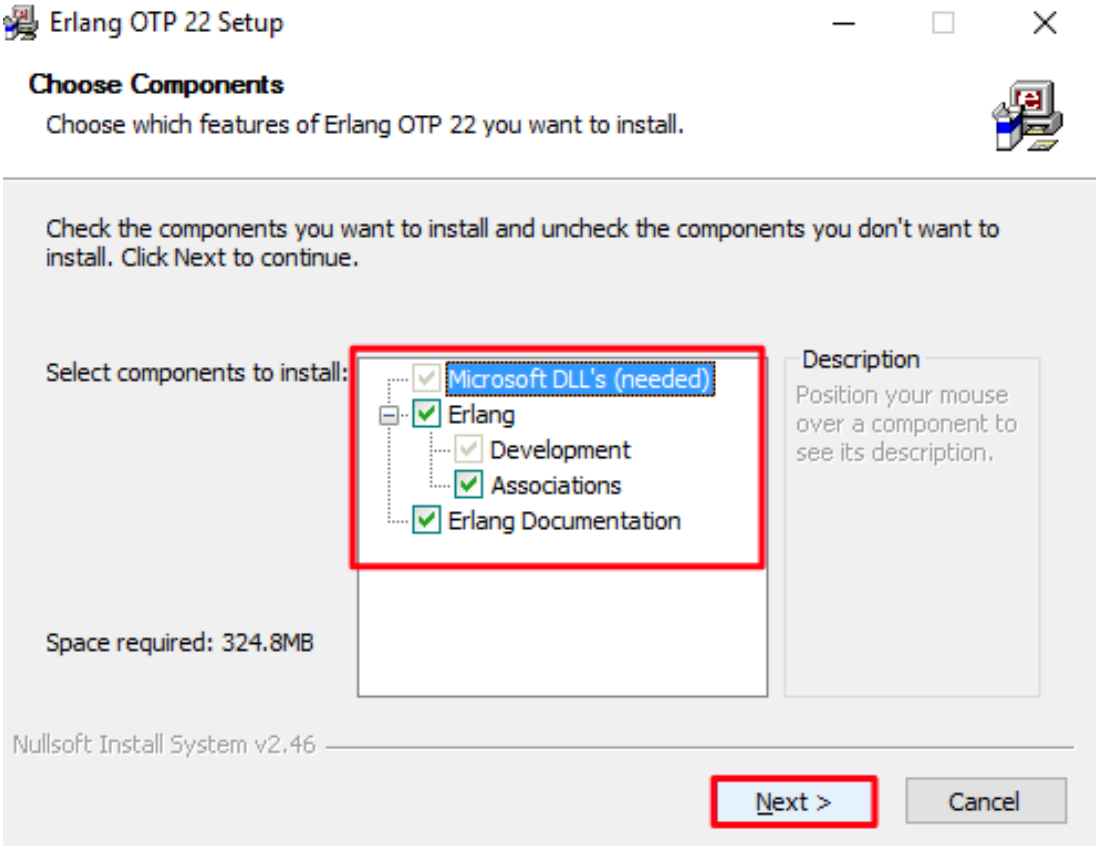
5. In this step, you will be asked choice for install Erlang or defer installation - just go with install Erlang and click on **Next** to continue.



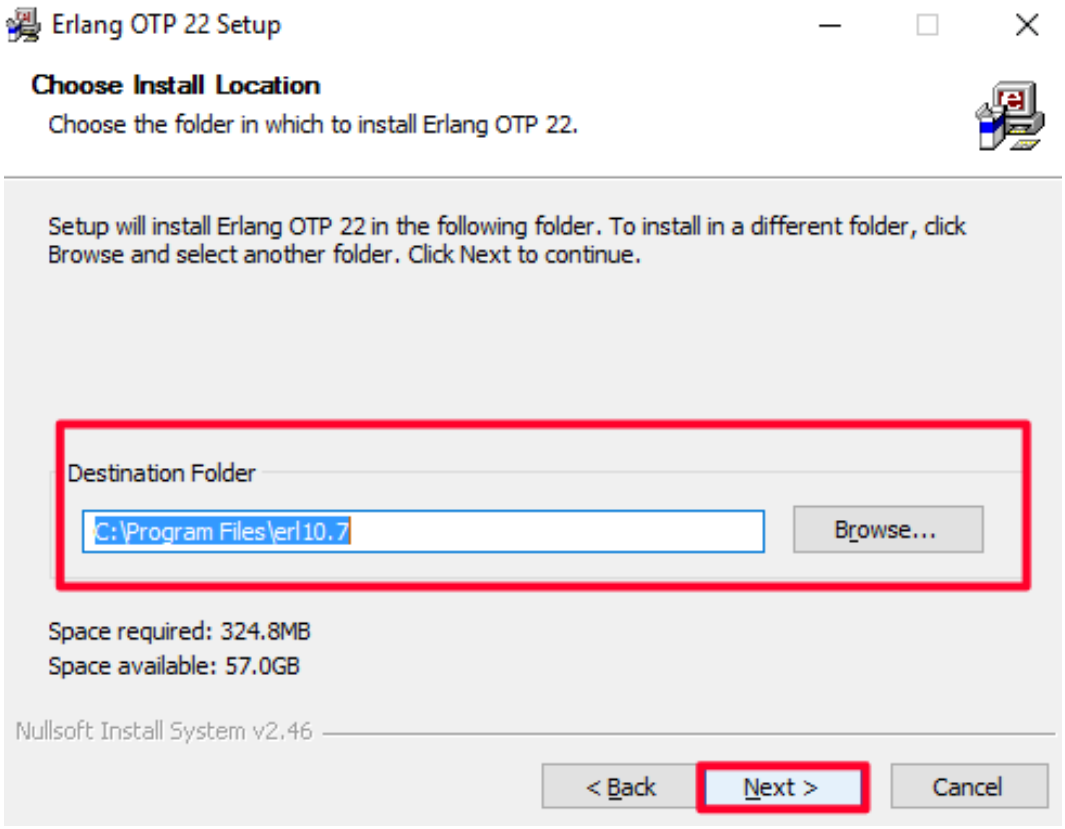
6. Now, Erlang component is ready to install - just click on "**Install**" to begin Elixir installation.



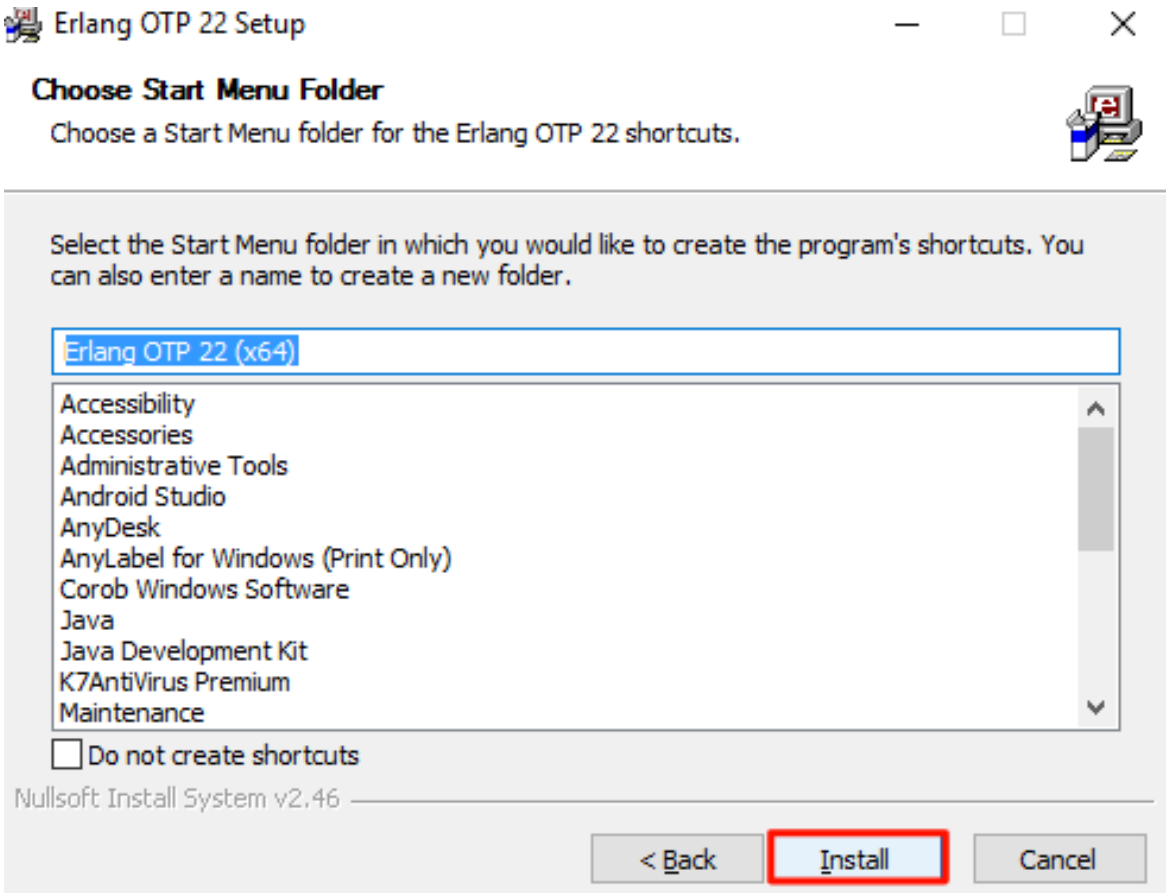
7. After installing Erlang you will also need to install Microsoft DDL's - for this, select this component and click on **Next** to continue.



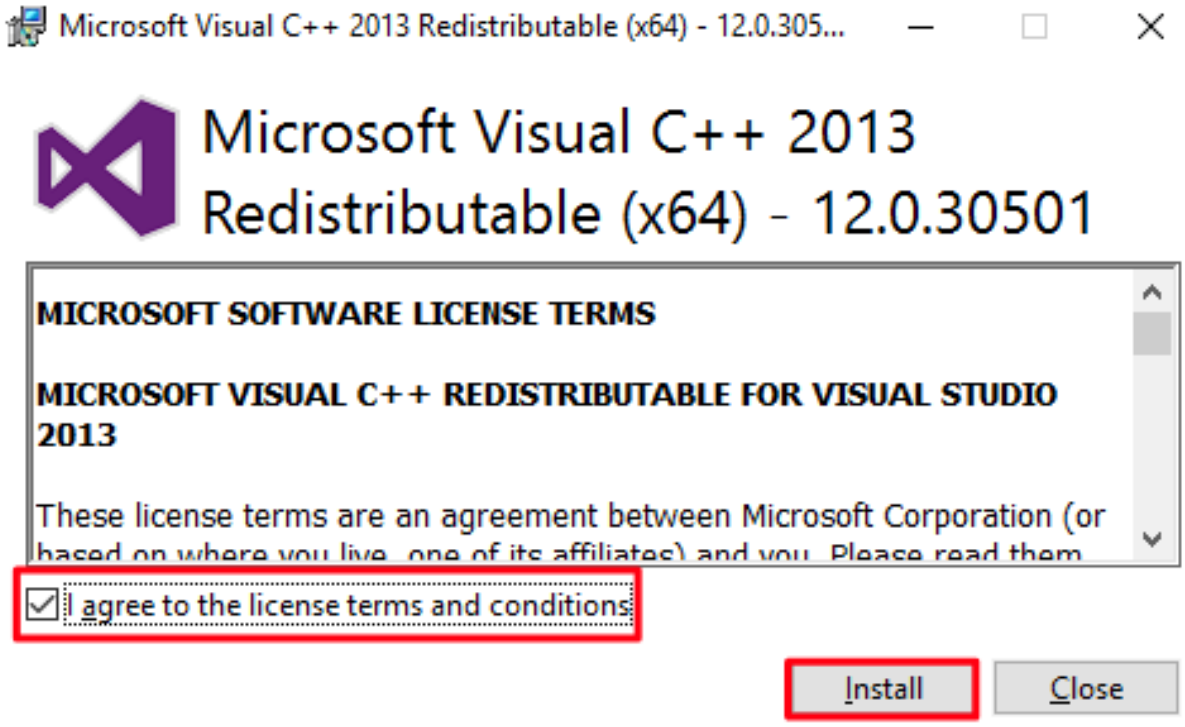
8. Now, choose the installation location for Erlang OTP 22 - browse the location and click on **Next**.



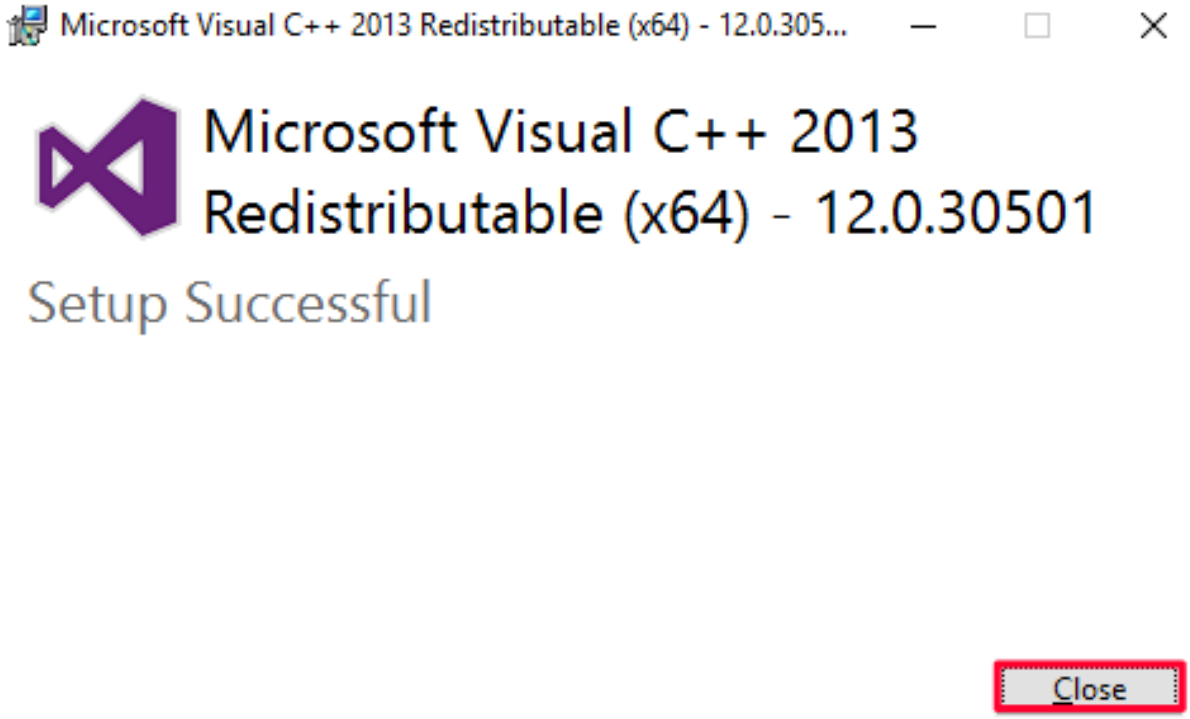
9. Here, choose the start menu folder for the Erlang OTP shortcut. After choosing start menu folder, click on **Install** to continue.



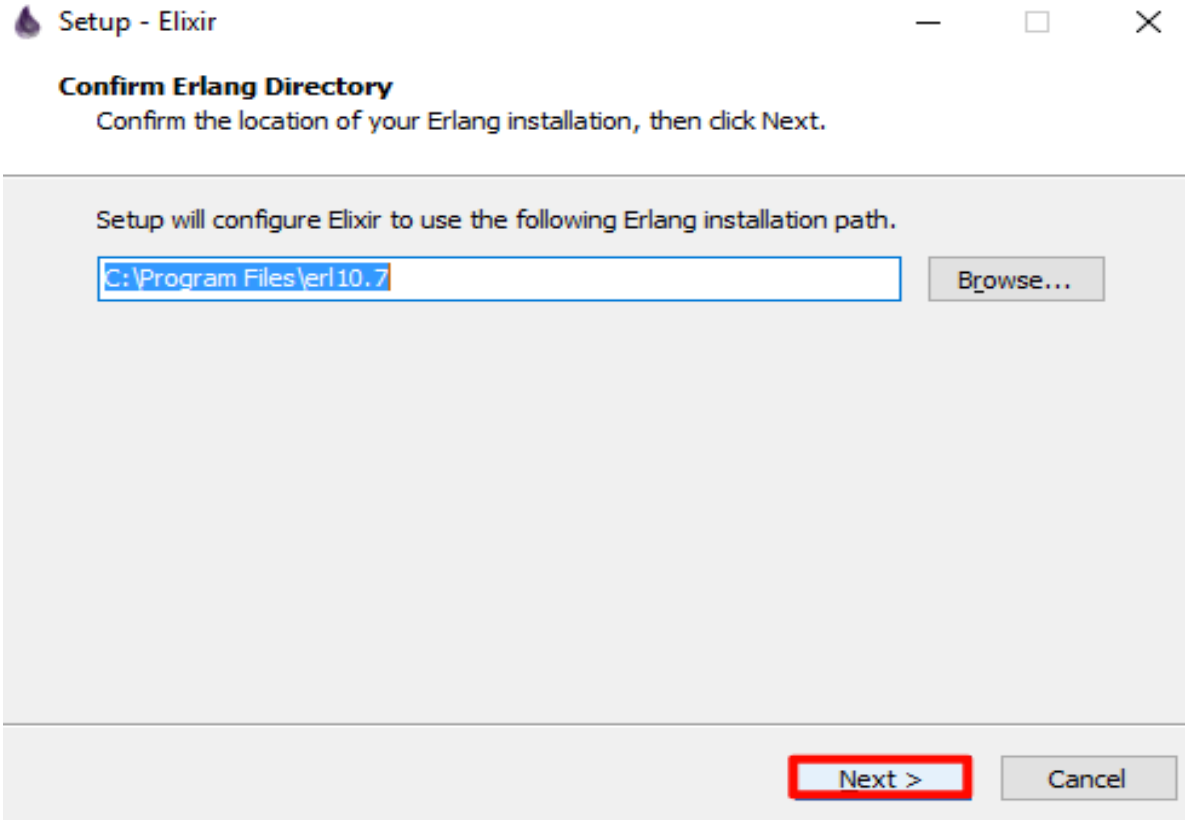
10. After this, Microsoft C++ 2013 Redistributable (x64) is ready to install - just Check out "**I accept license agreement**" and click on **Install** to install this component.



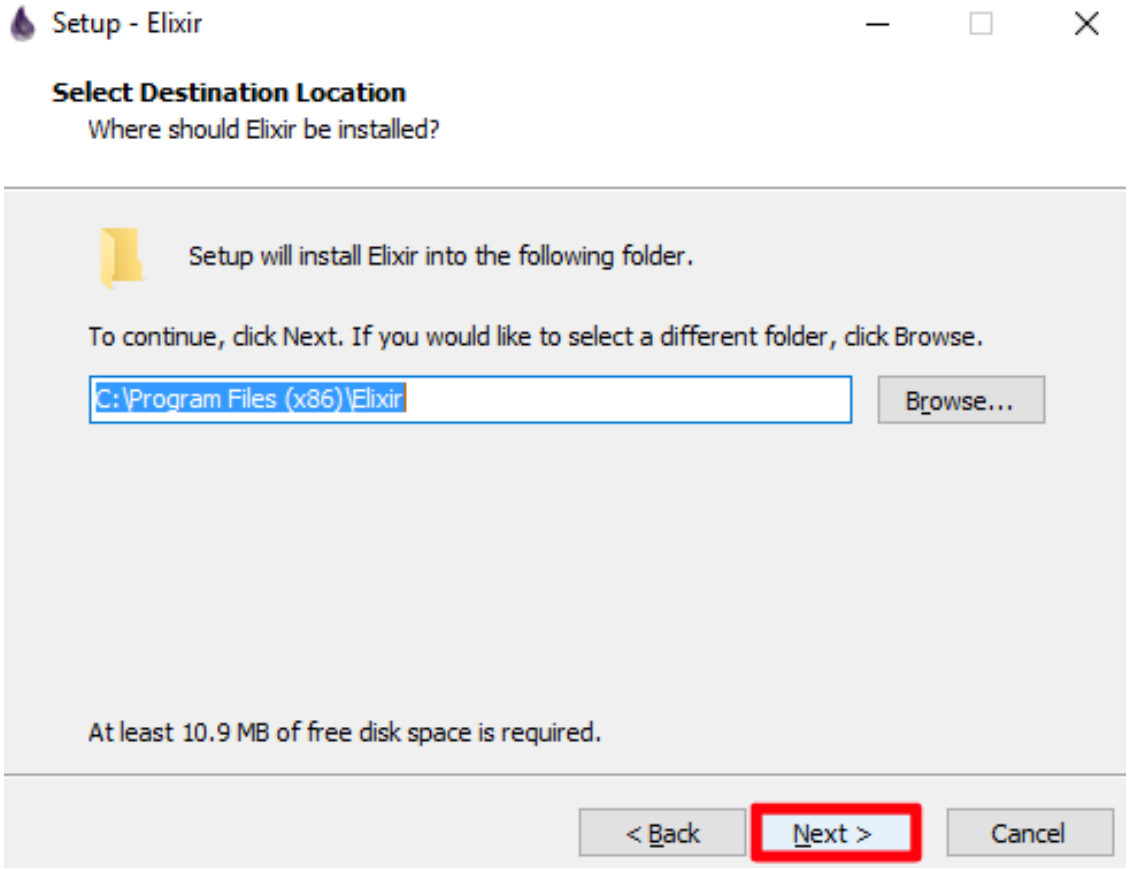
After doing this, it will take few seconds to install this. After installing Microsoft C++ 2013 Redistributable (x64) - just click on **Close**.



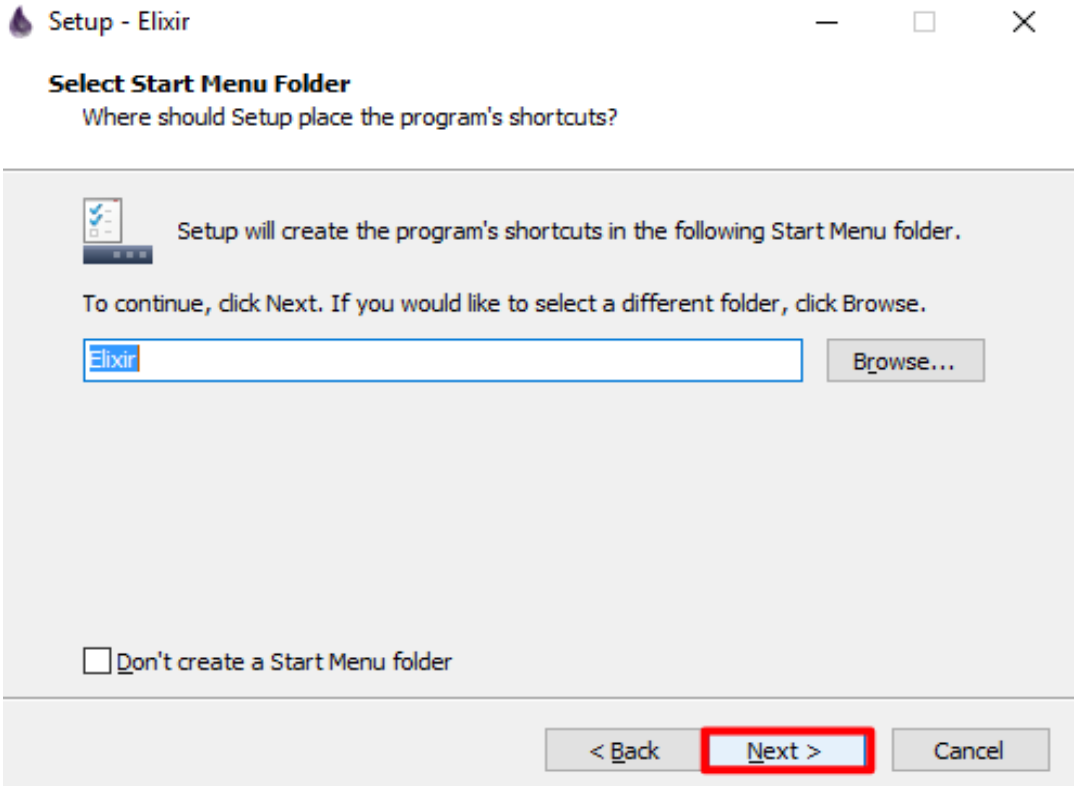
11. Now, configure the elixir to use following Erlang installation path - just go with system by default location and click on **Next** to continue Elixir installation.



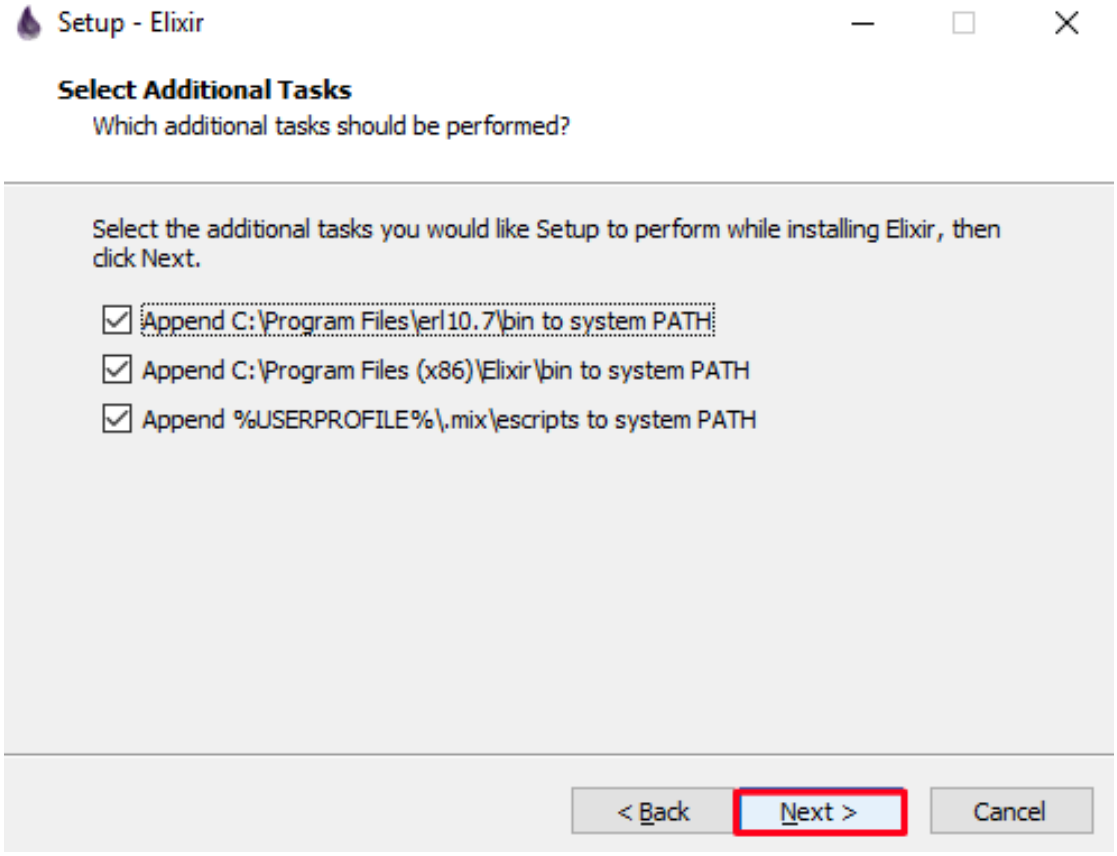
12. After doing this, you need to select the destination path where you want to install Elixir. After selecting the destination location, click on **Next** to continue.



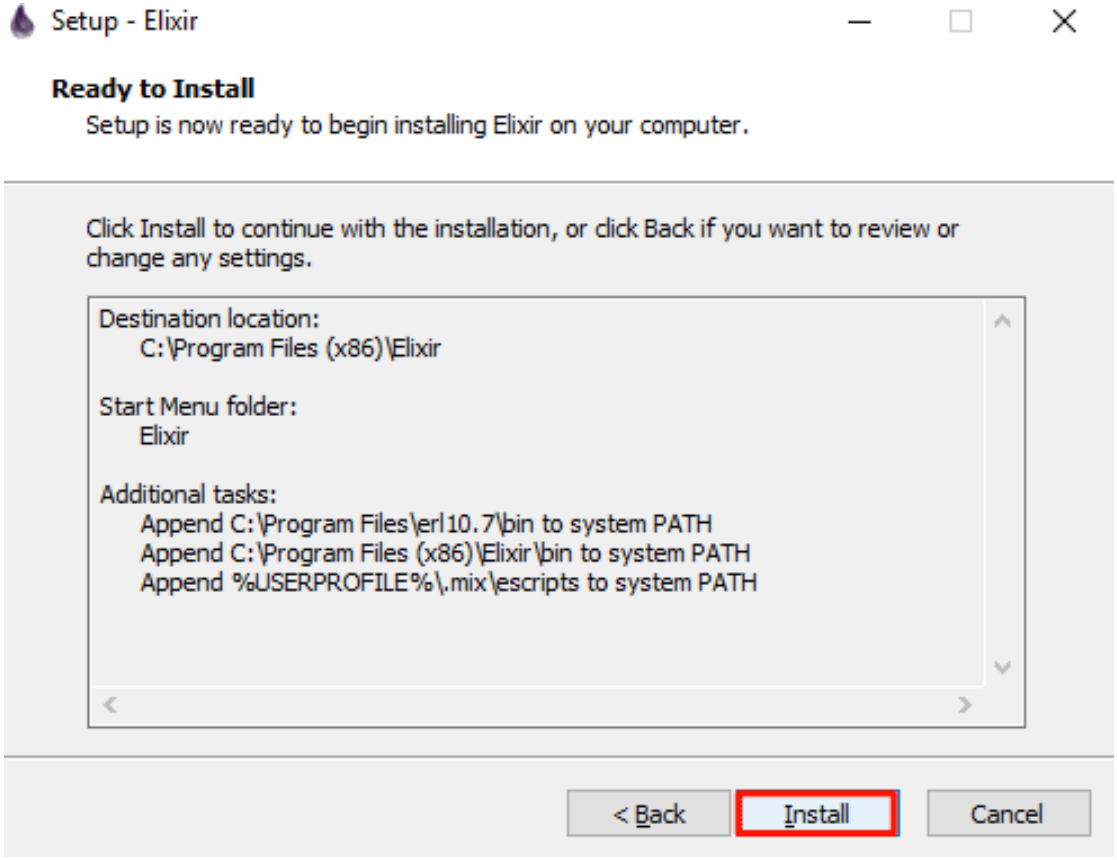
13. Now, select the start menu folder where you want to setup Elixir shortcut. After choosing this, just click on **Next** to continue.



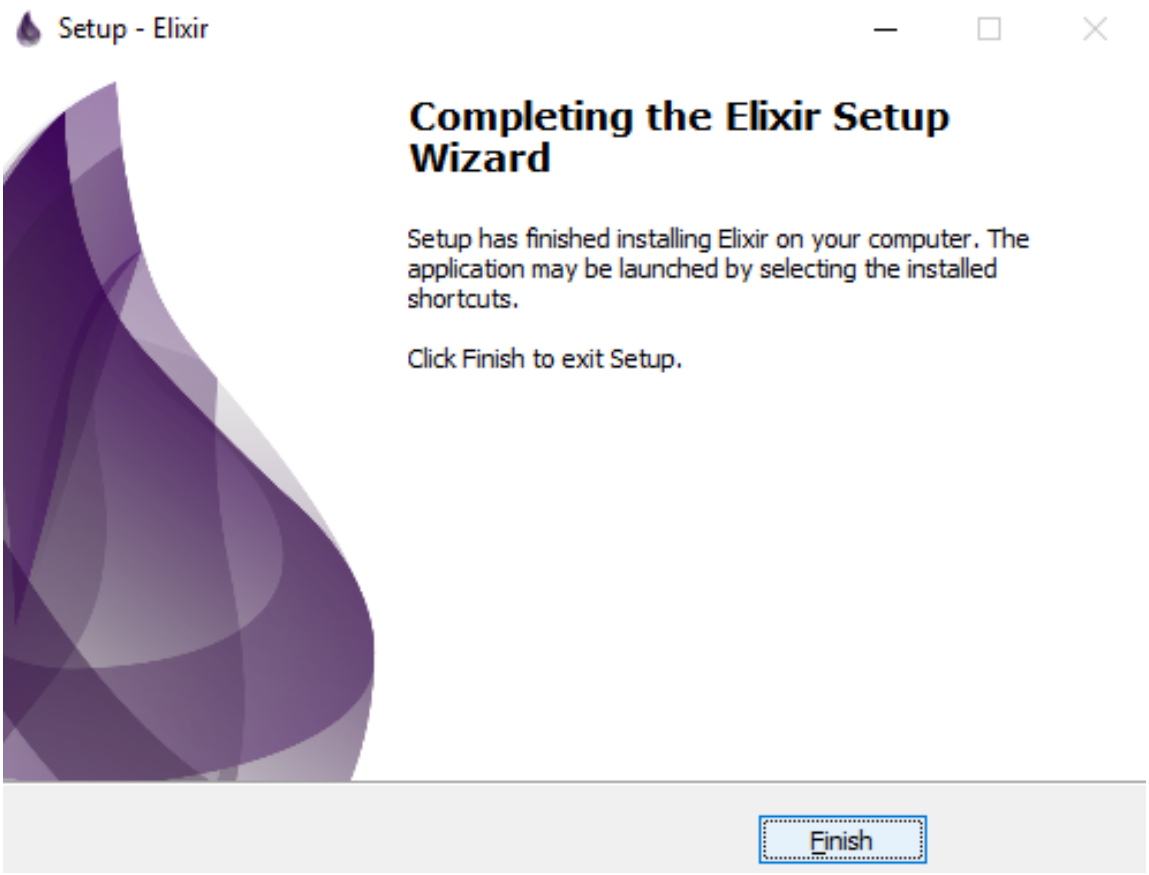
14. Here, select the additional task that you would like to setup to perform while install Elixir then click **Next** to continue.



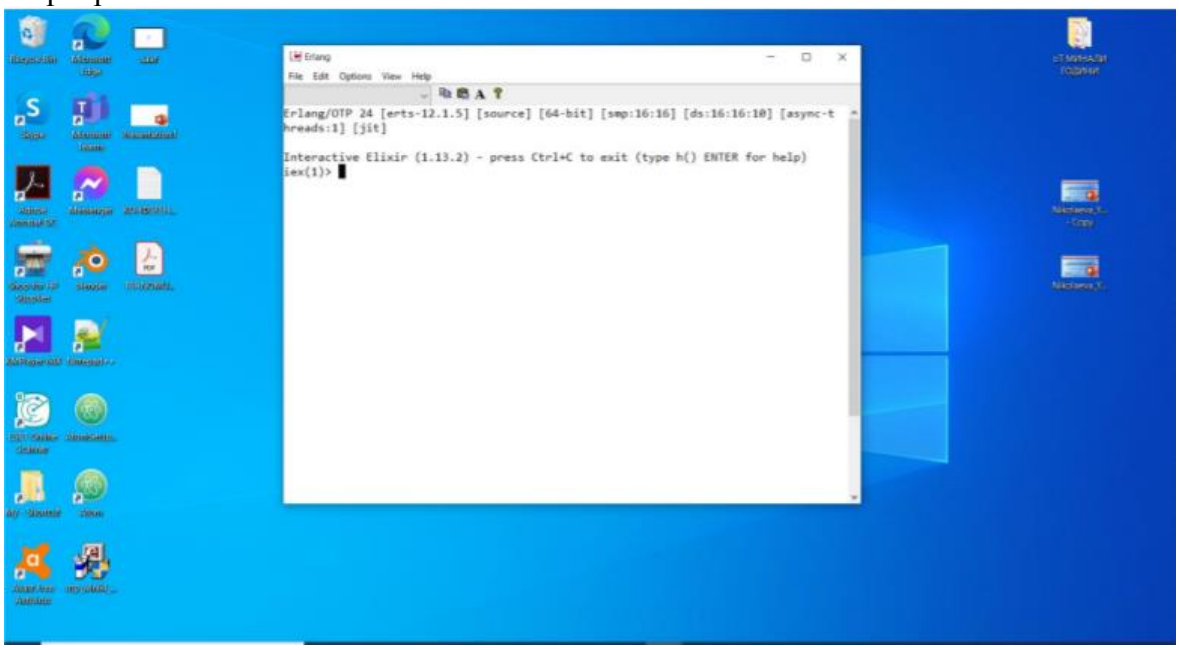
15. Elixir is ready to install on your Windows PC - just click on **Install** to begin installation process.



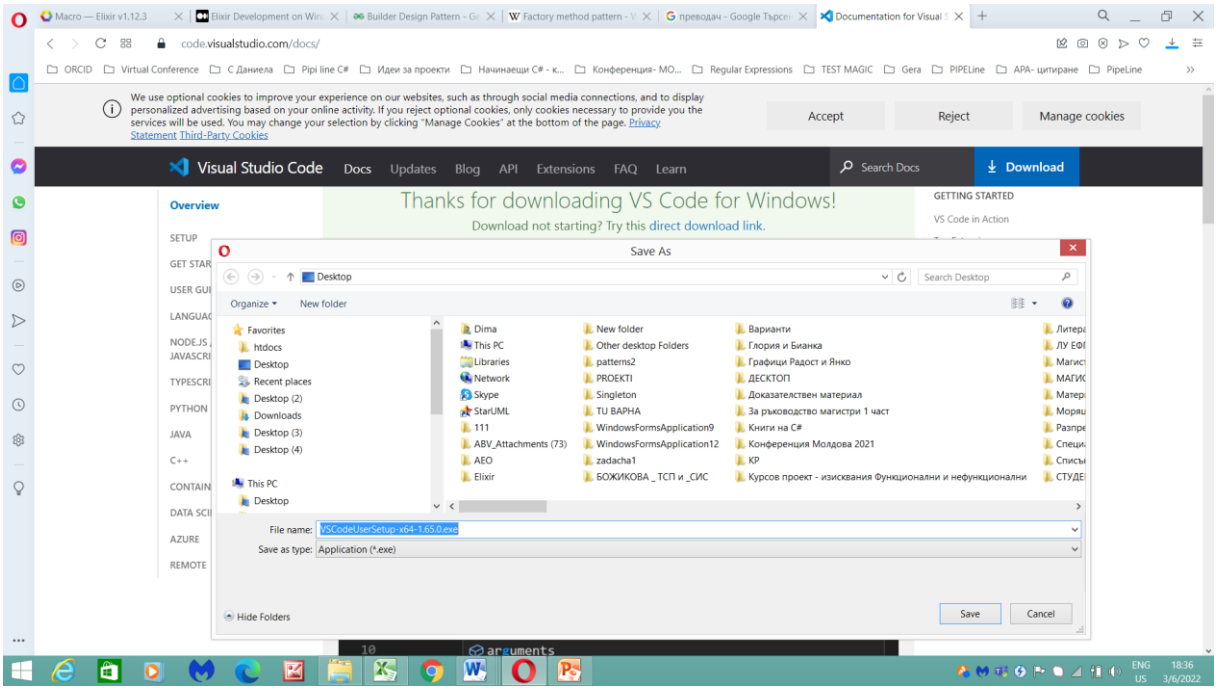
Wait few seconds until you get this confirmation;



3) Стартиране на Elixir



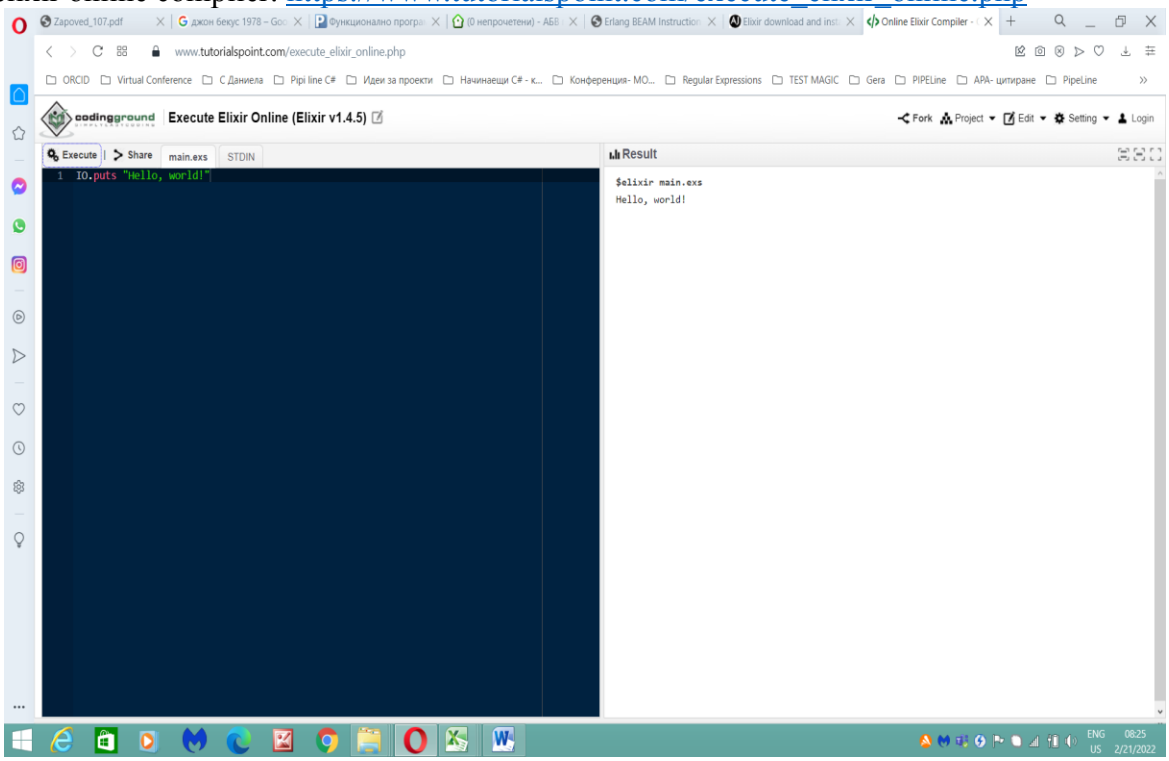
4) Инсталиране на Visual Studio Code: https://code.visualstudio.com/download



5) Инсталиране на добавки за програмният език, тъй като Elixir не е добавен в инсталацията.

...

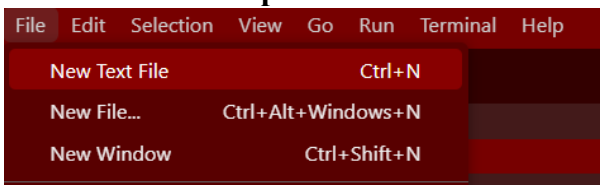
4) elixir online compiler: https://www.tutorialspoint.com/execute_elixir_online.php



II. Създаване на макроси. Компилиране.

Задача 1. Създайте вашата първа програма „hello world“.

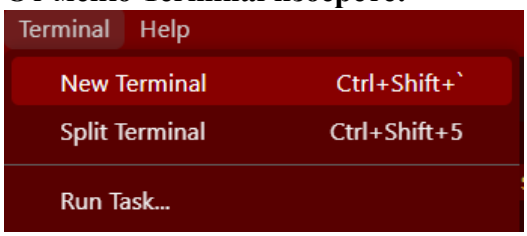
1. **От меню File изберете New Text File**



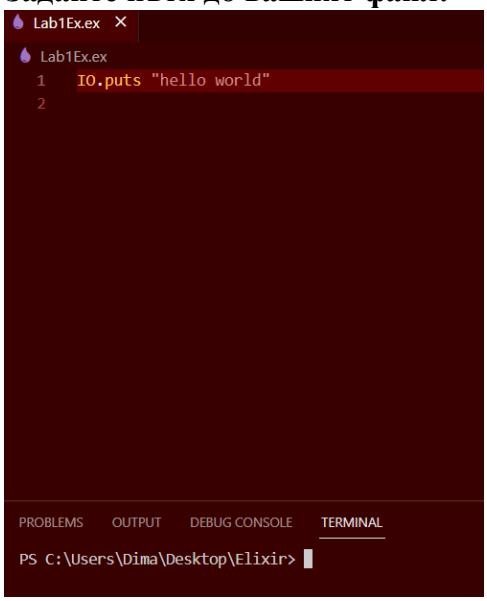
2. **Запишете следния код във файла:**

```
IO.puts "hello world"
```

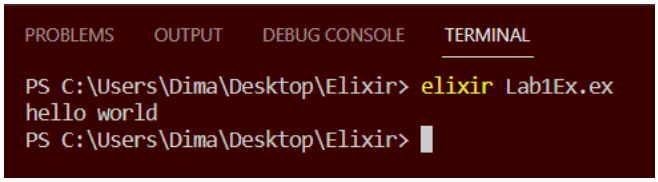
3. **Запазете файла с име: Lab1Ex. Разширението се добавя автоматично *.ex**
4. **От меню Terminal изберете:**



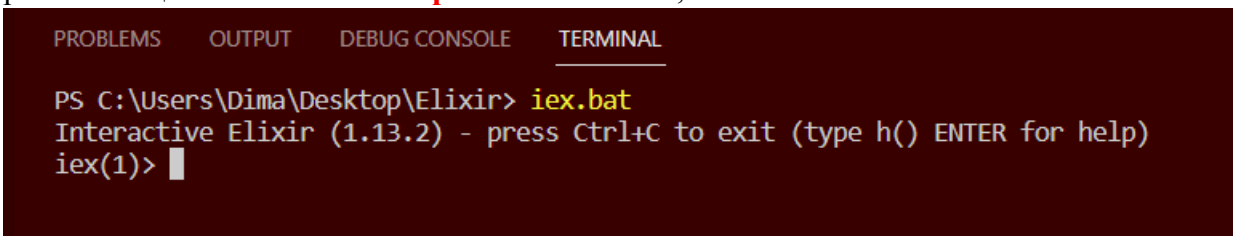
5. **Задайте пътя до вашият файл:**



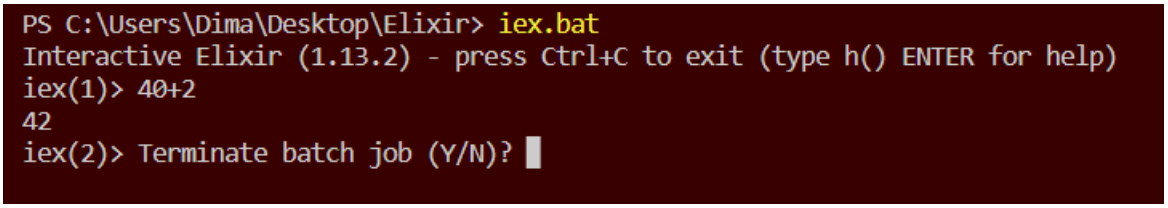
6. **За да изпълните файла запишете в терминала: elixir името_на_файла.ex**



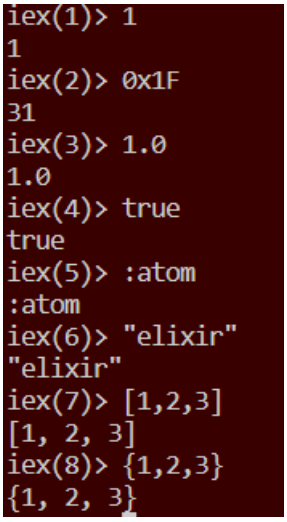
Задача 2. Създайте вашата първа програма, като използвате интерактивния режим за работа. За целта **запишете в терминала - iex.bat**, ако сте на Windows.



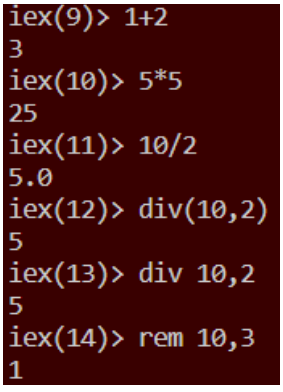
А изход от този режим **Ctrl+C**



Задача 3. Основни типове:



Задача 4. Основна аритметика:



Нотации за бърз достъп за въвеждане

```
iex(15)> 0b1010
10
iex(16)> 0o777
511
iex(17)> 0x1F
31
iex(18)> 1.0
1.0
iex(19)> 1.0e-10
1.0e-10
```

Закръгляне

```
iex(20)> round 3.58
4
iex(21)> trunc 3.58
3
```

Задача 5. Булеви стойности

```
iex(22)> true
true
iex(23)> true==false
false
```

```
iex(24)> is_boolean(true)
true
iex(25)> is_boolean(1)
false
```

Задача 6. Атоми

```
iex(1)> :hello
:hello
iex(2)> :hello==:world
false
```

```
iex(3)> true==:true
true
iex(4)> is_atom(false)
true
iex(5)> is_boolean(:false)
true
```

Задача 7. Низ

Клавишни комбинации за международни знаци

https://support.microsoft.com/bg-bg/topic/клавишни-комбинации-за-международни-знаци-108fa0c1-fb8e-4aae-9db1-d60407d13c35

```
iex(6)> "hel^A"
<<104, 101, 108, 1>>
iex(7)> "hello
...(7)> world"
"hello\nworld"
iex(8)> "hello \nworld"
"hello \nworld"
```

```
iex(9)> is_binary("hello")
true
iex(10)> byte_size("hello world")
11
iex(11)> String.upcase("hello")
"HELLO"
```

Задача 8. Анонимни функции fn … end

```
iex(16)> add=fn a, b ->a+b end
#Function<43.65746770/2 in :erl_eval.expr/5>
iex(17)> is_function(add)
true
iex(18)> is_function(add,2)
true
iex(19)> is_function(add,1)
false
iex(20)> add.(1, 2)
3
```

```
iex(21)> add_two= fn a ->add.(a,2) end
#Function<44.65746770/1 in :erl_eval.expr/5>
iex(22)> add_two.(2)
4
iex(23)> x=42
42
```

Задача 9. Свързани списъци

```
iex(29)> [1, 2, true, 3]
[1, 2, true, 3]
```

```
iex(30)> length [1,2,3]
3
```

```
iex(31)> [1,2, 3]++[4,5,6]
[1, 2, 3, 4, 5, 6]
iex(32)> [1,true,false, true]--[true,false]
[1, true]
```

```
iex(39)> list=[1, 2, 3]
[1, 2, 3]
iex(40)> hd(list)
1
iex(41)> tl(list)
[2, 3]
```

```
iex(42)> list=[]
[]
iex(43)> hd(list)
** (ArgumentError) errors were found at the given arguments
:

  * 1st argument: not a nonempty list

    :erlang.hd([])
```

```
iex(43)> [11, 12, 13]
'\v\f\r'
iex(44)> [104, 101, 108, 108, 111]
'hello'
iex(45)> 'hello'=="hello"
false
```

Задача 10. Кортежи

```
iex(49)> {:ok,"hello"}
{:ok, "hello"}
iex(50)> tuple={:ok,"hello"}
{:ok, "hello"}
iex(51)> elem(tuple,1)
"hello"
iex(52)> tuple_size(tuple)
2
iex(53)> tuple={:ok,"hello"}
{:ok, "hello"}
iex(54)> put_elem(tuple,1,"world")
{:ok, "world"}
iex(55)> tuple
{:ok, "hello"}
```

Задача 11. Списъци и кортежи.

```
iex(56)> list=[1|[2|[3|[]]]]
[1, 2, 3]
iex(57)> [0]++list
[0, 1, 2, 3]
iex(58)> list++[4]
[1, 2, 3, 4]
```

```
iex(66)> File.read("C:/Users/Dima/Desktop/Elixir/2023_2024_ELIXIR/Lab1Ex.exs")
{:ok, "IO.puts \"hello world\"\r\n"}
iex(67)> File.read("C:/Users/Dima/Desktop/Elixir/2023_2024_ELIXIR/LabEx.exs")
{:error, :enoent}
```

```
iex(68)> tuple={:ok,"hello"}
{:ok, "hello"}
iex(69)> elem(tuple,1)
"hello"
```

Задача 12. Основни оператори.

```
iex(72)> "foo"<>"bar"
"foobar"
iex(73)> true and true
true
iex(74)> false or is_atom(:examle)
true
iex(75)> 1 and true
** (BadBooleanError) expected a boolean on left-side of "and", got: 1

iex(75)> false and error("This error will never be raised")
** (CompileError) iex:75: undefined function error/1 (there is no such import)

    (stdlib 3.16.1) lists.erl:1358: :lists.mapfoldl/3
    (stdlib 3.16.1) lists.erl:1359: :lists.mapfoldl/3
    (stdlib 3.16.1) lists.erl:1358: :lists.mapfoldl/3
    (elixir 1.13.2) expanding macro: Kernel.and/2
iex(75)> true or error ("This error will never be raised")
** (CompileError) iex:75: undefined function error/1 (there is no such import)

    (stdlib 3.16.1) lists.erl:1358: :lists.mapfoldl/3
    (stdlib 3.16.1) lists.erl:1358: :lists.mapfoldl/3
    (elixir 1.13.2) expanding macro: Kernel.or/2
iex(75)> 1||true
1
```

```
iex(75)> 1||true
1
iex(76)> false||11
11
iex(77)> nil&&13
nil
iex(78)> true&&17
17
iex(79)> !true
false
iex(80)> !1
false
iex(81)> !nil
true
iex(82)> 1==1
true
iex(83)> 1!=2
true
iex(84)> 1<2
true
iex(85)> 1==1.0
true
iex(86)> 1===1.0
false
iex(87)> 1<:atom
true
```

Задача 13. **Съвпадение на шаблон (Pattern matching)**

```
iex(88)> x=1
1
iex(89)> x
1
iex(90)> 1=x
1
iex(91)> 2=x
** (MatchError) no match of right hand side value: 1

iex(91)> 1=unknown
** (CompileError) iex:91: undefined function unknown/0 (there is no such import)
```

```
iex(91)> {a,b,c}={:hello, "world", 42}
{:hello, "world", 42}
iex(92)> a
:hello
iex(93)> b
"world"
iex(94)> c
42
iex(95)> {a,b,c}={:hello, "world"}
** (MatchError) no match of right hand side value: {:hello, "world"}
```

```
iex(96)> {:ok, result}={:ok,13}
{:ok, 13}
iex(97)> result
13
iex(98)> {:ok, result}={:error, :oops}
** (MatchError) no match of right hand side value: {:error, :oops}
```

```
iex(98)> [a, b, c]=[1, 2, 3]
[1, 2, 3]
iex(99)> a
1
iex(100)> [head|tail]=[1,2,3]
[1, 2, 3]
iex(101)> head
1
iex(102)> tail
[2, 3]
iex(103)> [head|tail]=[]
** (MatchError) no match of right hand side value: []
```

```
iex(103)> list=[1,2,3]
[1, 2, 3]
iex(104)> [0|list]
[0, 1, 2, 3]
iex(105)> x=1
1
iex(106)> x=2
2
iex(107)> x=1
1
iex(108)> ^x=2
** (MatchError) no match of right hand side value: 2

iex(108)> {x, ^x}={2, 1}
{2, 1}
iex(109)> x
2
```

```
iex(110)> {x, x}={1, 1}
{1, 1}
iex(111)> {x, x}={1, 2}
** (MatchError) no match of right hand side value: {1, 2}

iex(111)> [h|_]=[1, 2, 3]
[1, 2, 3]
iex(112)> h
1
iex(113)> _
** (CompileError) iex:113: invalid use of _. "_" represents a value to be ignored in a pattern an
d cannot be used in expressions

iex(113)> length([1, [2], 3])=3
** (CompileError) iex:113: cannot invoke remote function :erlang.length/1 inside a match
```

```
iex(113)> case {1, 2, 3} do
...(113)> {4, 5, 6} ->
...(113)> "This caluse won't match"
...(113)> {1, x, 3} ->
...(113)> "This clause will match and bind x to 2 in this clause"
...(113)> _->
...(113)> "This clause would match any value"
...(113)> end
warning: variable "x" is unused (there is a variable with the same name in the context, use the pin operator (^) to match on it or prefix this varia
le with underscore if it is not meant to be used)
  iex:116
```

Задачи с примерни решения

Задача 1. Да се създаде приложение Лихвен калкулатор, което по зададена сума пресмята проста и сложна лихва за ден, месец и година.

Формули за проста лихва:
Kn=Ko*(1+((p*n)/100)) **(1.1)**
Kn – сума за погасяване
Ko - изтеглена сума

p– процент за олихвяване
n – период за година
n=m/12, където **m** е период в месеци
n=t/B, където **t** е период в дни, **B**=360 дни (приема се, че 1 месец има средно 30 дни)
Формули за сложна лихва:
$Kn=Ko*(1+(p/100))^n$ **(1.2)**

Примерно решение:

```
defmodule T do
 def enterData() do
   ko = IO.gets("Enter taken sum= ") |> String.trim()
   p = IO.gets("Enter procent= ") |> String.trim()
   op = IO.gets("Choice: y/ m/ d= ") |> String.trim()
   m = IO.gets("Enter period #{op}= ") |> String.trim()
   if String.match?(ko, ~r/^\d+$/) and String.match?(p, ~r/^\d{1,2}$/) and String.match?(m, ~r/^\d{1,3}$/) do
     {ko, _} = Integer.parse(ko)

         # ~r (sigil) дефинира низ и го предава на компилирането на regex в една и съща стъпка
         https://elixir-lang.org/getting-started/sigils.html
         # ^ Започва с
         # \d Връща съвпадение, където низът съдържа цифри (числа от 0-9)
         # + Едно или повече събития
         # $ Завършва с
     {p, _} = Integer.parse(p)
         #{op, _} = String.parse(op)
     {m, _} = Integer.parse(m)

     calcSimple(ko, p, op, m)
     calcDiff(ko, p, op, m)

   else
     IO.puts("One of the entered params was not correct!")
     T.enterData()
   end
 end

 def calcSimple(ko, p, op, m) do
   if op==="y" do
      n = m
      kn = (ko * (1 + ((p * n) / 100))) |> Float.round(2)
      IO.puts("Simple interest: #{kn}")
   end

    if op==="m" do
     n = m / 12
```

```
      kn = (ko * (1 + ((p * n) / 100))) |> Float.round(2)
      IO.puts("Simple interest: #{kn}")
    end

    if op==="d" do
     n = m / 360
      kn = (ko * (1 + ((p * n) / 100))) |> Float.round(2)
      IO.puts("Simple interest: #{kn}")
    end
  end

  def calcDiff(ko, p, op, m) do
    if op==="y" do
      n = m
      kn = (ko * :math.pow((1+(p/100)), n)) |> Float.round(2)
      IO.puts("Diff Interest: #{kn}")
    end
    if op==="m" do
      n = m / 12
      kn = (ko * :math.pow((1+(p/100)), n)) |> Float.round(2)
      IO.puts("Diff Interest: #{kn}")
    end
    if op==="d" do
      n = m / 360
      kn =  (ko * :math.pow((1+(p/100)), n)) |> Float.round(2)
      IO.puts("Diff Interest: #{kn}")
    end
  end
end

T.enterData()
```

..... вариант с else if

```
def calcSimple(ko, p, op, m) do
    if op==="y" do
       n = m
       kn = (ko * (1 + ((p * n) / 100))) |> Float.round(2)
       IO.puts("Simple interest: #{kn}")

     else
     if op==="m" do
      n = m / 12
       kn = (ko * (1 + ((p * n) / 100))) |> Float.round(2)
       IO.puts("Simple interest: #{kn}")

     else
```

```
    if op==="d" do
     n = m / 360
       kn = (ko * (1 + ((p * n) / 100))) |> Float.round(2)
       IO.puts("Simple interest: #{kn}")
    end
    end
   end
 end
```

Примерни входни данни:

Ще намерим количеството, **до което** ще нарастнат 100 лв, внесени на влог за 5 години при сложна лихва 3% за година. Прилагаме формулата за сложната лихва $K_n = K_0\left(1+\frac{p}{100}\right)^n = K_0 q^n$, където K_0=100, $p = 3$, $n = 5$, $q = 1+\frac{3}{100} = \frac{103}{100} = 1{,}03$. Тогава в края на петата година стоте лева ще нарастнат до количеството $K_5 = 100.\left(1+\frac{3}{100}\right)^5 = 100.\left(\frac{103}{100}\right)^5 = 100.(1{,}03)^5$.

[1]. http://mitko.villaverde-bansko.com/Matematika/single-lesson-04-7.html

Задача 2. Да се създаде приложение Квадратно уравнение от вида $a*x^2+b*x+c=0$. За целта да се използва формула (2.1). Да се пресметнат дискриминантата и квадратните корени на уравнението по зададени стойности за a, b и c. Да се направи проверка за коректни входни данни, да се изведат подходящи съобщения, ако уравнението има два еднакви корена или няма реални корени. Резултата да се закръгли до втори знак след десетичната запетая.

$x_{1/2} = (-b \pm \sqrt{(b^2-4*a*c)})/(2*a)$ (2.1)

Примерно решение:

```
defmodule T do
 def test(a, b, c) do
  if String.match?(a, ~r/^\d+(\.\d{1,2})?$/) and String.match?(b, ~r/^\d+(\.\d{1,2})?$/) and
      String.match?(c, ~r/^\d+(\.\d{1,2})?$/) do
    {a, _} = Float.parse(a)
    {b, _} = Float.parse(b)
    {c, _} = Float.parse(c)
    d = b * b - 4 * a * c
    IO.puts("D is: #{d}")
    if d < 0 do
      IO.puts("No real roots!")
    else
      if d == 0 do
        x1 = (-b / (2 * a)) |> Float.round(2)
        IO.puts("X1=X2 is: #{x1}")
      else
        if d > 0 do
          # d = :math.sqrt(d)
```

```
          x1 = (-b + :math.sqrt(d) / (2 * a)) |> Float.round(2)
          x2 = (-b - :math.sqrt(d) / (2 * a)) |> Float.round(2)
          IO.puts("X1 is: #{x1}")
          IO.puts("X2 is: #{x2}")
        end
      end
    end
  else
    IO.puts("One of the variables is not a number")
  end
 end
end

a = IO.gets("Enter a: ") |> String.trim()
b = IO.gets("Enter b: ") |> String.trim()
c = IO.gets("Enter c: ") |> String.trim()
IO.puts(T.test(a, b, c))
```

Вход:

Enter a: 2
Enter b: 4
Enter c: 2
D is: 0.0
X1=X2 is: -1.0
ok
....
Enter a: 2
Enter b: 6
Enter c: 4
D is: 4.0
X1 is: -5.5
X2 is: -6.5
ok
....
Enter a: 4
Enter b: 1
Enter c: 6
D is: -95.0
No real roots!
ok

Задължителни задачи за самостоятелна работа

1. Да се определи дали едно число е четно или нечетно.
2. Да се създаде модул чрез който се изислява лице на фигура (правоъгълник, квадрат, окръжност, триъгълник, правоъгълен паралелепипед, ромб, конус, триъгълна и четириъгълна пирамида)
3. Да се определи вида на триъгълник по зададени страни/ ъгли.
4. Да се създаде приложение „НОК“ на две числа.
5. Да се създаде приложение за намиране на сумата от цифрите на въведено число.
6. Сламки трябва да се разпределят в кутии – големи по 50 бр., средни по 20 бр. и малки по 5 бр. Да се създаде функция, която връща броят на кутиите .

Задължителна домашна работа
Самостоятелна реализация на игра по избор (не преписана от Интернет!!!):

1. Морски шах
2. Бесеница

Срок за изпълнение текущият ден от провеждане на ЛУ!!!

След всяко ЛУ, кодът и резултата от изпълнението на задачите се копират в word документ и изпращат в Teams на х.ас. инж. М. Добрев.

В началото на всеки документ се посочват три имена, фак. номер, дата на провеждане на ЛУ и преди всяка задача условието и.

Например:

ПРОТОКОЛ 1:

Дата:…… ФАК. ……	**Три имена:……**	
Зад. 1	Условие	Релизация
Зад. 2	Условие	Релизация
Зад. N	Условие	Релизация