

## Research Paper

---

## Pick a language

Pick one of the following programming languages: Scala, Erlang, Ada, Perl, Python, Javascript, Lua, Ruby, Php, Eiffel, C#, F#, Visual Basic, Objective-C, Fortran, Cobol, Lisp, Scheme, OCaml, Mathematica, Matlab, *etc.*

Other languages may be added to this list by request. (Not Java, Haskell, C-Shell, Bash or Smalltalk).

Every one has to choose a different language. You need to email me two possible programming languages you would like to study and get my ok as soon as you pick one. First person to pick a language gets to use it.

## Write a report on it

Your assignment is to look at your chosen language and write a report explaining its design choices.

Below is a list of potential questions you may want to answer. Some of the questions below will not be relevant to all languages. If you cannot answer these, make up your own reasonable questions to answer. We will have covered many more design subjects in this course than I mentioned below.

Please write in either full sentences and paragraphs or in some obvious-to-read format like a bulleted questions. Your goal is to be informative rather than long winded. You must type your report.

## Cite your references

You can use any books, knowledgeable websites (this means not random opinions found on the web unless you can show they were correct) or industry experts you find (Uncle Fred may be a language expert) for references. You may use Wikipedia as a starting point but not as a reference. Most languages have a website and tutorial provided by the people who created the language. Your report must have references cited. (You can use footnotes or a reference section). I will want citations such that I can tell where you got each fact you looked up.

Bad citation example:

fact fact fact  works cited: 1: <a href="http://www.website.com/qwerty/Foozle.html">www.website.com/qwerty/Foozle.html</a> The official Foozle website 2: How to program in Foozle, John Doe, ... 3: Uncle Fred, personal communication, Nov 11, 2011 4: <a href="http://www.somerandomwebsite.com">www.somerandomwebsite.com</a> Foozle examples
---

These citations are bad because I cannot tell where you got which facts without reading each and every one of your cited references and guessing.

Good citation example:

```
fact [1]
fact [2,3]
fact [4]
```

works cited:

- 1: [www.website.com/qwerty/Foozle.html](http://www.website.com/qwerty/Foozle.html) The official Foozle website
- 2: How to program in Foozle, John Doe, ...
- 3: Uncle Fred, personal communication, Nov 11, 2011
- 4: [www.somerandomwebsite.com](http://www.somerandomwebsite.com) Foozle examples

These citations are good because I can lookup more information on a topic and know which cited reference to use as a starting point.

## Code examples

When possible, give a code example that backs up your answers. (Don't take anything you for granted that you find on the Internet). Your code must actually compile and run. Show both the code and the test run. Provide instructions on how to compile and run your code.

Bad code example:

```
print "hello";
```

This is bad because it's not a complete example; I don't know how to run this; I don't know what the code should do when run.

Good code example:

```
begin main()
  print "hello";
end
```

Compile and run in one step by typing  
**foozle hello.fz** at the prompt.

The word hello will print.

You only have to explain how to run programs once. You should always provide code that can be run without further context and you should say what you to expect to happen when it runs.

## Grading

Your report must include *at least 25* questions and 6 code examples and citations to be acceptable. Grading will be done as letter grades. An merely acceptable report will earn at most a B+. Copious good code examples, good questions, and good citations will get you the rest of the way. Concise wording also helps. (Reports are generally between 3 and 10 pages).

## Phases of this assignment

These assignment phases will keep you on track and help you complete your report. They are due by midnight as listed on the course schedule. They should be put in your project folder of SVN and submitted. If a program requires several files, it should be in its own sub-directory.

You must write any code yourself but you are allowed to use **all** resources as long as you cite them in the assignment. (To clarify, if you find a program on the web, you can type it in yourself and cite it but say it is a copy not just an information source).

### P1

Write a helloworld program. It must include a comment which gives compilation and running instructions for Linux. It must also include a comment stating the languages name and paradigm(s).

## P2

Write a program demonstrating all the primitive types you can find. Give them values. Demonstrate what implicit type coercion happens and what explicit coercion you can do. Give an explanation in comments For example in Java:

```
public static class P2 {
    public static void main(String args[]) {
        bool a = true; // 1 bit
        byte b = 3; // 8 bit
        char c = 'x'; // 16 bit
        short d = 5; // 16 bit
        int e = 8; // 32 bit
        long f =9; // ...
        double g = 3.4; // ...
        // Size info from Java In A Nutshell O'Reilly 2.0 (citation)

        g = e; // implicit coercion of int to double, adds .0
        // e = g; implicit coercion of double to int fails
        e = (int) g; // explicit coercion to int, truncates
        ... (there are many more of these)
        // I found out the coercion info myself by trying it (therefore no citation)
    }
}
```

## P3-P6

Write a program demonstrating one of the following

1. 3 or more type constructors
2. inheritance
3. static or dynamic method binding (only applies to OOP)
4. deep vs shallow copying of objects (OOP only)
5. reflection (Reflective languages only)
6. hazards - which ones can you make?
7. declare variables, types, and/or methods to show the attributes you can set
8. static or dynamic scope or both
9. parameter passing mechanisms (reference or value, not functional languages)
10. list comprehensions
11. higher order functions
12. infinite lists
13. anonymous functions
14. polymorphism
15. exceptions
16. overloading of method names or operators

17. language pre-processors and macros
18. string parsing (regular expressions?)

More items can be added to this list if needed.

## Sample questions

Here are some sample questions for your report. Do not use the ones that do not apply to your language. You can also make up your own questions.

1. When was it created (range of years or a year)
2. Who created it (person, committee, group at MIT, ...)?
3. What paradigm(s) is it in?
4. Is it compiled or interpreted? (Or can you do both with it?) If it's somewhere in between, say so and explain.
5. Was it created for a purpose? What purpose?
6. Is anybody hiring programmers for this language right now? Who? What kind of jobs?
7. What are the regular expressions for its tokens?
8. What types of tokens does it have? (literals? keywords?)
9. Does it have a token delimiter? If so, what?
10. Free or fixed format? Dependent on the specific compiler? Always that way?
11. What comment styles does it support? State machines or regular expressions for them?
12. Static or dynamic scope?
13. What starts a scope? a function declaration? any curly braces? specific curly braces?

```
int x = 3;
if (x<5) {
    int x = 8;
    int y;
}
```

This runs in C because C makes a new scope for the if curly braces. This is a static semantic error in Java because it does not make a whole new scope and tells you that you have declared x twice but yet you cannot use y after the if ends. What does your language do?

14. What about the dangling else? Is it a problem? If yes, how do it handle it? If not, why not?
15. What basic (primitive) types does it have? (int, double, boolean, character, byte, ...)
16. Is string a primitive type? (In C++ it is, in Java, C, Haskell, ... it is not). What math operations can you do on strings?
17. What math operations can you do on numbers? Is this different for characters? Does it allow math that makes no sense (like 'd'/c').
18. What is the order or precedence for all the math operators? relational operators? any other operators?
19. Can you directly access or manipulate the bits of an integer variable? (You can in C, you cannot in Java)

20. Does your language use references? Can you manipulate them directly? What can be used with a reference? (C allows references to anything. Java allows references to Objects and Arrays, Some languages have them but dont let you manipulate them. Java lets you manipulate them.

```
int[] arr = new int[3];  
int[] arr2 = arr; // you set a reference directly
```

C even lets you do math on them. (Look up “pointer math” to see examples.)

21. Is it using a run time stack? A heap? Other? (This question may make no sense for functional languages).
22. Can a user define types (beyond the given type constructors)? If so, how?
23. What is the range of the various primitive types? In Java, an int holds values ranging from  $-(2^{31}) + 1$  to  $(2^{31}) + 1$  (32 bits, 1 is the sign bit, so 31 bits for the value, 0 takes up space too) Does this math look like magic? In a signed 4 bit integer, 1 bit is for the sign, the other 3 bits are for the value which ranges from 111 to 000 so we can represent -7 to +7 aka  $-(2^3) + 1..(2^3) + 1$  and we get both a negative and positive 0 even tho we dont need them both. The range has 15 values in it aka  $(2^4) - 1$  because i'm counting -0 and +0 as one value.
24. How do users represent types not found in the language? C has no booleans. Users say `int x = 0` is false and all other ints are true (-12 is true, 88 is true, 1 is true...) C and Haskell have no string. They both use an array of characters instead. C ends this array with a `'\0'` which is called the terminator.
25. What type constructors? (tuple, class, struct, matrix...) What formal definitions of constructors from your book and lecture do these match? (record, cartesian product, array, ...) Examples of you using these type constructors?
26. Can you do multi dimentional arrays? How? Are there limits?
27. Are there built in operations on constructed types? What are some? Libraries of operations that come with the language? (like the java libraries you import or the length function in Haskell, Haskell also lets you import more libraries).
28. Is it closest to untyped, weakly typed, or strongly typed? Example supporting your choice?
29. Does it do implicit type conversions? Example? If not, does it allow explicit conversions?
30. Does it have pointers? Example? If not, does it have some other way of halding references?
31. Does it have a garbage collector? Example? If not then how is it done?
32. Does it have exceptions? Example? If not, then how does it do error handling?
33. Does it use normal order evaluation or applicative order evaluation?
34. What kind of parameter passing does it do? (name, value, reference, value return, ...)
35. In what order are parameters to a function evaluated? Left to right? right to left? middle outwards? not guaranteed?
36. Does it use short circuit evaluation? Example?
37. What is a BNF grammar for it? (feel free to print at most one page if it is long and not the whole thing)
38. Does it differentiate between/allow use of statements or expressions?
39. Does it diferentiate between/allow use of procedures or functions?
40. Is it dynamically or statically scoped?
41. Does it use infix, prefix, postfix, mixfix operators? function calls? some combo of them? (For instance Java uses infix binary operators (a+b) and some pre/postfix unary operators (++i, i++), and prefix function calls foo(a,b,c))

42. What can you overload? Can you overload function names? Can you overload operators yourself? Does the language come with any overloaded operators? Can you overload anything wierd? (At least one language lets you overload the numeric literals (make  $5 = 8$ ))
43. What control structures does it use? Does it have any Java doesn't (ones Java can't manage some other way)?
44. Does it claim to be portable like Java? (Carefully define portable as Java uses the term. <http://en.wikipedia.org/wiki/Cross-platform> ) How portable is it?
45. What makes this language nifty? (For instance, Java fixes many of the unsafe features of C++ in the zillion ways we are covering in class). What strange things does it do?
46. If you could and had to change one thing about this language, what would you change? What would the result be?