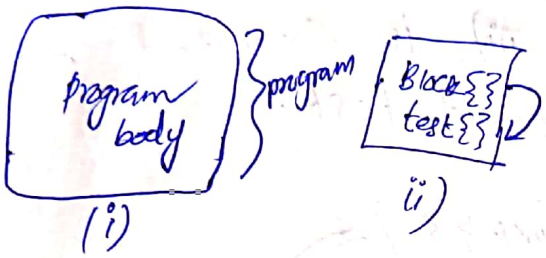
Smartherd (Ruby tutorials) → YT channel

#24 Blocks and yield, with and without parameters in ruby

```
Block{
#chunk of code
}
```

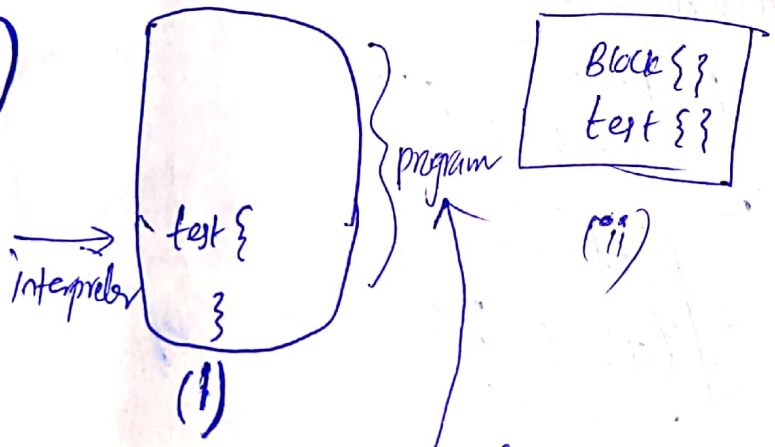
Syntax of Block without parameter

I)



Suppose (i) is the program body, and inside the program body we need to define a block, for example - we are having the block named test here. It is a user defined block. you can provide any name to block. test, test2 anything.

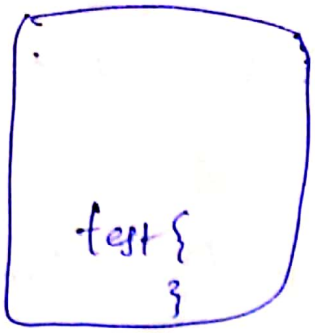
II)



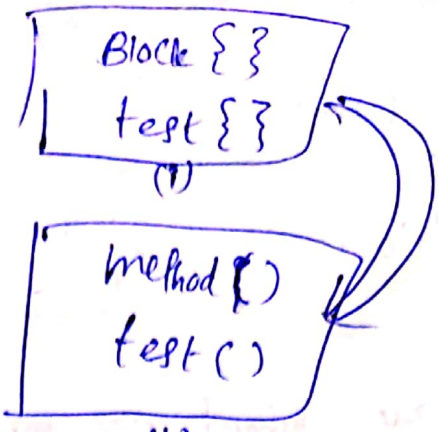
* we invoke a block from inside a function by using yield statement
* A block is always invoked from a function with the same name as that given block

So in (i) we defined the test block. within the curly braces we will put our codes here. Now whenever we execute this program, Then when interpreter comes to test block, then it is going to look for the method named test. (chunk)

III)



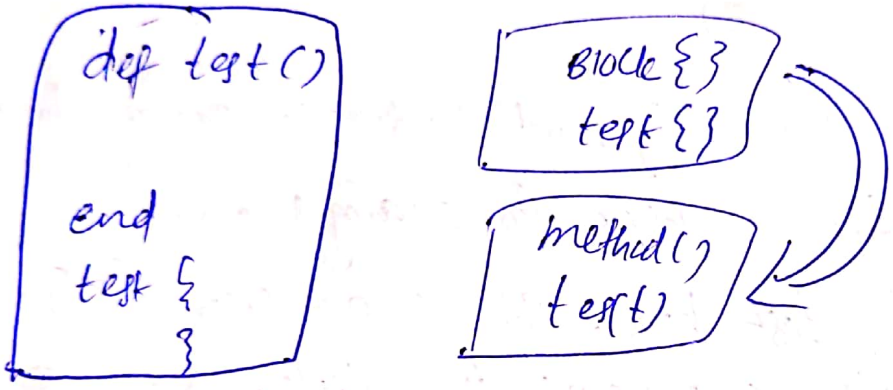
iii)



ii)

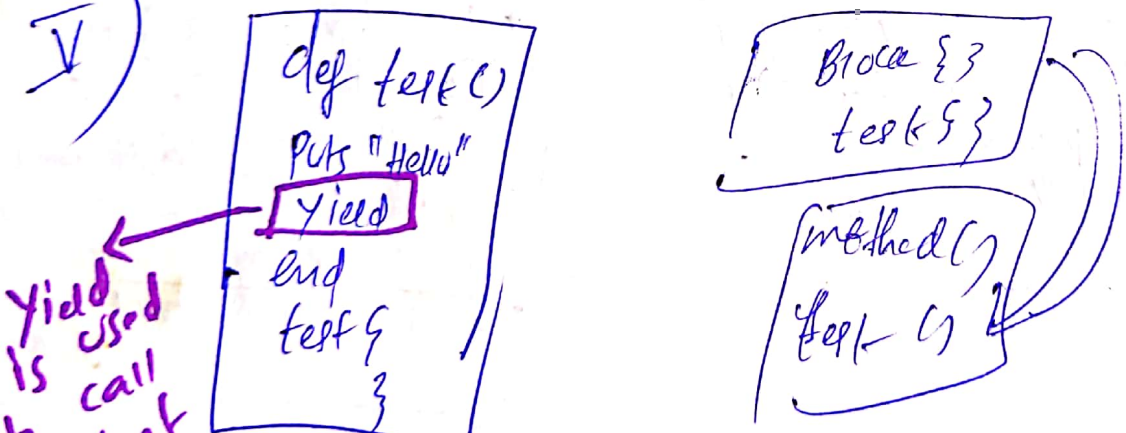
So when test Block in the iii) is executed it looks for method named test.

IV)



"Block and method should have same name"

V)



Yield is used to call the test block

We use yield to call the test{} block from inside the test() function.

VI)

```
def test()
puts "you are in the method"
yield
end
test{
puts "you are in the block"
}
```

```
Block { }
test { }
Method()
test()
```

"Block" search for method having the same as that of block and if "Block" sees yield statement inside the method. the "Block" gets called or codes inside the block gets executed.

yield

Execute the codep of the blocke

Output

you are in the method
you are in the blocke

yield statement prints both method and blocke code.

if method and blocke both present only in the program

but

```
def test()
puts "you are in the method"
yield
end
```

→ output ⇒ nothing

You can say that if you provide yield inside method the codep inside both method and block gets printed.

```
def test()
puts "you are in the method"
yield
end test {
puts "you are in the blocke"
}
```

→ output ⇒

1) you are in the method
2) you are in the blocke

- when the test {} block is encountered, it will call the test() method, then "you are in the method" printed. And after that yield is encountered and yield will execute the codes b/w the curly braces of block which is "you are in the block"

main concept

Syntax of block with parameter (argument)

1)

```
def test()
  puts "you are in the method"
  yield 5
end
test {
}
```

```
Block {}
test {}
```

```
Method()
test()
```

yield

Execute the codes of the block {}

yield statement

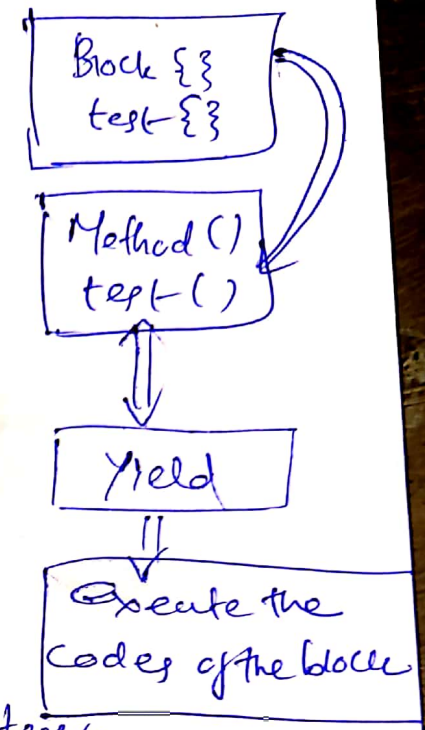
* yield → 5 → parameter → counter variable

5 is the parameter we send to the test {} block, for the execution of the code b/w the curly braces of block.

II) when yield 5 is encountered it passes (5) to the test{} block.

```
def test()
  puts "you are in the method"
  yield 5
end
  test {
    |i| puts "you are in the block #{i}"
  }
```

"interpolation"



⇒ Value (5) or parameter (5) is catched by the variable 'i', then i becomes an integer. and with the help of interpolation we are printing the value 'i'. then output is

```
You are in the method.
You are in the block 5
```

check #4 video

Note:-

BEGIN and END Blocks ⟹ Not user defined blocks. They are ruby in build blocks (Pre defined blocks)

(keywords)

```
BEGIN {
  This block will be executed at the beginning
}
  Block of program body
END {
  This block will be executed last
}
```