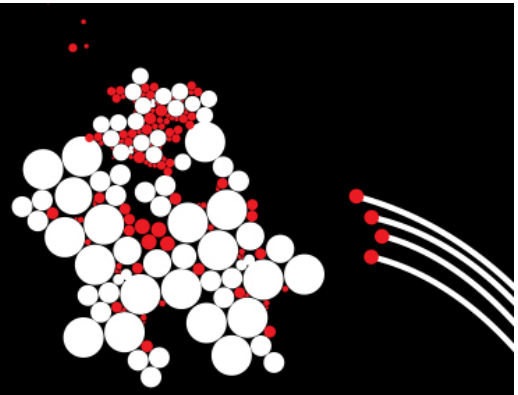
UNIVERSITY OF TWENTE.

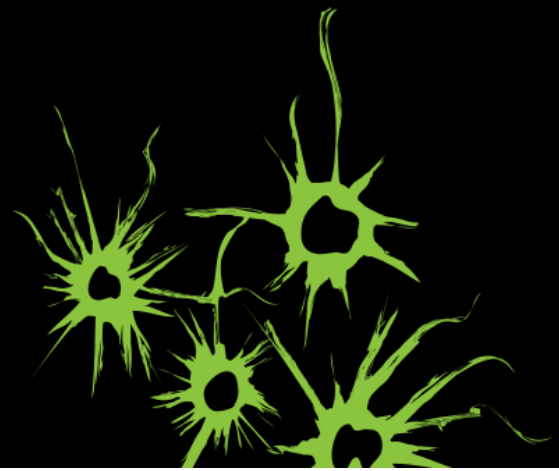


P1.1: VALUES AND VARIABLES

LUÍS FERREIRA PIRES

201700117-1B MODULE 2: SOFTWARE SYSTEMS

11 NOVEMBER 2019

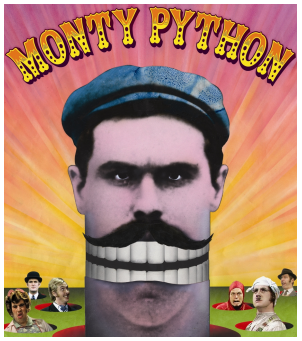


PROGRAMMING LINE OVERVIEW

Week 1 Values and variables Control flow	**Week 2** Classes and objects Testing	**Week 3** Interfaces and Inheritance Subtyping Security 1
Week 4 Arrays and Lists List implementations Collections	**Week 5** Stream I/O and MVC Exceptions Security 2	**Week 6** Concurrency Project kick-off IDE Tips & Tricks
Week 7 Basic Networking Networking and Multithreading GUIs	**Week 8/9** Advanced Java facilities Test	**Week 10** Project Test resit

MODULE 1 PREREQUISITES

- TCS Week 2: Algorithms (Python)
  - Lists
  - Searching, sorting



- Week 4: Functional Programming (Haskell)
  - Types
  - Recursion

For BIT programming weeks!!!

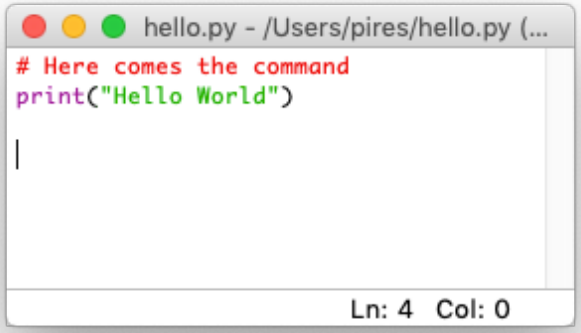
Haskell Curry
1900-1982

MODULE 2 PROGRAMMING THIS WEEK

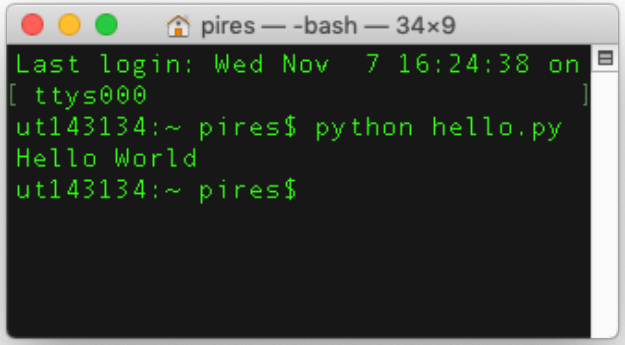
- Serious programming in Java
- Today
  - From Python (and Haskell) to Java
  - Variables, types, methods, basic statements
- Tuesday
  - Control flow
- Eck: Chapters 1 - 3

‘HELLO WORLD’ IN PYTHON

File: hello.py



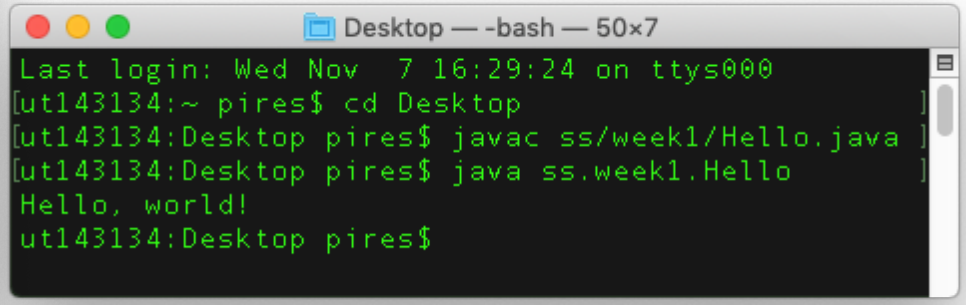
Usage



HELLO WORLD IN JAVA

File: ss\week1\Hello.java

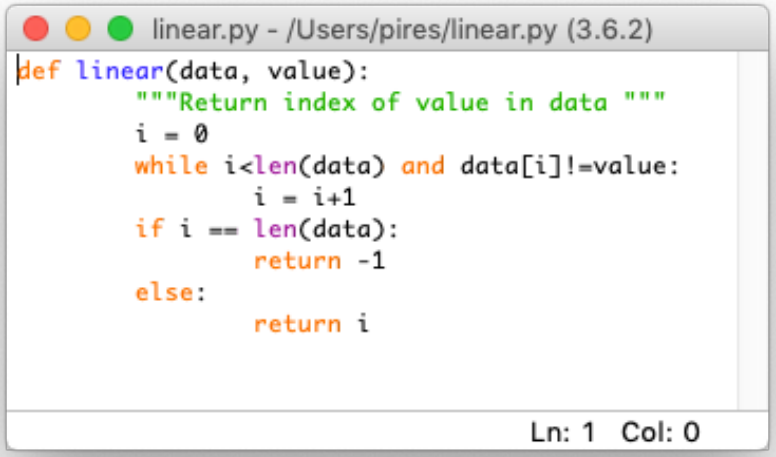
Usage



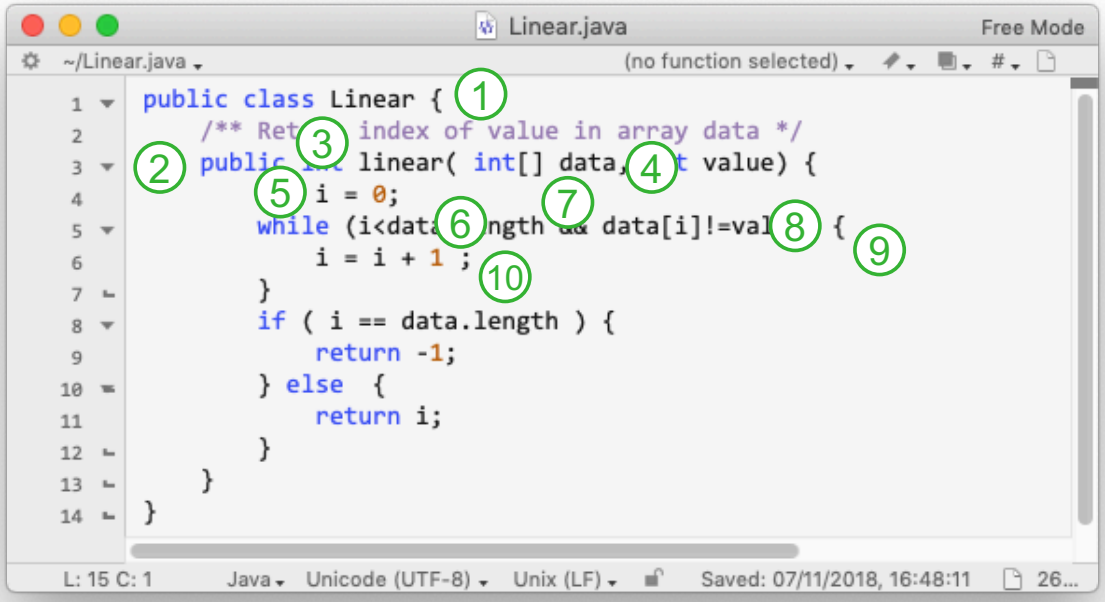
LINEAR SEARCH IN PYTHON VS. JAVA

SPOT THE DIFFERENCES!

Python



Java



DIFFERENCES!

1. Surrounding class declaration
2. Visibility modifiers (`public`, `private`)
3-5. Explicit data types (return, method variables and local variables)
6. `length` as an attribute instead of a method
7. `&&` instead of *and*
8. `( )` condition delimitation instead of :
9. `{ }` block delimitation instead of indentation
10. `;` statement separators instead of newline

Conceptually the same!

JAVA PROGRAM

```java
/**
 * Hello World class
 *
 *  @author Various authors
 */
public class HelloWorld {

    // main method to print "Hello world"
    public static void main(String[] args) {
        System.out.println("Hello world!");
    }
}
```

- Sequence of instructions

Bootstrapping

- In Java, to start a program the JVM is called with a reference to a class that has a `main` method (main class)
- Sequences of instructions are then called in the order in which they are declared in the body of `main` method (between `{` and `}`)
- Comments after `//` (single line) or between `/*` and `*/` (many lines)

VALUES

- Simple values ('literals')
  - Numbers: integer (`-11`), real (`1.3`)
  - Text (`'c'`, `"Hello World"`)
  - Truth values (`true`, `false`)
- Composed values
  - Lists of things
  - Combinations of things (dates, persons, etc.)

Primitive types in Java

`int`, `double`

`char`, `String`

`boolean`

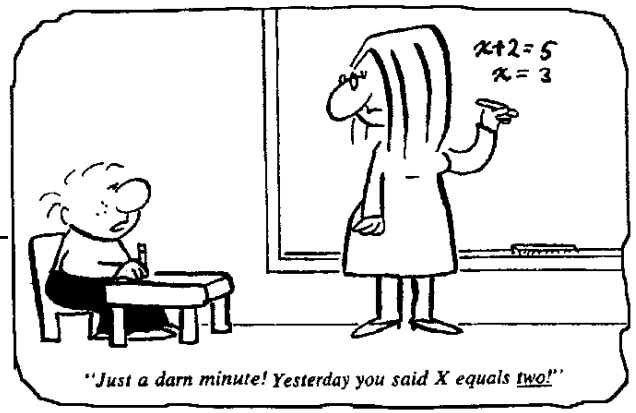
Check Eck Section 2.2 for primitive types!

Classes

→ `List` (pre-defined class)

→ User-defined classes

VARIABLES



- We want a way to refer to an unknown value
- Variable is a name for storage space of a value
- Variables are typed → only accept values of certain type
- Variables must be declared before being used
- Variables can change their value through an assignment
- At runtime, variables are stored in memory

In Java

```
int count;
count = 10;
```

declaration

assignment

In memory

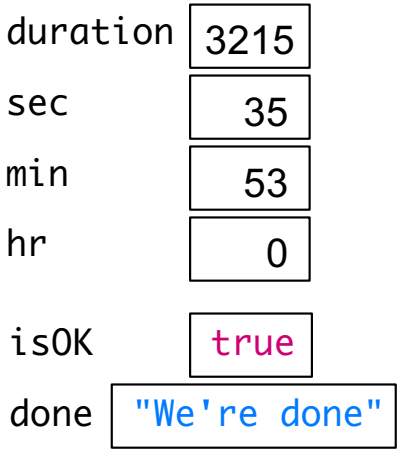
EXPRESSIONS

- Literals, variables and operators can be used in expressions
- Expressions must be assigned to variables of right type (with =)

Example

```
int duration = 3215;          // declaration + assignment
int sec = duration % 60;      // % stands for modulo
int min = duration / 60;      // integer division (no remainder)
int hr = min / 60;
min = min - 60*hr;            // re-assignment (no declaration)
boolean isOK = (duration == sec + 60*(min + 60*hr));
                              // boolean declaration + assignment
String done = "We're done"; // String assignment
```

boolean comparison



BOOLEANS (TRUTH VALUES)

George Boole
1815 – 1864

- `true` and `false` are values of type `boolean`
  - Values stored in variables and used in expressions
  - Operators: `v1 && v2` (and), `v1 || v2` (or), `!v` (not)
- Conditional evaluation of `&&` and `||` (lazy evaluation)
- With `&` and `|` both sides are evaluated

Sounds familiar?
(T, F, ¬ ∧ ∨)

```
boolean isLeapYear =
     yr % 4 == 0 && yr % 100 != 0 || yr % 400 == 0;
int febDays = isLeapYear ? 29 : 28;
boolean isEven = number % 2 == 0;
```

```
double x = 0.0, y = 1.2;
boolean xDividesY = x != 0 && y%x==0;
```

Would crash for `x == 0`
with `&` in place of `&&`

SUBROUTINES (JAVA METHODS)

DIVIDE AND CONQUER!

- Sequence of instructions could get long or may contain repetition
- Structuring and reusability have motivated subroutines
  → pieces of code that can be 'called' in a certain program
- Execution flow moves to subroutine and returns possibly with result

Example

Function: method that returns result!

- Method `Math.sqrt(x)` can be used to calculate the square root of `x`
- No need to know how `Math.sqrt(x)` is implemented (black box)
- Others: `System.out.print(text)`, `System.out.println(text)`

STRINGS

JAVA STRING API

- `String` is a pre-defined Java class to represent pieces of text
  - Example: `String advice = "Seize the day!";`
- Some functions:
  - `int s1.length()`: returns length of `s1`
  - `boolean s1.equals(s2)`: returns `true` if `s1` and `s2` are the same `String`
  - `char s1.charAt(n)`: returns `char` value in position `n` (first position 0!)
- Concatenate Strings: `"Hello " + "World"` same as `"Hello World"`

TEXTUAL OUTPUT

- Java has methods to print a `String` to the standard output (console)
- `System.out.print(text)`: prints text to standard output
- `System.out.println(text)`: prints text followed by line feed
- `System.out.printf(format, args)`: prints `args` according to `format`

Example

- `System.out.printf("Formatted string: %8d %.3f %c %s", x , y, c, s)`

TEXT IO LIBRARY

USED IN THE LAB SESSIONS

- Library introduced by Eck to facilitate Textual I/O mainly textual input
- Not really used in practice (only for education)
- Facilities to read input from the keyboard

```
int j = TextIO.getlnInt();              // Reads a value of type int.
double y = TextIO.getlnDouble();        // Reads a value of type double.
boolean a = TextIO.getlnBoolean();      // Reads a value of type boolean.
char c = TextIO.getlnChar();            // Reads a value of type char.
String w = TextIO.getlnWord();          // Reads one "word" as a String.
String s = TextIO.getln();              // Reads an entire input line as a String.
```

CONSTANTS

‘VARIABLE’ WITH VALUES THAT CANNOT BE CHANGED

- Declared as `final` (next lecture: `static`)
- Purpose: Understandability and maintainability
- Use constants if possible!

```
public static final int ROOK = 0;
public static final int KNIGHT = 1;
public static final int BISHOP = 2;
```

Alternative: enumerated type (Eck Sec. 2.3.4)

```
enum ChessPiece { ROOK, KNIGHT, BISHOP }
```

```
public static final int M2S = 60; // minutes to seconds
public static final int H2M = 60; // hours to minutes

int duration = 3215;
int sec = duration % M2S;
int min = duration / M2S;
int hr = min / H2M;

min = min - H2M*hr;
```

TAKE HOME MESSAGES

- Java is explicitly and strongly typed
  - Literals, variables, expressions, constants, methods all are typed
  - Everything must be declared before it can be used
- Variables refer to memory positions (to be better explained later)
- Strings are special Java pre-defined objects (`String` class)
- Subroutines have been introduced in the early years of programming to allow (basic) structuring and reusability