



Data Structures Discipline with Python

@fmasanori



I love teaching
CS Professor at FATEC

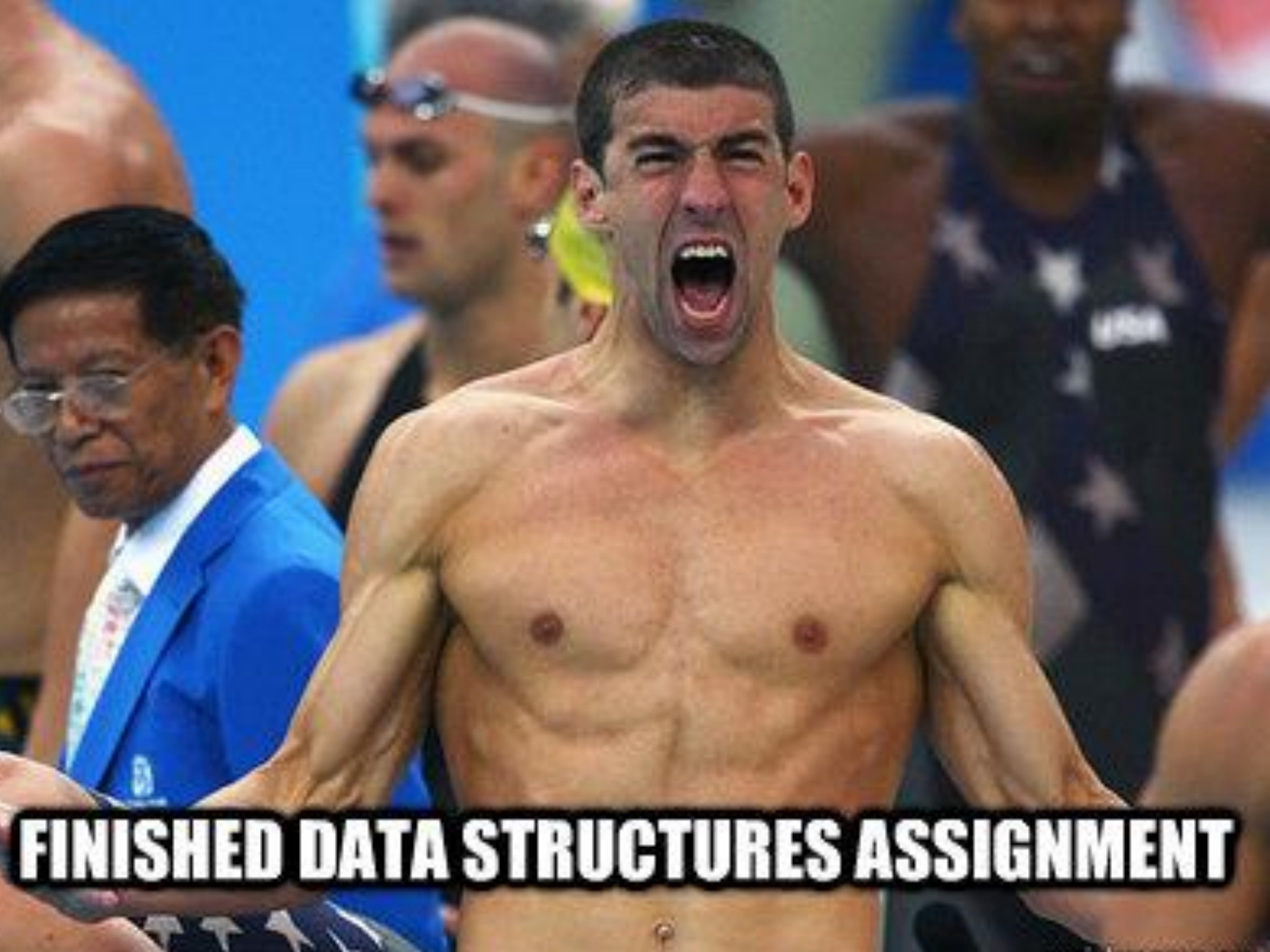
<https://about.me/fmasanori>

<http://pycursos.com/python-para-zumbis/>



A close-up of Yoda's face and upper torso. He has a green, wrinkled complexion and large, expressive eyes. He is wearing his characteristic white, shaggy robe. His hands are clasped in front of him, holding a small, brown, cylindrical object. The background is dark and out of focus.

**DATA STRUCTURES FAIL YOU
WILL**



FINISHED DATA STRUCTURES ASSIGNMENT

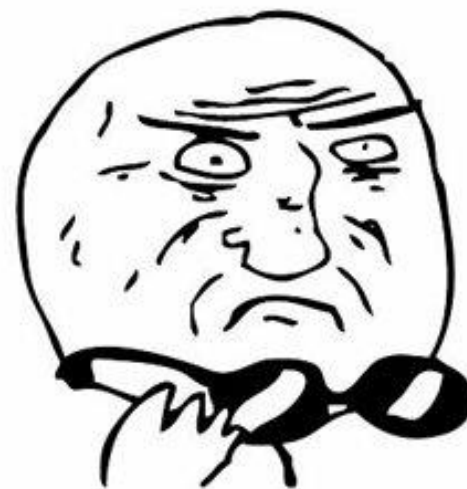
Algoritmos



Eng. de Software
















Estrutura de dados



MOTHER OF GOD

Difficult with C language

Programming then and now	Reading specification	Starting dev tools	Coding..	Coding...	Debugging	Finished.	
C							 LIKE A BOSS
Python				 *le unexpected indent	BRB		 LIKE A CHILD

But I love Data Structures



Data Structures are cool

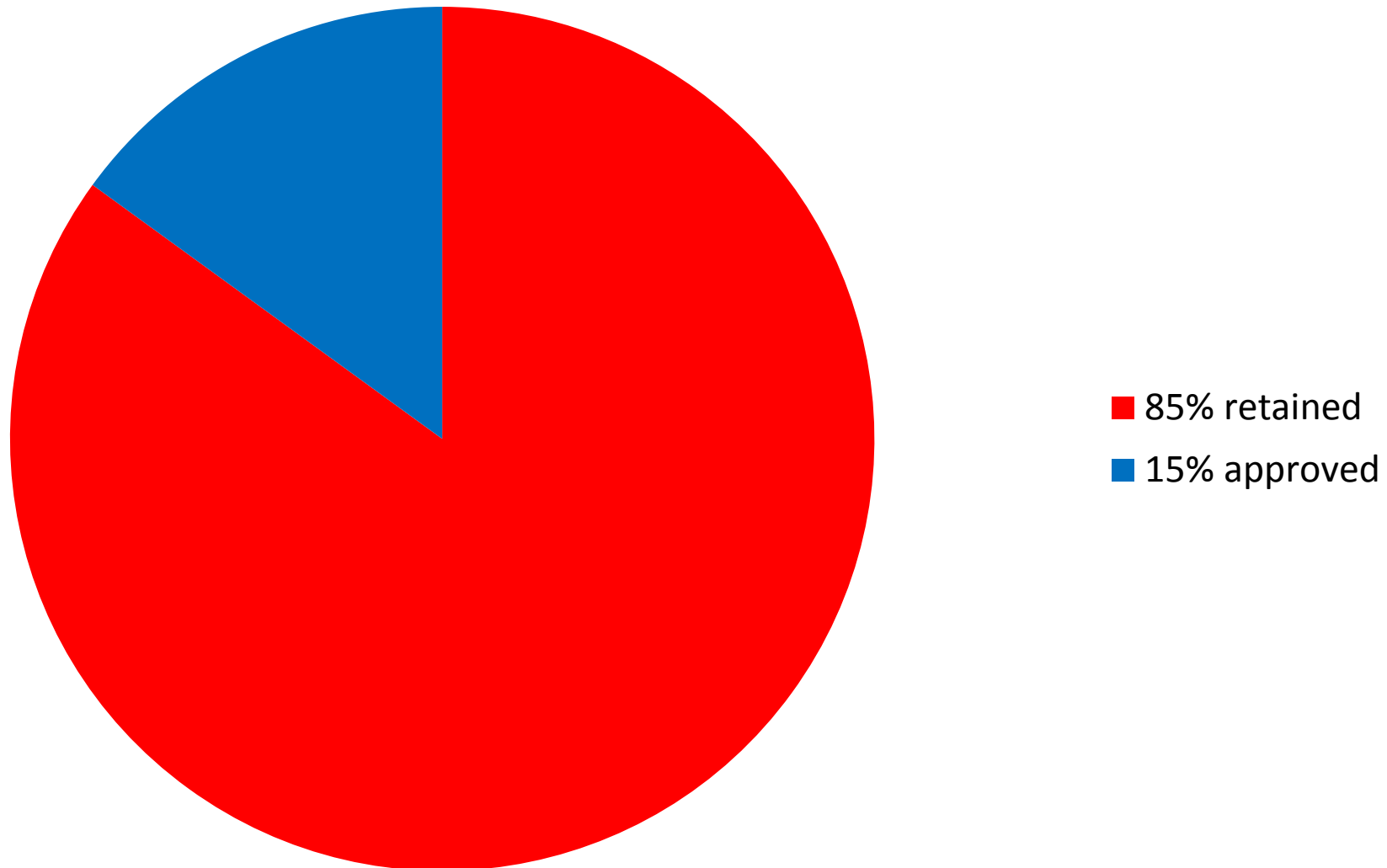


Data Structures with Python at FATEC



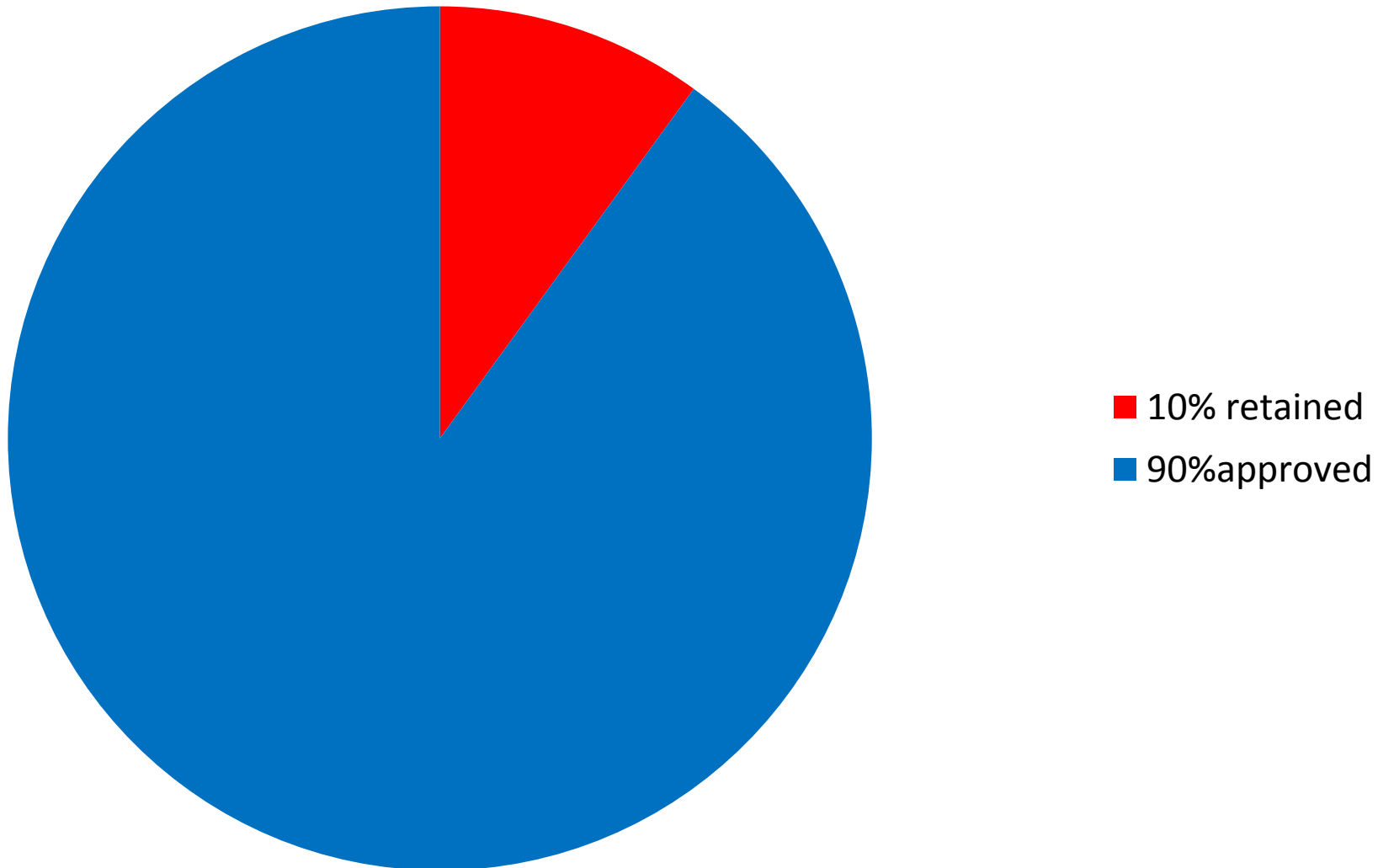
Data Structures with C (2008)

Retention 2008



Data Structures with Python (2015)

Retention 2015



Summary

Data Structures with C:

- 85% retained (2008)

Python > C:

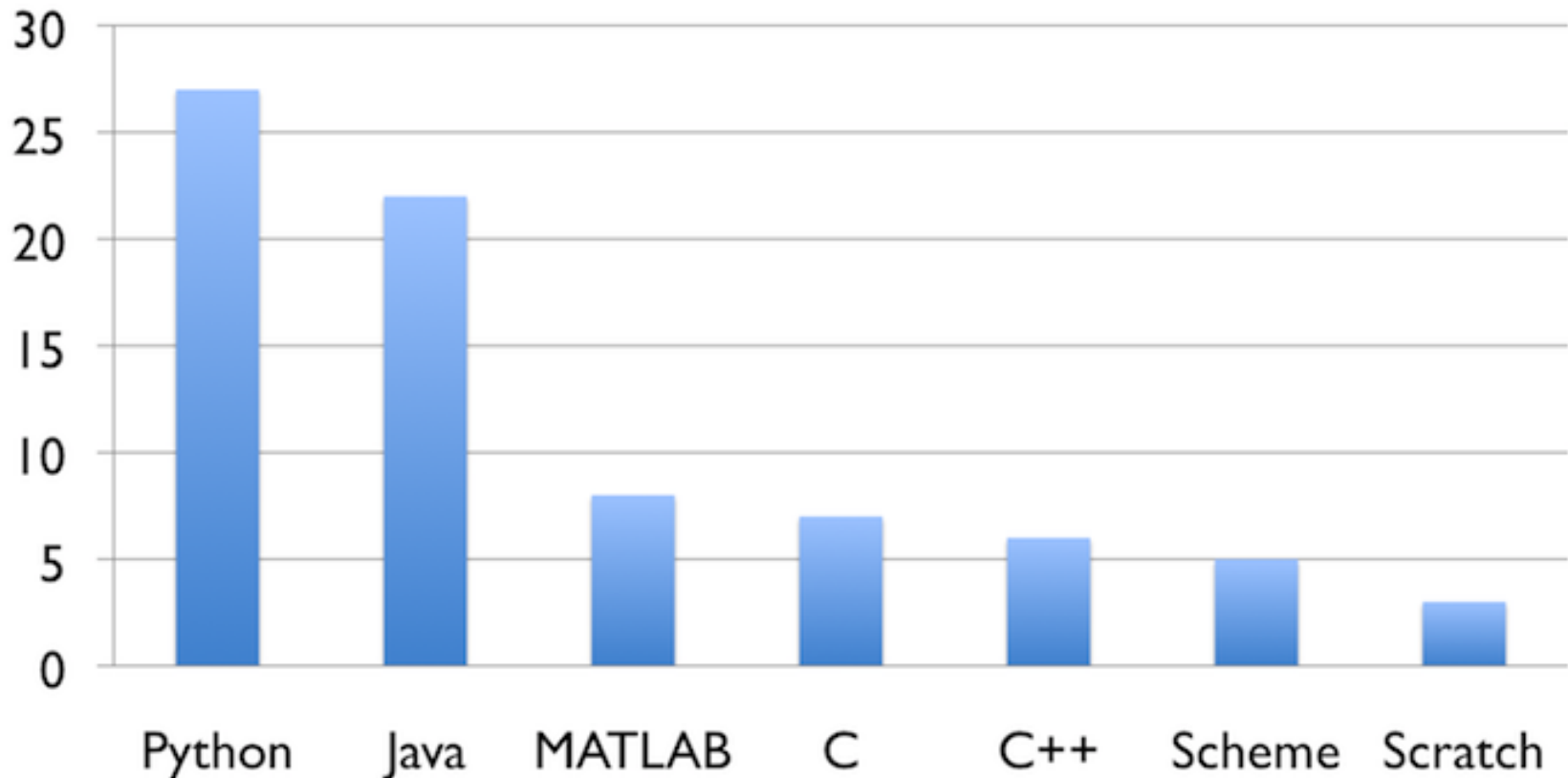
- **12%** retained (2014)
- **10%** retained (2015)
- ENADE grade 5/5 (max) (2013)
- 1st Programming Contest InterFATECs (1st/62) (2014)

Details

- Lab Only – 4 classes/week
- 4 Lab Projects (Python)
- Big Brother (some of the best students could help the other students as coaches)
- Algorithms in C (few) and Python (mainly)

Why Python?

Number of top 39 U.S. computer science departments that use each language to teach introductory courses



Analysis done by Philip Guo (www.pgbovine.net) in July 2014, last updated 2014-07-29

<http://cacm.acm.org/blogs/blog-cacm/176450-python-is-now-the-most-popular-introductory-teaching-language-at-top-us-universities/fulltext>

Usability is a problem for DS also...

"Results show that many aspects of traditional C-style syntax, while it has influenced a generation of programmers, exhibits problems in terms of usability for novices".

"Perl and Java did not accuracy rates significantly higher than a language with randomly generated keywords"

[Andreas Stefik and Susanna Siebert: "An Empirical Investigation into Programming Language Syntax." ACM Transactions on Computing Education, 13\(4\), Nov. 2013.](#)



The most common fault in computer classes is to emphasize the rules of specific programming languages, instead of to emphasize the algorithms that are being expressed in those languages. D. Knuth interview at People of ACM, June, 2014.



**Talk is cheap.
Show me the code.**

Linus Torvalds

Variables are just names (references)

```
>>> a = 42
```

```
>>> id(a)
```

```
1518584480
```

```
>>> id(42)
```

```
1518584480
```

```
>>> a = 'Python'
```

```
>>> id(a)
```

```
15542496
```

References == “pointers”

```
>>> a = [1, 2, 3]
>>> b = a
>>> id(a)
48767184
>>> id(b)
48767184
>>> a[0] = 42
>>> a
[42, 2, 3]
>>> b
[42, 2, 3]
```

```
>>> a = [4, 5, 6]
>>> b = list(a)
>>> id(a)
48772240
>>> id(b)
49151472
>>> a[0] = 42
>>> a
[42, 5, 6]
>>> b
[4, 5, 6]
```

Big integers

```
>>> 2 ** 1024
```

```
1797693134862315907729305190789024733617976978942  
3065727343008115773267580550096313270847732240753  
6021120113879871393357658789768814416622492847430  
6394741243777678934248654852763022196012460941194  
5308295208500576883815068234246288147391311054082  
7237163350510684586298239947245938479716304835356  
329624224137216
```

```
>>>
```


Natural integer division

```
>>> 1 / 2  
0.5
```

```
1 #include <stdio.h>  
2  
3 int main(void) {  
4     printf ("%f\n", 1 / 2);  
5     system ("pause");  
6 }
```

D:\Aulas\ED\divisao inteiros.exe

0.000000

Pressione qualquer tecla para continuar. . .

Multiple assignment

```
>>> a = 42
>>> b = 'avocado'
>>> a, b = b, a
>>> a
'avocado'
>>> b
42
>>> name, share, price, (year, month, day) = ['ACME', 50, 91.1,
(2015, 12, 21)]
>>> first, *middle, last = [-1, 1, 2, 3, 4, 5, -1]
>>> name, email, *fones = ('masanori', 'fmasanori@gmail.com', '
3923-3858', '8113-5934', '3905-4851')
```

Indentation

"The programming activity should be viewed
as a process of creating works of literature,
written to be read. "

--D.E. Knuth

Indentation in C

```
1  for (i = 0; i < 10; i++);  
2      printf ("Ten times Hello World!");  
3  
4  if (x < y)  
5      if ((pred(x)))  
6          printf("One");  
7  else if (x == y)  
8          printf("Two");  
9  else  
10     printf("Tree");
```

Identication in C

```
629     if ((err = SSLHashSHA1.update(&hashCtx, &signedParams)) != 0) {
630         goto fail;
631     }
632     if ((err = SSLHashSHA1.final(&hashCtx, &hashOut)) != 0)
633         goto fail;
634     err = sslRawVerify(ctx,
635                        ctx->peerPubKey,
636                        dataToSign,
637                        dataToSignLen,
638                        signature,
639                        signatureLen);
640     if(err) {
641         sslErrorLog("SSLDecodeSignedServerKeyExchange: sslRawVerify "
642                    "returned %d\n", (int)err);
643         goto fail;
644     }
645 }
646 fail:
647     SSLFreeBuffer(&signedHashes);
648     SSLFreeBuffer(&hashCtx);
649     return err;
650
651
652 }
```

SSL public key verification
added and removed here

but mostly removed
!!

Recursion

"To understand recursion, one must first understand recursion."

--folklore

"To solve the problem, I found barriers within barriers. So, I adopted a recursive solution. "

--a student

Ref.: Feofiloff, P., Algoritmos em C, Editora Campus, 2009.

Recursion

```
def fib(n):  
    print ('fib(%d)' % n)  
    if n <= 2:  
        return 1  
    #bad O(2**n)  
    return fib(n-1) + fib(n-2)
```

```
print (fib(5))
```

```
>>>
```

```
fib(5)
```

```
fib(4)
```

```
fib(3)
```

```
fib(2)
```

```
fib(1)
```

```
fib(2)
```

```
fib(3)
```

```
fib(2)
```

```
fib(1)
```

```
5
```

Recursion

```
fibcache = {}  
def fib(n):  
    if n <= 2:  
        return 1  
    if n in fibcache:  
        return fibcache[n]  
    fibcache[n] = fib(n-1) + fib(n-2)  
    return fibcache[n]  
  
print (fib(100))  
  
>>>  
354224848179261915075
```

Recursion

```
from functools import lru_cache
@lru_cache(maxsize=None)
def fib(n):
    if n <= 2:
        return 1
    else:
        return fib(n-1) + fib(n-2)

print (fib(100))

>>>
354224848179261915075
```

Recursion (student solution for dec2bin problem)

```
def dec2bin(n) :  
    if n == 0:  
        return ''  
  
    return dec2bin(n//2) + str(n%2)  
  
print (dec2bin(18))  
  
>>>  
10010
```

Linked Lists

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  struct cel {
5      int cargo;
6      struct cel *next;
7  };
8
9  typedef struct cel celula;
10
11 void Print (celula *lst) {
12     celula *p;
13     for (p = lst->next; p != NULL; p = p->next)
14         printf ("%d\n", p->cargo);
15 }
16
```


Linked Lists

```
17 void Insert (int y, celula *p) {  
18     celula *nova;  
19     nova = malloc (sizeof (celula));  
20     nova->cargo = y;  
21     nova->next = p->next;  
22     p->next = nova;  
23 }  
24  
25 int main (void){  
26     celula head;  
27     celula *lst;  
28     lst = &head;  
29     head.next = NULL;  
30     Insert (3, lst);  
31     Insert (2, lst);  
32     Insert (1, lst);  
33     Print (lst);  
34     system ("pause");  
35 }
```

Linked Lists

```
class Node:
    def __init__(self, cargo=None, next=None):
        self.cargo = cargo
        self.next = next
    def __str__(self):
        return str(self.cargo)

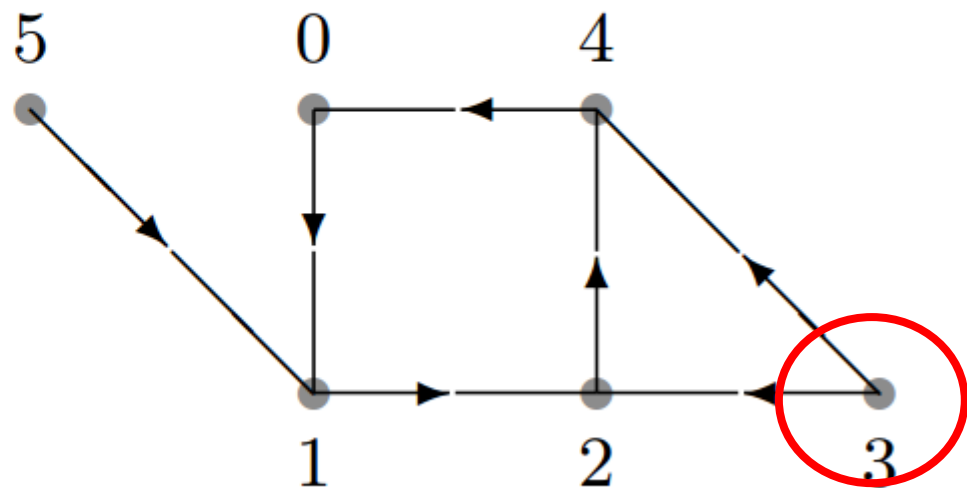
def print_list(node):
    while node is not None:
        print(node, end=" ")
        node = node.next
    print()

node1 = Node(1)
node2 = Node(2)
node3 = Node(3)
node1.next = node2
node2.next = node3
print_list(node1)

>>>
1 2 3
```

FIFOs: Distance in Networks

	0	1	2	3	4	5
0	0	1	0	0	0	0
1	0	0	1	0	0	0
2	0	0	0	0	1	0
3	0	0	1	0	1	0
4	1	0	0	0	0	0
5	0	1	0	0	0	0



	0	1	2	3	4	5
d	2	3	1	0	1	6

FIFOs: Distance in Networks

```
1  #define TAM 6
2  int A[TAM][TAM] = {{0, 1, 0, 0, 0, 0},
3                     {0, 0, 1, 0, 0, 0},
4                     {0, 0, 0, 0, 1, 0},
5                     {0, 0, 1, 0, 1, 0},
6                     {1, 0, 0, 0, 0, 0},
7                     {0, 1, 0, 0, 0, 0}};
8  int *Distancias (int n, int o) {
9      int *d, x, y;
10     int *f, s, t;
11     d = malloc (n * sizeof (int));
12     for (x = 0; x < n; x++) d[x] = -1;
13     d[o] = 0;
14     f = malloc (n * sizeof (int));
15     s = 0; t = 1; f[s] = o;
16     while (s < t) {
17         x = f[s++];
18         for (y = 0; y < n; y++)
19             if (A[x][y] == 1 && d[y] == -1) {
20                 d[y] = d[x] + 1;
21                 f[t++] = y;
22             }
23     }
24     free (f);
25     return d;
26 }
```

FIFOs: Distance in Networks

```
A = [[0, 1, 0, 0, 0, 0],  
      [0, 0, 1, 0, 0, 0],  
      [0, 0, 0, 0, 1, 0],  
      [0, 0, 1, 0, 1, 0],  
      [1, 0, 0, 0, 0, 0],  
      [0, 1, 0, 0, 0, 0]]
```

```
def Distancias(n, origen):  
    d = [-1] * n  
    d[origen] = 0  
    f = []  
    f.append(origen)  
    while len(f) > 0:  
        x = f[0]  
        del f[0]  
        for y in range(n):  
            if A[x][y] == 1 and d[y] == -1:  
                d[y] = d[x] + 1  
                f.append(y)  
    return d
```


Stacks: well-formed expression

```
1 int BemFormada (char s[]) {
2     char *p; int t;
3     int n, i;
4     n = strlen (s);
5     p = malloc (n * sizeof (char));
6     t = 0;
7     for (i = 0; s[i] != '\0'; i++) {
8         /* p[0..t-1] é uma pilha */
9         switch (s[i]) {
10             case ')': if (t != 0 && p[t-1] == '(') --t;
11                       else return 0;
12                       break;
13             case '}': if (t != 0 && p[t-1] == '{') --t;
14                       else return 0;
15                       break;
16             default: p[t++] = s[i];
17         }
18     }
19     free (p);
20     return t == 0;
21 }

22
23 int main (void) {
24     printf ("%s\n", BemFormada ("({(){()})") ? "Bem formada" : "Mal formada");
25     printf ("%s\n", BemFormada ("({})") ? "Bem formada" : "Mal formada");
26     system ("pause");
27 }
```

Stacks: well-formed expression

```
def BemFormada(s):  
    p = []  
    for c in s:  
        if c == ')':  
            if p[-1] == '(':  
                p.pop()  
            else:  
                return False  
        elif c == '}':  
            if p[-1] == '{':  
                p.pop()  
            else:  
                return False  
        else:  
            p.append(c)  
    return True  
  
print (BemFormada(' ( ( ) { ( ) } ) '))  
print (BemFormada(' ( { } ) '))
```

Selection Sort

```
1 void Selecao (int n, int v[]) {
2     int i, j, k, min, x;
3     for (i = 0; i < n-1; i++) {
4         min = i;
5         for (j = i+1; j < n; j++)
6             if (v[j] < v[min])
7                 min = j;
8         x = v[i];
9         v[i] = v[min];
10        v[min] = x;
11    }
12 }
13 int main(void){
14     int i;
15     int v[10]={7, 4, 3, 9, 0, 8, 5, 2, 6, 1};
16     Selecao (10, v);
17     for (i = 0; i < 10; i++)
18         printf ("%d", v[i]);
19     putchar('\n');
```

Selection Sort

```
def selection(v):  
    resp = []  
    while v:  
        m = min(v)  
        resp.append(m)  
        v.remove(m)  
    return resp  
  
import random  
v = list(range(10))  
random.shuffle(v)  
v = selection(v)  
print (v)
```

Quicksort

```
1 int Divide (int p, int r, int v[]) {  
2     int c, j, k, t;  
3     c = v[r]; j = p;  
4     for (k = p; k < r; k++)  
5         if (v[k] <= c) {  
6             t = v[j], v[j] = v[k], v[k] = t;  
7             j++;  
8         }  
9     v[r] = v[j], v[j] = c;  
10    return j;  
11 }  
12 void Quicksort (int p, int r, int v[]) {  
13     int j;  
14     if (p < r) {  
15         j = Divide (p, r, v);  
16         Quicksort (p, j - 1, v);  
17         Quicksort (j + 1, r, v);  
18     }  
19 }
```


Quicksort

```
def quicksort(v):  
    if len(v) <= 1:  
        return v  
  
    pivot = v[0]  
    equals  = [x for x in v if x == pivot]  
    smaller = [x for x in v if x <  pivot]  
    higher  = [x for x in v if x >  pivot]  
    return quicksort(smaller) + equals + quicksort(higher)  
  
print (quicksort([5, 7, 9, 3, 4, 0, 2, 1, 6, 8]))
```

Word Count

```
with open('alice.txt') as arq:
    texto = arq.read()
    texto = texto.lower()
import string
for c in string.punctuation:
    texto = texto.replace(c, ' ')
texto = texto.split()

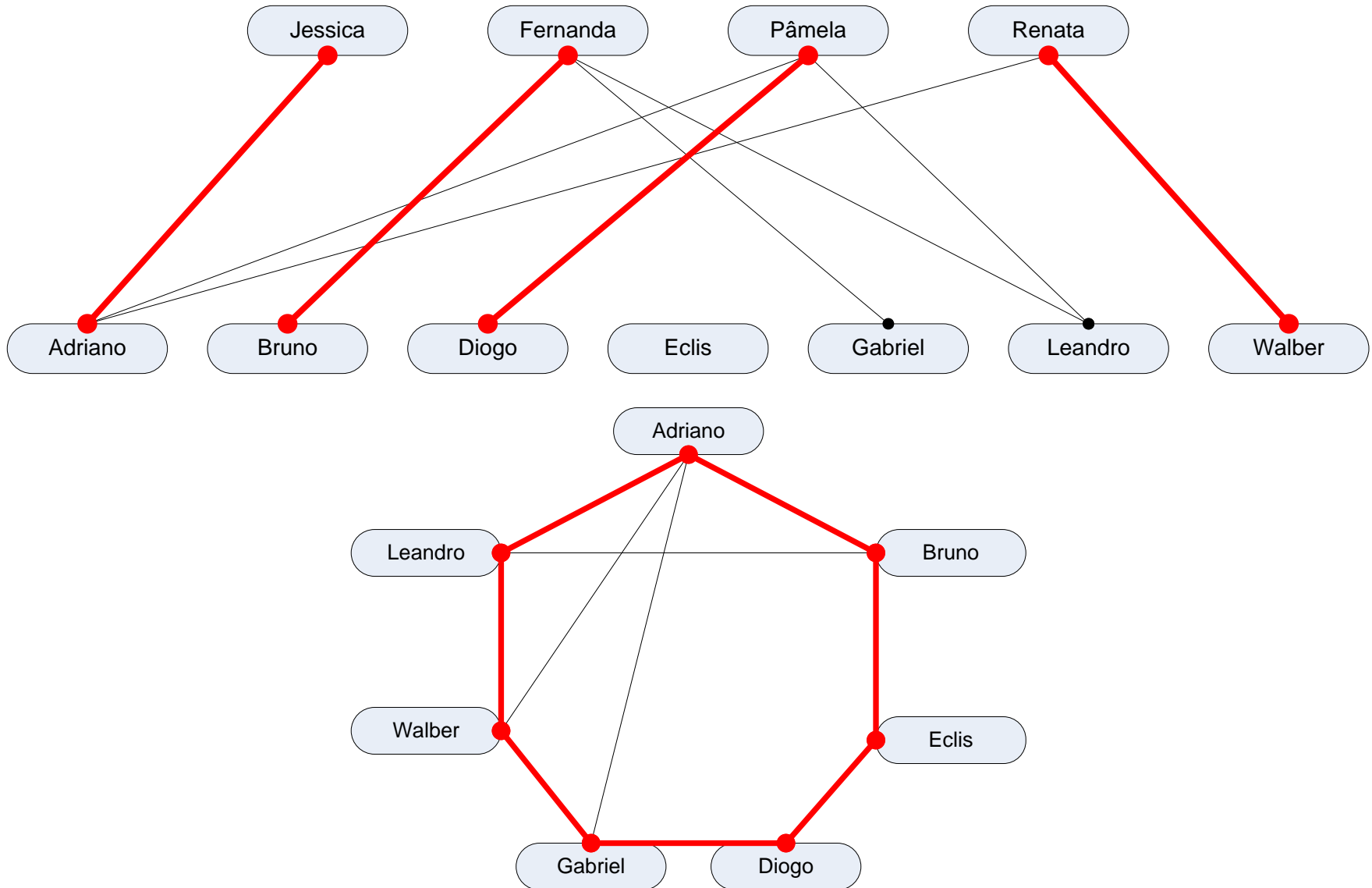
dic = {}
for p in texto:
    if p not in dic:
        dic[p] = 1
    else:
        dic[p] += 1
```

Projects: Sorting Algorithm comparison

		time (s)				

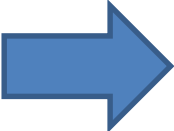
		Mergesort	Quicksort	Selection	Native	
2000		0.05	0.00	0.40	0.00	
4000		0.05	0.02	1.65	0.00	
6000		0.05	0.03	3.64	0.00	
8000		0.07	0.02	6.49	0.00	
10000		0.10	0.03	10.16	0.01	
12000		0.10	0.05	14.69	0.00	
14000		0.13	0.06	19.90	0.02	
16000		0.14	0.06	26.55	0.01	
18000		0.16	0.08	32.94	0.02	
20000		0.20	0.08	42.20	0.02	
22000		0.23	0.08	50.67	0.02	

Projects: Arthur Merlin Games simplification

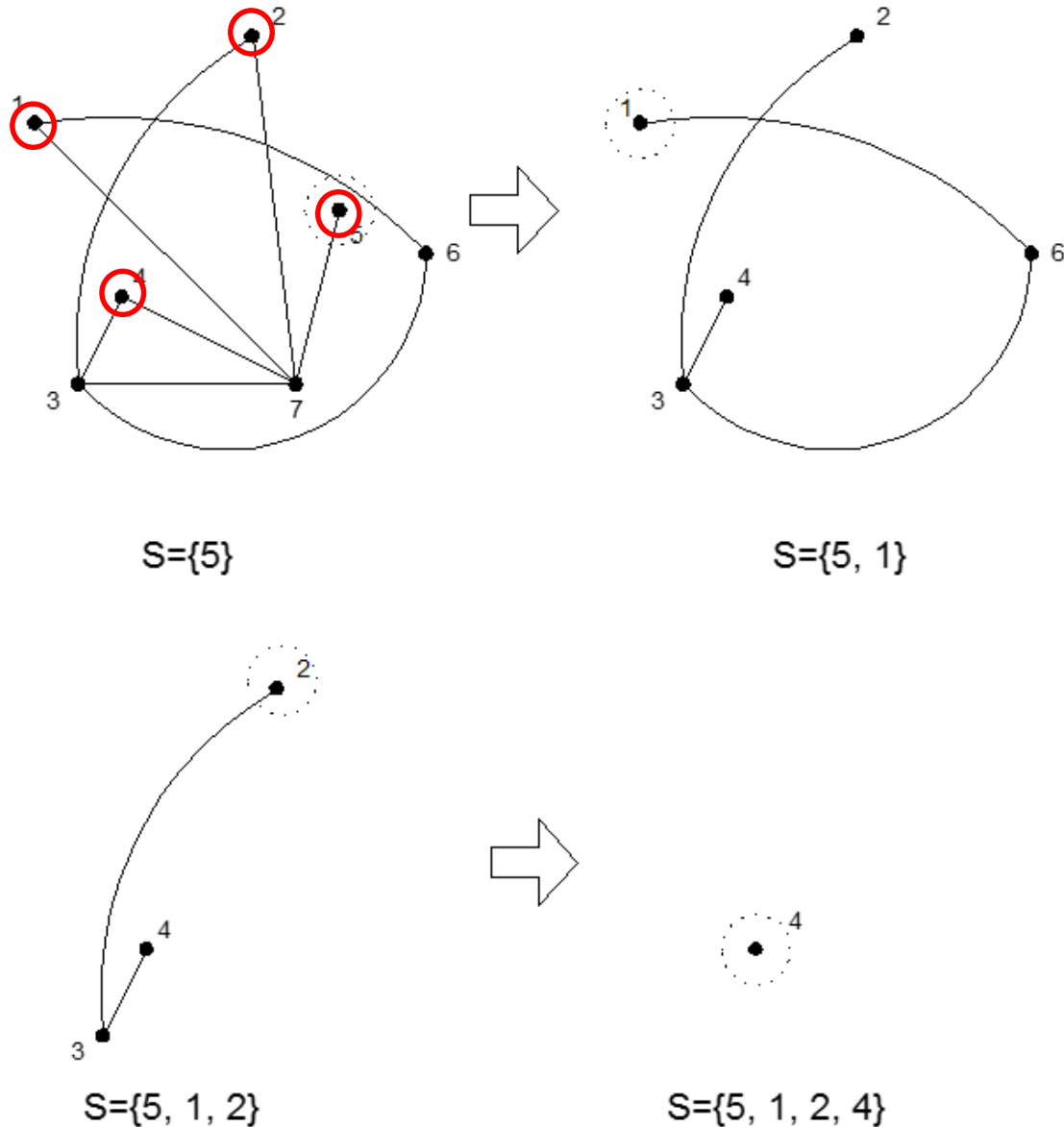


Projects: Binaries regions

```
teste3 = '''0011001010      0011002020
             0110001010      0110002020
             0011001110      0011002220
             0000000000      0000000000
             0010001010      0030004040
             0010011111      0030044444
             1111100000      3333300000
             0010001110      0030005550
             0010001110      0030005550
             ...
```



Projects: Minimum Degree Greedy Heuristic for MIS



Conclusions

- C is good for optimization (details, low level)
- Python is good to show the essence of the algorithms (readability, high level)
- If the algorithm is the same (complexity) “premature optimization is evil” also in teaching Data Structures.

Questions?

gist.github.com/fmasanori

<http://about.me/fmasanori>

fmasanori@gmail.com

Slides: bit.ly/python-DS