Functions without names:

- anonymous funs

```
scala> def iterPattern(n:Int,limit:Int,base:Int,combine:(Int,Int)=>Int,reduce:(Int)=>Int):Int = {
     | if (n==limit) base
     | else combine(n,iterPattern(reduce(n),limit,base,combine,reduce))
     | }
iterPattern: (n: Int, limit: Int, base: Int, combine: (Int, Int) => Int, reduce: Int => Int)Int

scala> def fac5(n:Int) : Int = {
     | iterPattern(n,0,1,(a:Int,b:Int) => { a*b },(a:Int) => { a-1 })
     | }
fac5: (n: Int)Int

scala> fac5(5)
res0: Int = 120

scala> fac5(4)
res1: Int = 24

scala> val t = (a:Int,b:Int) => { a * b }
t: (Int, Int) => Int = $$Lambda$1063/1864007931@5a237731

scala>
```

t is actually a function we can use.

```
scala> t(6,5)
res2: Int = 30

scala> def t(a:Int,b:Int) = { a * b }
t: (a: Int, b: Int)Int

scala> t(4,5)
res3: Int = 20
```

```
scala> iterPattern(5,0,1,(a:Int,b:Int) => { a*b },(a:Int) => { a-1 })
res4: Int = 120

scala> iterPattern(5,0,1,_ * _, _ - 1)
res5: Int = 120

scala> iterPattern(5,0,1,_*_, _ - 1)
res6: Int = 120

scala> iterPattern(5,0,1,_*_, _-1)
res7: Int = 120
```

```
scala> def addFactory(n:Int) : (Int) => Int = _ + n
addFactory: (n: Int)Int => Int

scala> val add5 = addFactory(5)
add5: Int => Int = $$Lambda$1173/1478984550@7f5fcfe9

scala> add5(4)
res8: Int = 9

scala> add5(2)
res9: Int = 7

scala> add5(0)
res10: Int = 5


scala> val add10 = addFactory(10)
add10: Int => Int = $$Lambda$1173/1478984550@19b5214b

scala> add10(0)
res11: Int = 10

scala> add5(0)
res12: Int = 5

scala>
```

```
scala> def addFactory2(n:Int) : (Int) => Int = {
     |  def adder(x:Int) = { x + n }
     |  adder
     | }
addFactory2: (n: Int)Int => Int

scala> addFactory
addFactory   addFactory2

scala> addFactory2(5)
res13: Int => Int = $$Lambda$1185/1336483163@7ce30c0b

scala> addFactory2(5)(8)
res14: Int = 13

scala> addFactory2(5)(7)
res15: Int = 12
```

* return VS. call.

```
scala> def addFactory2(n:Int) : (Int) => Int = {
     |  def adder(x:Int) = { x + n }
     |  adder _
     | }
addFactory2: (n: Int)Int => Int
```

* return the function adder without putting the arguments itself.

```
scala> import scala.io.Source
import scala.io.Source

scala> val file = Source.fromFile("alice.txt","UTF-8")
file: scala.io.BufferedSource = non-empty iterator

scala> val text = file.mkString
```

```
scala> val words = text.split("\\s+")
words: Array[String] = Array(Project, Gutenberg's, Alice's, Adventures, in, Wonderland,, by, Lewis, Carroll, Thi
s, eBook, is, for, the, use, of, anyone, anywhere, at, no, cost, and, with, almost, no, restrictions, whatsoever
., You, may, copy, it,, give, it, away, or, re-use, it, under, the, terms, of, the, Project, Gutenberg, License,
 included, with, this, eBook, or, online, at, www.gutenberg.net, Title:, Alice's, Adventures, in, Wonderland, Il
lustrated, by, Arthur, Rackham., With, a, Proem, by, Austin, Dobson, Author:, Lewis, Carroll, Illustrator:, Arth
ur, Rackham, Release, Date:, May, 19,, 2009, [EBook, #28885], Language:, English, ***, START, OF, THIS, PROJECT,
 GUTENBERG, EBOOK, ALICE'S, ADVENTURES, IN, WONDERLAND, ***, Produced, by, Jana, Srna,, Emmy, and,...
```

```
scala> words.sorted
res18: Array[String] = Array("'--found, "'Miss, "'Tis, "'Tis, "'_We, "----, "----likely, "----or, "----that, "--
--well, "----where's, "--I, "--Mystery,, "--and, "--and, "--as, "--but, "--change, "--for, "--so, "--yes,, "--yo
u, "A, "A, "A, "A, "A, "A, "A, "A, "A, "After, "After, "Ah!, "Ah!, "Ah,, "Ah,, "Ahem!", "Alice!", "Alice"], "All
, "All, "An, "And, "And, "And, "And, "And, "And, "And, "And, "And, "And, "And, "And, "And, "And, "And, "And, "An
d, "And, "Anything, "Are, "Are, "Are, "Are, "Are, "As, "As, "As, "At, "Back, "Beautiful, "Beautiful, "Begin, "Be
head, "Boots, "Bring, "But, "But, "But, "But, "But, "But, "But, "But, "But, "But, "But, "But, "But, "But, "But,
"But, "But, "But, "But, "But, "But, "But, "By-the-bye,, "CURIOUSER, "Call, "Call, "Call, "Call, "Ca...
```

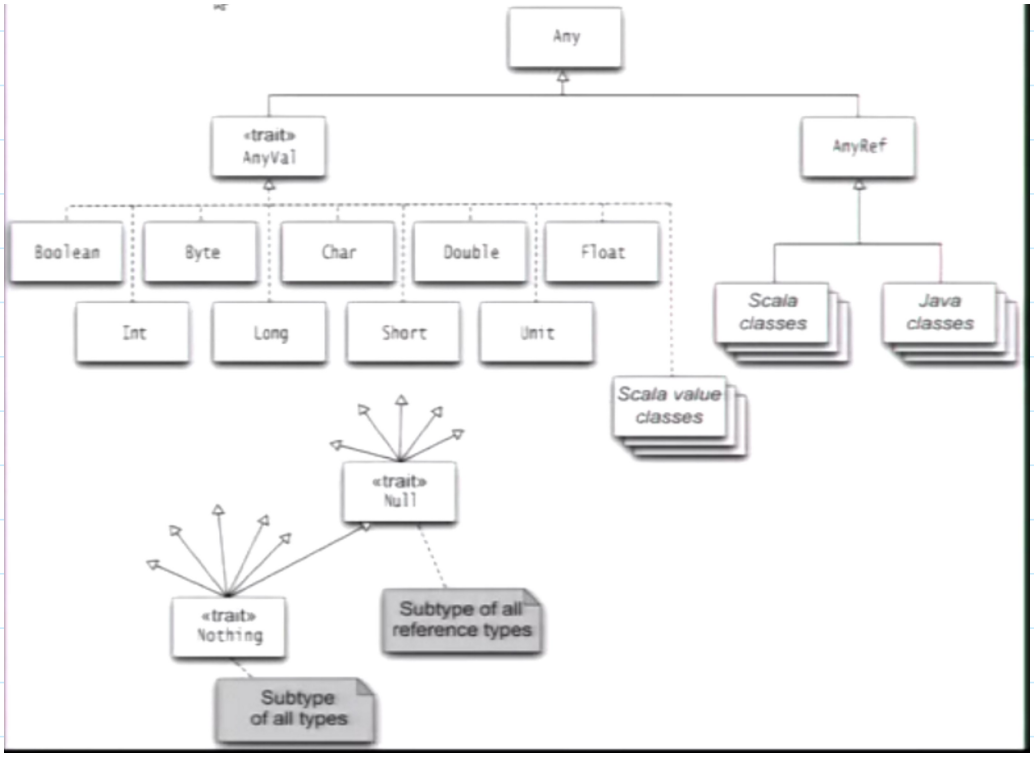
```
scala> words.sortWith(_ > _)
res19: Array[String] = Array(Project, zigzag,, youth,", youth,", youth,", youth,, youth,, youth,, yourself.", yo
urself.", yourself,", yourself!", yourself, yourself, yourself, yourself, yourself, yourself, yours.'", yours, y
our, your, your, your, your, your, your, your, your, your, your, your, your, your, your, your, your, your, your,
 your, your, your, your, your, your, your, your, your, your, your, your, your, your, your, your, your, your, you
r, your, your, your, your, your, your, your, your, your, your, your, your, your, your, your, your, your, your, y
our, your, your, your, your, your, your, young, young, young, young, young, you_,'", you?_'--What,, you?", you?"
, you?", you?", you?", you?", you?, you?, you:, you.", you., you--are, you--all, you,", you,", you...
```

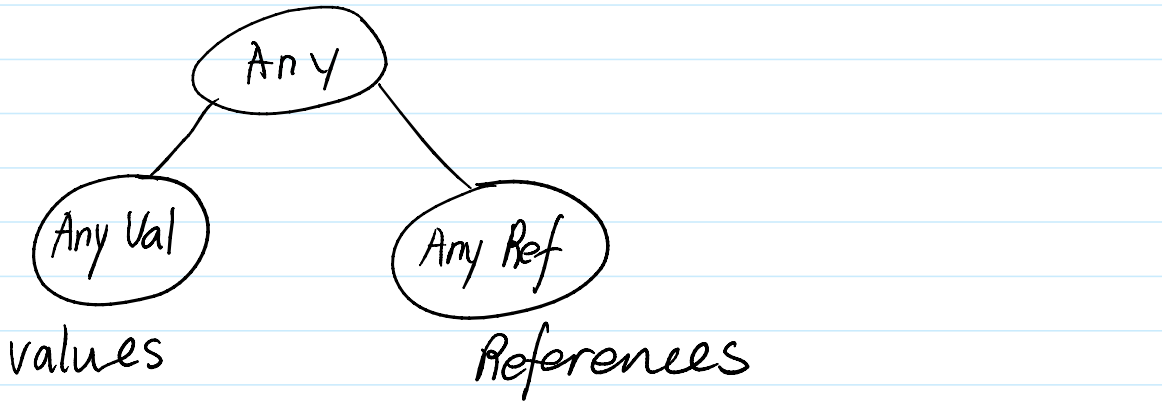
```
scala> words.sortWith((x,y) => (x>y))
res20: Array[String] = Array(Project, zigzag,, youth,", youth,", youth,", youth,, youth,, youth,, yourself.", yo
urself.", yourself,", yourself!", yourself, yourself, yourself, yourself, yourself, yourself, yours.'", yours, y
our, your, your, your, your, your, your, your, your, your, your, your, your, your, your, your, your, your, your,
 your, your, your, your, your, your, your, your, your, your, your, your, your, your, your, your, your, your, you
r, your, your, your, your, your, your, your, your, your, your, your, your, your, your, your, your, your, your, y
our, your, your, your, your, your, your, young, young, young, young, young, you_,'", you?_'--What,, you?", you?"
, you?", you?", you?", you?", you?, you?, you:, you.", you., you--are, you--all, you,", you,", you...
```

```
scala> words.sortWith((x:String,y:String) => (x>y))
res21: Array[String] = Array(Project, zigzag,, youth,", youth,", youth,", youth,, youth,, youth,, yourself.", yo
urself.", yourself,", yourself!", yourself, yourself, yourself, yourself, yourself, yourself, yours.'", yours, y
our, your, your, your, your, your, your, your, your, your, your, your, your, your, your, your, your, your, your,
 your, your, your, your, your, your, your, your, your, your, your, your, your, your, your, your, your, your, you
r, your, your, your, your, your, your, your, your, your, your, your, your, your, your, your, your, your, your, y
our, your, your, your, your, your, your, young, young, young, young, young, you_,'", you?_'--What,, you?", you?"
, you?", you?", you?", you?", you?, you?, you:, you.", you., you--are, you--all, you,", you,", you...
```

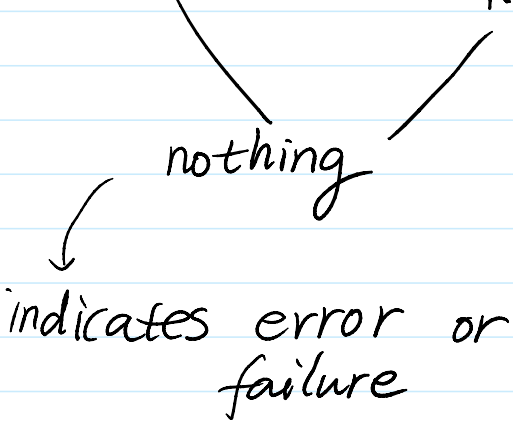
```
scala> words.sortWith((x:String,y:String) => (x>y):Boolean)
res23: Array[String] = Array(Project, zigzag,, youth,", youth,", youth,", youth,, youth,, youth,, yourself.", yo
urself.", yourself,", yourself!", yourself, yourself, yourself, yourself, yourself, yourself, yours.'", yours, y
our, your, your, your, your, your, your, your, your, your, your, your, your, your, your, your, your, your, your,
 your, your, your, your, your, your, your, your, your, your, your, your, your, your, your, your, your, your, you
r, your, your, your, your, your, your, your, your, your, your, your, your, your, your, your, your, your, your, y
our, your, your, your, your, your, your, young, young, young, young, young, you_,'", you?_'--What,, you?", you?"
, you?", you?", you?", you?", you?, you?, you:, you.", you., you--are, you--all, you,", you,", you...
```

```
scala> words.sortWith(((x:String,y:String) => (x>y)):(String,String) => Boolean)
res24: Array[String] = Array(Project, zigzag,, youth,", youth,", youth,", youth,, youth,, youth,, yourself.", yo
urself.", yourself,", yourself!", yourself, yourself, yourself, yourself, yourself, yourself, yours.'", yours, y
our, your, your, your, your, your, your, your, your, your, your, your, your, your, your, your, your, your, your,
 your, your, your, your, your, your, your, your, your, your, your, your, your, your, your, your, your, your, you
r, your, your, your, your, your, your, your, your, your, your, your, your, your, your, your, your, your, your, y
our, your, your, your, your, your, your, young, young, young, young, young, you_,'", you?_'--What,, you?", you?"
, you?", you?", you?", you?", you?, you?, you:, you.", you., you--are, you--all, you,", you,", you...
```





Null (null) → Subtype




```
scala> val s = ()
s: Unit = ()

scala> val t = if (3>4) 5 else 8
t: Int = 8

scala> val t:Int = if (3>4) 5 else 8
t: Int = 8

scala> val t = if (3>4) 5 else true
t: AnyVal = true

scala> val t = if (3>4) 5 else "hello"
t: Any = hello
```

```
def divide(x:Int, y:Int):Int = {
int . int if ( y ' = 0 ) x/y
```

```
def divide(x:Int, y:Int):Int = {
  if (y != 0) x/y
  else throw new RuntimeException(....)
}
```

int < int / nothing

Nothing

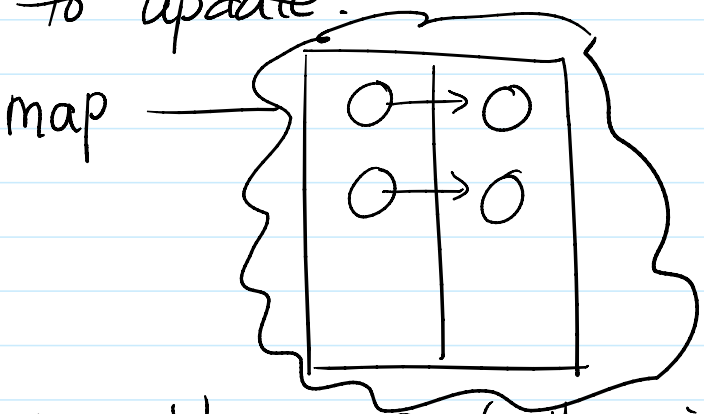
```
scala> def foo (x:Nothing) : Unit = {}
foo: (x: Nothing)Unit
```

```
scala> val grades = Map("Wallace" -> 0.5, "Esmeralda" -> 0.83)
grades: scala.collection.immutable.Map[String,Double] = Map(Wallace -> 0.5, Esmeralda -> 0.83)

scala> grades("Wallace")
res0: Double = 0.5

scala> grades("Wallace") = 0.7
<console>:13: error: value update is not a member of scala.collection.immutable.Map[String,Double]
       grades("Wallace") = 0.7
       ^
```

```
scala> val gradesM = scala.colletion.mutable.Map("Wallace" -> 0.5, "Esmeralda" -> 0.83)
<console>:11: error: object colletion is not a member of package scala
       val gradesM = scala.colletion.mutable.Map("Wallace" -> 0.5, "Esmeralda" -> 0.83)
                     ^

scala> val gradesM = scala.collection.mutable.Map("Wallace" -> 0.5, "Esmeralda" -> 0.83)
gradesM: scala.collection.mutable.Map[String,Double] = Map(Esmeralda -> 0.83, Wallace -> 0.5)

scala> gradesM("Wallace") = 0.7

scala> gradesM("Wallace")
res3: Double = 0.7
```

Mutable

→ We usually don't use mutable in Scala, instead, we use new collections to update.



Mutable map (Why it is a val?): only update the value inside it, but the whole thing doesn't change.

```
scala> def bell(pct:Double,grades:Map[String,Double]) : Map[String,Double] = {
```

```
scala> def bell(pct:Double,grades:Map[String,Double]) : Map[String,Double] = {
     | for((name,mark) <- grades) yield (name,mark*(1-pct)+pct)
     | }
bell: (pct: Double, grades: Map[String,Double])Map[String,Double]

scala> grades
res4: scala.collection.immutable.Map[String,Double] = Map(Wallace -> 0.5, Esmeralda -> 0.83)

scala> bell(0.1,grades)
res5: Map[String,Double] = Map(Wallace -> 0.55, Esmeralda -> 0.847)

scala> bell(0.8,grades)
res6: Map[String,Double] = Map(Wallace -> 0.9, Esmeralda -> 0.966)
```

map

map ((x) ⇒ { .... })

o □

Map [String, Double]

(,)

(,)

X._1

X._2

```
scala> val x = ("foo",3)
x: (String, Int) = (foo,3)

scala> x._1
res7: String = foo

scala> x._2
res8: Int = 3

scala> val (first,second) = x
first: String = foo
second: Int = 3
```

enhanced 'switch'

variable match {

case .... ⇒ ...

case ... ⇒ ...

defalut case _ ⇒ ...

}

```
scala> val name = "Bart"
name: String = Bart

scala> name match {
     | case "Bart" => "Yes"
     | case "Lisa" => "Lisa"
     | case _ => "default"
     | }
res9: String = Yes
```

```
scala> def testit(name:String) : String = {
     | name match {
     | case "Bart" => "Yes"
     | case "Lisa" => "Lisa"
     | case _ => "default"
     | }
     | }
testit: (name: String)String

scala> testit("Bart")
res10: String = Yes

scala> testit("Lisa")
res11: String = Lisa

scala> testit("Fred")
res12: String = default
```

```
scala> def checker(x:Any) : String = {
     | x match {
     | case v: String => "it's a string"
     | case (t1,t2) => "its a tuple: " + t1
     | case _ => "dunno"
     | }
     | }
checker: (x: Any)String

scala> checker("hello")
res13: String = it's a string

scala> checker(5)
res14: String = dunno

scala> checker((6,4))
res15: String = its a tuple: 6
```