



Programming Languages



Programming languages differ in...

- ▶ Style
- ▶ Ease
- ▶ Speed (Runtime)
- ▶ Philosophy
- ▶ Resources



C, Python, LISP

- ▶ **C**
 - ▶ very fast, elegant memory control
- ▶ **Python/LISP**
 - ▶ don't worry about types, automatic gc
- ▶ **Python**
 - ▶ very neat, easy to read (oo if needed)
- ▶ **LISP**
 - ▶ can be hard to read, but fully functional



▶ **What is**

- ▶ Imperative
- ▶ Object-oriented
- ▶ Functional

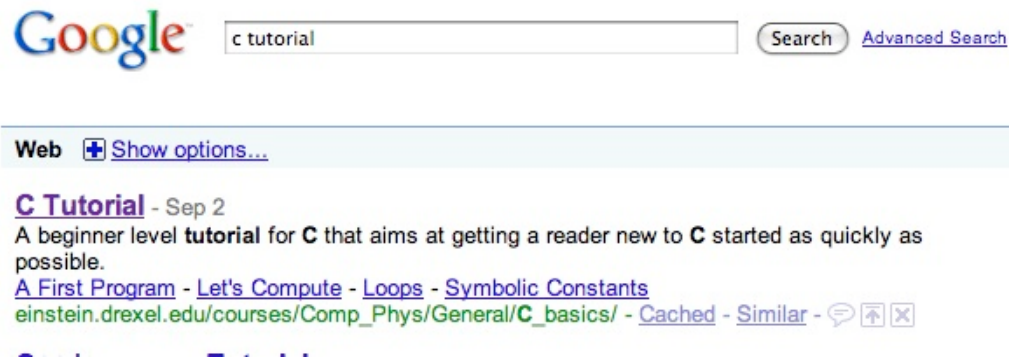
▶ **Differences between**

- ▶ Compiled code
- ▶ Interpreter-based language



On-demand information

- ▶ I don't know how to...
 - ▶ GO TO GOOGLE!



- ▶ http://einstein.drexel.edu/courses/Comp_Phys/General/C_basics/
- ▶ python.org
- ▶ <http://mypage.iu.edu/~colallen/lp/node3.html>



On-demand information

- ▶ I don't know how to...
 - ▶ if you are offline, make sure you have reference manuals offline



- ▶ Download Python/C reference manuals
- ▶ Lispworks comes with a reference manual already



On-demand information

- ▶ I don't know how to...
 - ▶ finding what you are looking for in a reference manual [when you don't know exactly what you are looking for]
 - ▶ get creative (not really)
 - ▶ check tutorials
 - ▶ look at other people's code

Imagine you needed to remove something from a list, what function would you be looking for?

- [real](#)
- [realp](#)
- [realpart](#)
- [reduce](#)
- [reinitialize-instance](#)
- [rem](#)
- [remf](#)
- [remhash](#)
- [remove](#)
- [remove-duplicates](#)
- [remove-if](#)
- [remove-if-not](#)
- [remove-method](#)
- [remprop](#)
- [rename-file](#)
- [rename-package](#)
- [replace](#)
- [require](#)
- [rest](#)



To download...

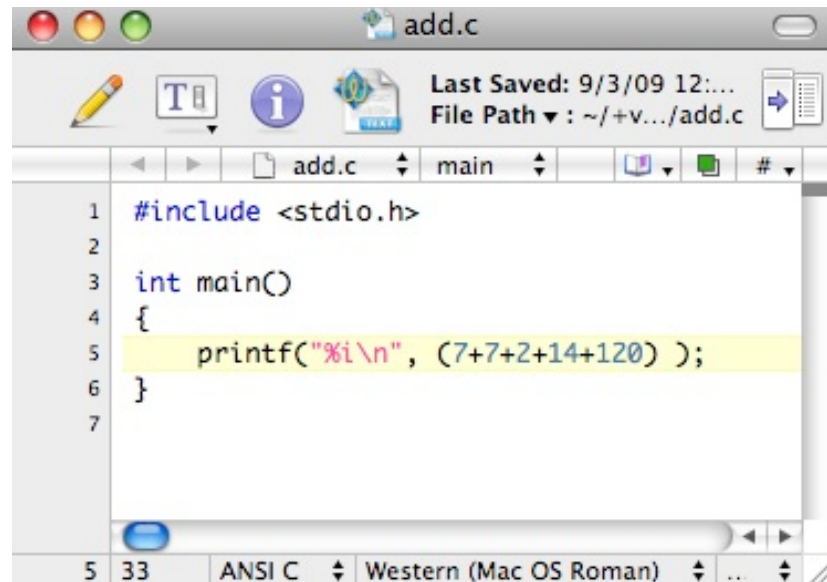
- ▶ Python interpreter (you probably already have it)
 - ▶ python.org (get 2.5)
- ▶ Python SciPy library
 - ▶ scipy.org
- ▶ LISP interpreter
 - ▶ lispworks.com (personal version is free)
- ▶ C compiler (you probably already have it)
 - ▶ <http://gcc.gnu.org>
- ▶ Text editing (for C/Python code)
 - ▶ TextWrangler for Mac
 - ▶ Windows?
- ▶ This class is not meant to test your IT skills, if you are having trouble compiling/interpreting code, ask your classmates, ask the helpdesk, ask me.



7+7+2+14+120

▶ C

- ▶ create file =====>
- ▶ compile
 - ▶ gcc add.c -o add.out
- ▶ run
 - ▶ ./add.out



The screenshot shows a text editor window titled 'add.c'. The code is as follows:

```
1 #include <stdio.h>
2
3 int main()
4 {
5     printf("%i\n", (7+7+2+14+120) );
6 }
7
```

The code is highlighted in yellow. The status bar at the bottom shows '5 33 ANSI C Western (Mac OS Roman)'.

▶ LISP

- ▶ open Lispworks
- ▶ type code
 - ▶ (+ 7 7 2 14 120)

▶ Python

- ▶ start Python, type code
 - ▶ python
 - ▶ 7+7+2+14+120



C

```
1 #include <stdio.h>
2 #include <time.h>
3
4 int foo(i,j){
5     if(i>j){
6         return i+j;
7     } else if (i<j) {
8         return i-j;
9     } else {
10        return i*j;
11    }
12 }
13
14 void printmatrix(int m[][10]){
15     int i,j;
16     for (i=1; i<10; i++){
17         for (j=1; j<10; j++){
18             printf("%2d ", m[i][j]);
19         }
20         printf("\n");
21     }
22 }
23
24 int main()
25 {
26     clock_t start = clock(); //we are going to check how long this takes
27
28     int m[10][10];
29     int i,j;
30
31     for (i=1; i<10; i++){
32         for (j=1; j<10; j++){
33             m[i][j]=foo(i,j);
34         }
35     }
36
37     printmatrix(m);
38
39     printf("\nTime elapsed: %f\n", ((double)clock() - start) / CLOCKS_PER_SEC);
40 }
```

external libraries

function returns int

procedure

matrix declaration

[after declarations]
execution start here

C

- ▶ **pointers**

- ▶ <http://www.ontko.com/~rayo/cs35/pointers.html>

- ▶ **typedef**

- ▶ **typedef struct**

- ▶ **malloc(), sizeof(), free()**

- ▶ **files**

- ▶ `char str[50]; FILE *fp;`

- ▶ `fp=fopen("myfile.txt","r");`

- ▶ `fscanf(fp, "%s", str)`



Python

```
1 import timeit
2
3 def foo(i,j):
4     if i>j:
5         return i+j
6     elif i<j:
7         return i-j
8     else:
9         return i*j
10
11 def printmatrix(m):
12     for i in m:
13         for j in i:
14             print "%2d" % j,
15             print
16
17 def main():
18     m=[]
19
20     for i in range(10):
21         m.append([])
22         for j in range(10):
23             m[i].append(foo(i,j))
24     printmatrix(m)
25
26 t=timeit.Timer('main()', 'from __main__ import main')
27 print t.timeit(1) #this will run main() and report how long it took
```



Python

```
1 import timeit
2 import numpy
3
4 def foo(i,j):
5     if i>j:
6         return i+j
7     elif i<j:
8         return i-j
9     else:
10        return i*j
11
12 def printmatrix(m):
13     for i in m:
14         for j in i:
15             print "%2d" % j,
16             print
17
18 def main():
19     m=numpy.zeros((10,10), dtype=int)
20
21     for i in range(10):
22         for j in range(10):
23             m[i,j]=foo(i,j)
24     printmatrix(m)
25
26 t=timeit.Timer('main()', 'from __main__ import main')
27 print t.timeit(1) #this will run main() and report how long it took
```



Python

▶ lists

- ▶ `a=['a','b']`
- ▶ `a.append('c')`
- ▶ `a+=range(5)*2`
- ▶ `a*=2`
- ▶ `import random`
- ▶ `random.sample(a,5)`
- ▶ `b=set(a)`

▶ loops

- ▶ `for item in a: print item`

- ▶ `c=['b'+str(i) for i in a]`

- ▶ `i=0`
- ▶ `while i<len(a):`
 - ▶ `print a[i]`

▶ dir/help

- ▶ `dir(a)`
- ▶ `dir(a[0])`

▶ files

- ▶ `f=file('tmp.txt','w')`
- ▶ `f.write('hi')`
- ▶ `f.close()`
- ▶ `f=file('tmp.txt','r')`
- ▶ `d=f.read()`
- ▶ `f.close()`
- ▶ `d`

▶ internet

- ▶ `from urllib import urlopen`
- ▶ `urlopen('http://www.cogsci.rpi.edu/')`



LISP

```
(defun foo (i j)
  (if (> i j)
      (+ i j)
      (if (< i j)
          (- i j)
          (* i j))))

(defun printmatrix (m)
  (dotimes (i 10)
    (dotimes (j 10)
      (format t "~2d " (aref m i j))
      (format t "~%" ))))

(defun main ()
  (setq m (make-array '(10 10)))
  (dotimes (i 10)
    (dotimes (j 10)
      (setf (aref m i j) (foo i j))))
  (printmatrix m))

(time (main))
```



LISP

▶ lists

- ▶ `(setq a ())`
- ▶ `(push 'a a)`
- ▶ `(push "a" a)`
- ▶ `(setq a (append a '(1 2 3)))`
- ▶ `(nconc a '(1 2 3))`
- ▶ `(pop a)`
- ▶ `(car a)`
- ▶ `(apply '+ (cdr a))`
- ▶ `(mapcar #'(lambda (x) (+ 2 x)) (cdr a))`
- ▶ `(loop for i in (cdr a) collect (+ 2 i))`
- ▶ `(loop for i in (cdr a) sum i)`
- ▶ `(dolist (element a) (print element))`

▶ functional

- ▶ `(if somecondition (print 1) (print 2))`
- ▶ `(print (if somecondition 1 2))`



HW

- ▶ get RMSE between each vector in a 1000x1000 matrix of rnd floats
- ▶ in C, un/compiled LISP, Python
 - ▶ for each of these, record and email me:
 - ▶ how long it took you to write the code (including debugging)
 - ▶ how many lines of [neat] code (excluding whitespace)
 - ▶ how long it took to run
 - ▶ \approx how long it took for the extracurriculars (e.g. setting up)
- ▶ *do the same thing in Python/LISP, using lists of lists, hashes of lists, hashes of hashes [instead of matrices]

