scala

קריטריונים להערכת שפה
קריאות Readability

- Overall simplicity
  - חסר: ישנם דרכים רבות לבצע פעולות כגון:
    - הדפסה לקובץ
    - החזרת ערך מפונק' שלא מחייב שימוש ב return
  - יש העמסת אופרטורים (מתיחס אל אופרטורים ואל פונקציות בצורה זהה)
- Orthogonality
- Control Statements: יש שימוש במבנים מוכרים כגון while, for, switch
- Data types and structures
  - מאפשר שימוש בטיפוסים מגוונים כגון: class, object, trait, int, short, long, string
  - מבני נתונים מגוונים כגון: Map,Array, List, Set.
- Syntax consideration
  - הגדרה ברורה של משתנים, קל לזהות הגדרת משתנה לפי המילים השמורות val, var.
  - שימוש במילים שמורות keywords מוכרות ובעלות משמעות

קריטריונים להערכת שפה
כתיבות Writability

- Simplicity and orthogonality
  - חסר: יש מגוון טיפוסים, מבנים, ואופני ביצוע.
- Support for abstraction
  - יש אפשרות להגדיר פונקציות, מחלקות וממשקים.
- Expressivity
  - יש דרכים מקלות לבצע פעולות כגון:
    - בstring יש פונק' רבות כמו startWith, endWith
    - מגוון דרכים נוחות למיון כמו filter
    - שימוש בjava שמאפשר בעצמו כלים נוחים.
    - שימוש בobject למחלקת singleton

קריטריונים להערכת שפה
אמינות reliability

- Type checking
  - הקומפיילר בודק שגיאות השמה.
- Exception handeling
  - יש תמיכה במנגנון try-catch
- Aliasing
  - יש אפשרות ליצור כמה מצביעים לאותו מקום בזכרון
- Readability and writability
  - הקריאות לוקה בחסר ובשל כך האמינות נפגמת.

קריטריונים להערכת שפה
עלות cost

- מתכנתי scala גובים שכר גבוה מאוד.
- עלות קימפול נמוכה יחסית בשל הסתמכות על סביבות פיתוח קיימות.

קריטריונים להערכת שפה שונות

- Portability
  - מתקמפל לקוד הביניים java bytecode ואז ניתן להריץ באמצעות jvm על כול מחשב בעל סביבת עבודה של java.
- Generality
  - שימוש במגוון רחב של יישומים.
- Well-definedness
  - יש ספרים ומדריכים רבים וטובים.

פרדיגמות של השפה

- Scala משולבת מכמה פרדיגמות:
- אימפרטיבית- הצורה הרגילה.
- פונקציונאלית- פונקציה מתנהגת כמו טיפוס בשפה.
- מונחה עצמים- יש אבסטרציה, ירושה, פולימרפיזם.

גישות מימוש Implementation Methods

- שפת scala היא compilation.
- הקומפיילר מקמפל את הקוד לjava bytecode שיכול לרוץ על ידי scala commamd (הדומה לjava command). כלומר הוא רץ על java virtual machine.
- Scala נעזרת בJIT כי מסתמכת על JVM.

גישות מימוש Implementation Methods

- scala היא Interpretation דמה.
- קיימת לה shell בשם repl שמגיע יחד עם שפות שהם interperted. אבל למעשה כול "בלוק" מקומפל לbytecode ורצה ב jvm "על המקום".

scala	python

> def f():Unit={
var x=0
x="str"
}
On line 4: error: type mismatch;
found : String("lll")
required: Int
	> def f():
	x=0
	y="str"
	print(x+y)
	> f()
	Traceback (most recent call last):
	File "<pyshell#12>", line 1, in <module>
	f()
	File "<pyshell#11>", line 4, in f
	print(x+y)
	TypeError: unsupported operand type(s) for +: 'int' and 'str'

פקודות Preprocessors

- יש פקודות מקרו Preprocessors כגון:
  - Package, import

Programming Environments

Archive	System
scala-2.13.3.tgz	Mac OS X, Unix, Cygwin
scala-2.13.3.msi	Windows (msi installer)
scala-2.13.3.zip	Windows
scala-2.13.3.deb	Debian
scala-2.13.3.rpm	RPM package
scala-docs-2.13.3.txz	API docs
scala-docs-2.13.3.zip	API docs
scala-sources-2.13.3.tar.gz	Sources

- ניתן להוריד גרסאות מגוונות של scala למערכות הפעלה שונות. פרוט בטבלה להלן.
- סביבות פיתוח לscala:
  - Eclipse
  - IntelliJ

Type system

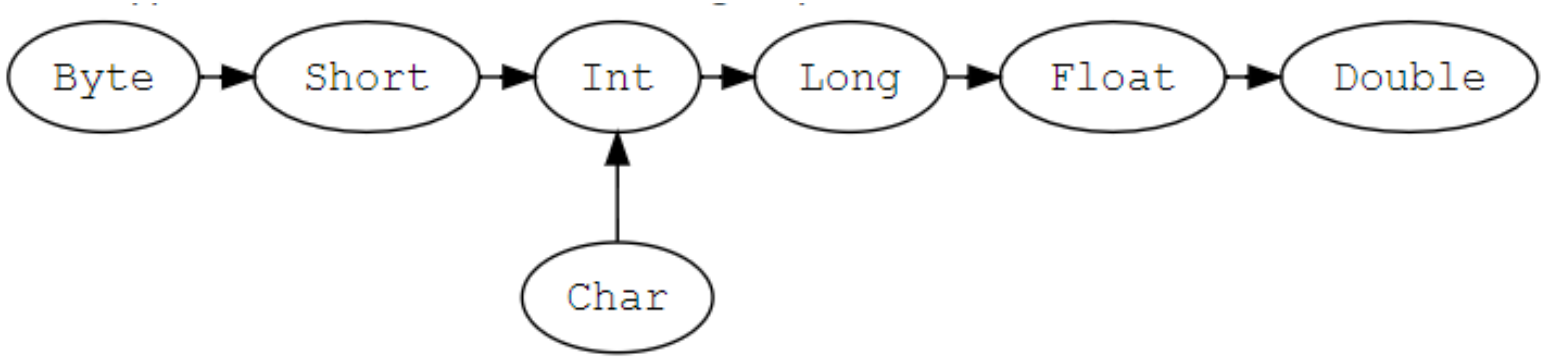
- Static typing
  - סוג המשתנה בscala נקבע בזמן הקומפילציה.
  - בזמן הצהרה על משתנה:
    - אין צורך לציין את סוג הטיפוס.
    - חובה לאתחל אותו.

Type system

- Strong type
  - מונע המרה מרומזת לדוג':

```
def f(x:String,y:String)=x+y
var x=2
var y:String=“3"
f(x,y)//compile error
```

- הסתייגות: מתבצעת המרה מרומזת עבור המקרים להלן:



טיפוסי נתונים ומימושם

- מניה- לscala יש מחלקה Enumeration הניתנת להרחבה כדי לאפשר לנו ליצור מניה במחלקות שלנו
- בוליאני- בscala יש סוג בוליאני Boolean.
- תו- בscala יש סוג נפרד לתוים Char.
- תת סוג של טיפוסים - (יש הפוך, אפשרות להגדיר טיפוס חדש שהוא הרחבה של טיפוס קיים באמצעות המילה type)

מצביעים

- כמו בjava אין אפשרות ליצור משתנים מסוג מצביע.
- אין dereference
- אין בעיות של מצביעים כמו מצביע מתנדנד.
- הטיפול בזליגת זכרון הינו על ידי הgarbage collector של הJVM.

reference

- על ידי הval אפשר להגדיר מצביע קבוע לאוביקט לדוג':

```
val mo=new MyOperand()
mo.setm(6)
 var mo2=new MyOperand()
 mo=mo2//compile error
```

- מחליף את השימוש במצביעים

scope

- scala מאפשרת קינון פונקציות.
- משום כך קיים static scope.
- יש גם קינון של בלוקים.
- Dynamic scope

טיפול בזיכרון

- jvm אחראי לשחרור הזכרון ולכן האלגוריתם של garbage collector תלוי בגרסה של jvm.

תתי תוכניות (פונקציות)

- העברת נתונים בשתי שיטות:
  - Positional parameters (מומלצת כשיש מעט פרמטרים)
  - Keyword parameters לדוג':

```
def main(args: Array[String])
 { printInt(b = 5, a = 7); }
 def printInt( a:Int, b:Int ) =
 {
   println("Value of a : " + a );
   println("Value of b : " + b );
}
```

- אפשר להעביר נתונים גם לפי שם(keyword) ומיקום, אך כשמעבירים פרמטר לפי שם חובה שהמשתנים שאחריו גם יועברו כך.

תתי תוכניות (פונקציות)

- ערכי ברירת מחדל:
  - אפשר לתת ערכי ברירת מחדל, אך כשמגדירים פרמטר עם ערך ברירת מחדל, חובה שהמשתנים שאחריו גם יוגדרו כך.
- Scala מאפשר קבלת מספר לא מוגדר של פרמטרים, ובתנאי שכולם מאותו הסוג.
  - באמצעות * מימין לסוג המשתנה.
- יש בדיקת סוג וכמות הפרמטרים.

תתי תוכניות (פונקציות)

- צורות קריאה:
  - נתונים מועברים by value ומועברים כval. ולכן מופחתות תופעות הלוואי.
  - כיון שלמעשה מועבר reference לנתון, מתקבל תוצאה של העברה by reference
  - מתקיים pass by name לדוג'

```
def something()= {
  println("calling something")
1
}
def callByName(x:=>Int)={
  print("x1="+x)
  print("x2="+x)
}
callByName(something())
/*output:
calling something
x1=1
calling something
x2=2
 */
```

נקודות התיחסות נוספות לתתי תוכניות

- חפיפת תתי תוכניות- מאפשר לחפוף תתי תוכניות overloading לפי:
  - סוג, כמות וסדר הפרמטרים.
  - אך לא לפי הערך המוחזר.
- גנריות- scala הקישור לפונקציה הינו סטטי ולכן מוגדר בו מנגנון למימוש תכנות גנארי.
- ערכים מוחזרים- יכול להחזיר מערכים, אובייקטים
- יש אפשרות לחפוף אופרטורים אבל למעשה הוא מתיחס אליהם כפונק'.
- רקורסיה- בscala אין משתנים סטטיים בשל כך ניתן לערוך רקורסיות
- גם אין פונקציות סטטיות

הפשטה והכמסה

- Scala מאפשרת להגדיר class, object, trait, struct, type
- סוגי הרשאות גישה לשדות ולפונקציות:
  - ניתן להגדיר הרשאות גם ברמת הscope
  - Protected מאפשר גישה רק ליורשים.
  - הרשאת ברירת המחדל היא public.

Modifier	Class	Package	Subclass	Every class
public	Yes	Yes	Yes	Yes
protected	Yes	No	Yes	No
private	Yes	No	No	No

- לאפשר גישה רגילה רק לסוגי בסיסים, כל השאר אובייקטים
- פולימרפיזם
- אין לו אפשרות של ירושה מרובה מclass אבל יש אפשרות לרשת מהרבה traits