Taejoon Kwon

University of Texas at Austin

Xenopus Bioinformatics Workshop, May 2014

http://www.ebay.co.uk/itm/I-Hate-Programming-T-shirt-PC-IT-Computer-Code-Programmer-Slogan-S-XXL-6-cols-/121326428845

Programmers

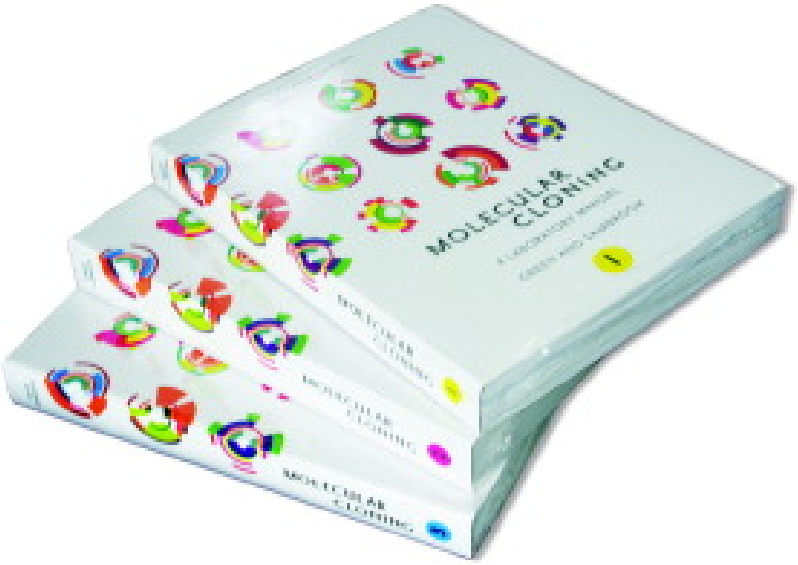
21st century magicians

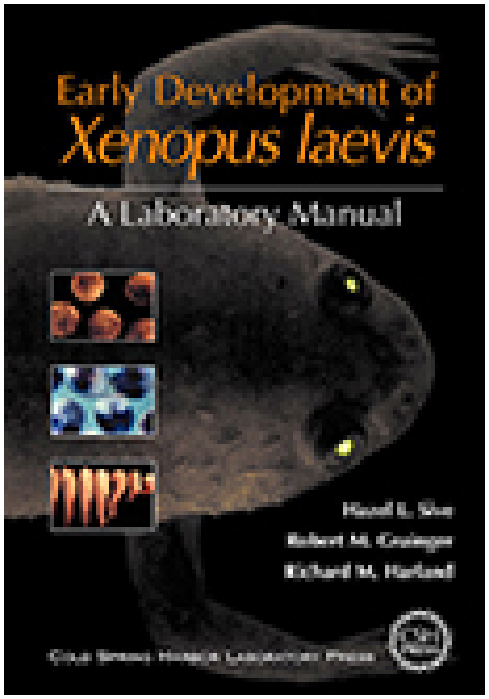
http://37.media.tumblr.com/tumblr_ly0dpda7ns1r7t60no1_500.jpg

http://rlv.zcache.com/i_am_a_programmer_not_a_magician_t_shirts-r9395444599414ccc8b320b64c89a13a8_wig7n_512.jpg

(OR BIOLOGIST)

http://rlv.zcache.co.uk/i_am_a_biochemist_not_a_magician_mousemat-r6acca9a78acd4ea883e32229b9d6c8f6_x74vi_8byvr_324.jpg





http://www.homeschool-activities.com/images/slime-potion.jpg

Too much recipes!?!?

http://www.lackuna.com/wp-content/uploads/2013/01/programming.jpg

My personal journey in programming

- ~ 1994: C (and little bit of BASIC)
- 1994 ~ 1995: Fortran(!)
- 1996 ~ 1998: Visual C++ (and little bit of Java)
- 1998 ~ 2008: PERL
  - 2002 ~ 2005: Java (in the company)
  - 2005 ~ 2006: R & MATLAB (MPhil in Comp. Bio)
- 2008 ~ current: Python
  - & little bit of R, PHP5, JavaScript, C# & Ruby

Programming language – my opinion

- C/C++ : Most powerful. Period.
  - Need to know a lot about computer itself (i.e. memory allocation).
  - Steep learning curve (even you know another language).
- Java & C# : Powerful & comprehensive.
  - Need to understand ‘object-orient programming’.
  - Ideal for ‘huge project’, but too ‘heavy’ to use in small tasks.
- JavaScript & PHP: A language for the web. Limited.
- Unix shell scripting (BASH, TCSH): A language for the command line. Limited.

Programming language – my opinion

- PERL: Powerful in text manipulation
  - Check out Lincoln Stein's article "How PERL save the human genome" in the wiki.
  - Still widely used in bioinformatics (i.e. EnsEMBL, GBrowse)
- Ruby: "New Kids On The Block"
  - Hybrid of PERL (flexibility) and Python (object oriented structure); but little bit premature yet.
- MATLAB: Powerful in machine learning & statistics.
  - Expensive (many institutes may have a site license, though)
- R: Powerful in statistics.
  - Little bit 'strange' syntax; steep learning barrier.

- If you can do it with python,
  - You can do it with PERL
  - You can do it with Ruby
  - You can do it with MATLAB or R
  - You can do it with C or C++
  - You can do it with Java or C#
  - (but may not with BASH, JavaScript, PHP)
- Just pick any of them, learn it, and use it everyday. Soon you will become a programmer (or a magician).
- Don’t be stressed to google it when you have a question. All programmers also do it.
  - It is same to check a protocol before the experiment. I don’t believe any biologist can memorize all parts of “Molecular Cloning” or “*Xenopus* handbook”.

Why Python?

- Compared to C/C++/Java/C#
  - Easier to learn.
  - More suitable for ‘simple tasks’ that we are interested in.
- Compared to Ruby
  - More mature (personal opinion).
- Compared to PERL
  - Easier to organize codes (more object-oriented).
  - All-in-one package (free from module dependency).
  - Bioinformatics community with python is getting bigger.
  - Useful libraries: numpy/scipy & matplotlib
  - Personally I don’t want to go back to PERL.
- Python3 has some good features, but many libraries do not support it yet. We will use python-2 instead here.

Two ways to work with python

- Traditional way
  - Write a code with your favorite text editor.
  - Run the code with '> python <my_code.py>' command.
- Interactive way
  - Execute '> python' in your command terminal.
  - Do the programming inside 'interactive' shell.
  - Check out ipython & its notebook function at http://ipython.org/notebook.html
- Find more comfortable way for yourself.

Ok, let’s get to work!

http://www.maniacworld.com/get-to-work.jpg

- Make your code beautiful.
  - It is like to “make your bench clean”. Nobody wants to do the experiment in dirty bench, even it is actually YOU who to make all those messes.
- If you don’t have any preference, just follow well-established coding style. There is reasons for this style, and you may know them in the future.
  - http://legacy.python.org/dev/peps/pep-0008/
  - http://google-styleguide.googlecode.com/svn/trunk/pyguide.html
- Spend a time before naming anything (variables, functions, filenames)
  - test1.py, test2.py, test3.py, …
  - taejoon1.py, taejoon2.py, ….

Indentation matters in python

(be aware if you have experienced in other languages)

C code

```c
int main(int argc, char **argv) {

  /* Data */
  int p, n, niter;
  DATA data;   /* all data */
  PAR par;
  char newName[256];

  const gsl_rng_type *T;
  gsl_rng *r;
  gsl_rng_env_setup();
  T = gsl_rng_default;
  r = gsl_rng_alloc(T);

  if (!(argc == 2 || argc == 4)) {
    fprintf(stderr, "usage: msblender [data] [ncompPos] [ncom
    fprintf(stderr, "usage: msblender [data] (default: ncompP
    return 1;
  }
  FILE *fp = fopen(argv[1], "r");
  if(fp == NULL) {
    fprintf(stderr, "Data file %s does not exist.\n", argv[1]
    return 1;
  }
```

Braces

Semicolon

Python code

```python
#!/usr/bin/env python
import os
import sys

if( len(sys.argv) < 3 ):
    sys.stderr.write("usage: python msblender_out-to-pep_count-mFDR
    sys.exit(1)

filename_mb_out = sys.argv[1]
FDR_cutoff = float(sys.argv[2])
FDR_string = sys.argv[2].replace('.','')

error_model = 'mFDRpsm'
error_model_list = ['eFDRpsm','mFDRpsm']
if( len(sys.argv) == 4 and sys.argv[3] in error_model_list ):
    error_model = sys.argv[3]

sys.stderr.write("FDR cutoff: %.3f\nError model:%s\n"%(FDR_cutoff,e

filename_base = filename_mb_out.replace('.msblender_in','').replace

psm_mvScore = dict()
psm_TD = dict()

f_mb_out = open(filename_mb_out,'r')
f_mb_out.readline()
for line in f_mb_out:
    tokens = line.strip().split("\t")
    tmp_mvScore = float(tokens[-1])
    tmp_psm = tokens[0]
    psm_mvScore[tmp_psm] = float(tokens[-1])
    psm_TD[tmp_psm] = tokens[1]
f_mb_out.close()
```

Indentation defines a block

Major components in programming

- Variables: “How to store data?”
  - Scalar: number, string
  - Array/List
  - Dictionary/Hash
- Control flows: “How to process data to get a result?”
  - Conditions (if … then … else …)
  - Loop (for … while …)
- Operations & functions
- Input/Output: “How to read data/write result?”

80 built-in functions

Built-in Functions				
abs()	divmod()	input()	open()	staticmethod()
all()	enumerate()	int()	ord()	str()
any()	eval()	isinstance()	pow()	sum()
basestring()	execfile()	issubclass()	print()	super()
bin()	file()	iter()	property()	tuple()
bool()	filter()	len()	range()	type()
bytearray()	float()	list()	raw_input()	unichr()
callable()	format()	locals()	reduce()	unicode()
chr()	frozenset()	long()	reload()	vars()
classmethod()	getattr()	map()	repr()	xrange()
cmp()	globals()	max()	reversed()	zip()
compile()	hasattr()	memoryview()	round()	__import__()
complex()	hash()	min()	set()	apply()
delattr()	help()	next()	setattr()	buffer()
dict()	hex()	object()	slice()	coerce()
dir()	id()	oct()	sorted()	intern()

https://docs.python.org/2/library/functions.html

Numbers: integer & float

File Edit View Kernel Magic Window Help

```
In [5]: 2+2
Out[5]: 4

In [6]: print 2+2
4

In [7]: 7/2
Out[7]: 3

In [8]: 7.0/2
Out[8]: 3.5

In [9]: 7/2.0
Out[9]: 3.5

In [10]: 7/-2
Out[10]: -4

In [11]: 7/-2.0
Out[11]: -3.5

In [12]: width = 20

In [13]: height = 5

In [14]: width*height
Out[14]: 100

In [15]: area = width*height

In [16]: print area
100
```

File Edit View Kernel Magic Window Help

```
In [19]: 5%2
Out[19]: 1

In [20]: 6%2
Out[20]: 0

In [21]: 3+2*5
Out[21]: 13

In [22]: (3+2)*5
Out[22]: 25

In [23]: 10%7
Out[23]: 3

In [24]: 28%7
Out[24]: 0

In [25]: float(7)/2
Out[25]: 3.5

In [26]: int(7.0)/2
Out[26]: 3

In [27]: a = 2

In [28]: b = 3.1415

In [29]: print "%d,%02d\t%05d"%(a,a,a)
2,02    00002

In [30]: print "%.2f %.5f %.2e"%(b,b,b)
3.14 3.14150 3.14e+00
```

https://docs.python.org/2/tutorial/introduction.html

List/Array

```
In [71]: a = ['spam','eggs',100,1234]

In [72]: a[0]
Out[72]: 'spam'

In [73]: a[3]
Out[73]: 1234

In [74]: a[-1]
Out[74]: 1234

In [75]: a[0:2]
Out[75]: ['spam', 'eggs']

In [76]: a[0:2]+['bacon',2*2]
Out[76]: ['spam', 'eggs', 'bacon', 4]

In [77]: a[::-1]
Out[77]: [1234, 100, 'eggs', 'spam']

In [78]: a[2]+100
Out[78]: 200

In [79]: len(a)
Out[79]: 4

In [80]: a = []

In [81]: a
Out[81]: []
```

Use bracket!

```
In [82]: a = [10,20] + ['chicken','egg']

In [83]: a
Out[83]: [10, 20, 'chicken', 'egg']

In [84]: a.append('frog')

In [85]: a
Out[85]: [10, 20, 'chicken', 'egg', 'frog']

In [86]: a.pop()
Out[86]: 'frog'

In [87]: a
Out[87]: [10, 20, 'chicken', 'egg']

In [88]: a.append(['hello','MBL'])

In [89]: a
Out[89]: [10, 20, 'chicken', 'egg', ['hello', 'MBL']]

In [90]: len(a)
Out[90]: 5

In [91]: a[3]
Out[91]: 'egg'

In [92]: a[4]
Out[92]: ['hello', 'MBL']

In [93]: a[4][1]
Out[93]: 'MBL'
```

```
In [140]: a = ['banana','kiwi','pear','apple']

In [141]: sorted(a)
Out[141]: ['apple', 'banana', 'kiwi', 'pear']
```

```
In [144]: b = [23,13,53,2]

In [145]: sorted(b)
Out[145]: [2, 13, 23, 53]
```

String

```
In [36]: a = 'Hello'

In [37]: print a
Hello

In [38]: b = '\tXenopus\tRocks!\t'

In [39]: print b
        Xenopus Rocks!

In [40]: print b.strip(),a
Xenopus Rocks! Hello

In [41]: print a.upper()
HELLO

In [42]: print a.lower()
hello

In [43]: print a.upper()+a.lower()
HELLOhello

In [44]: print '3'+'5'
35

In [45]: print int('3')+int('5')
8

In [46]: print "%s"%(b)
        Xenopus Rocks!

In [47]: print "xxx%syyy"%(b)
xxx     Xenopus Rocks!  yyy
```

```
In [56]: x = 'Xenopus Rocks!'

In [57]: print x
Xenopus Rocks!

In [58]: print x[2]
n

In [59]: print x[:2]
Xe

In [60]: print x[:6:-1]
!skcoR

In [61]: print x[::-1]
!skcoR suponeX

In [62]: len(x)
Out[62]: 14

In [63]: y = x.split(' ')

In [64]: print y
['Xenopus', 'Rocks!']

In [65]: print y[0]
Xenopus
```

```
In [66]: z='a\tb\tc\td\te'

In [67]: print z
a       b       c       d       e

In [68]: z_list = z.split("\t")

In [69]: print z_list
['a', 'b', 'c', 'd', 'e']

In [70]: print z_list[2]
c
```

Dictionary (a.k.a Hash) & Set

```
In [94]: tel = {'jack':4098, 'sape':4139}

In [95]: tel['quido'] = 4127

In [96]: tel
Out[96]: {'jack': 4098, 'quido': 4127, 'sape': 4139}

In [97]: print tel['jack']
4098

In [98]: del tel['sape']

In [99]: tel
Out[99]: {'jack': 4098, 'quido': 4127}

In [100]: len(tel)
Out[100]: 2

In [102]: another_tel = dict(sape=4139, guido=4127, jack=4098)

In [103]: another_tel
Out[103]: {'guido': 4127, 'jack': 4098, 'sape': 4139}

In [104]: tel = {}

In [105]: tel
Out[105]: {}

In [106]: tel['guido'] = 4127

In [107]: tel
Out[107]: {'guido': 4127}

In [108]: tel = dict()
```

```
In [112]: a = set('abracadabra')

In [113]: b = set('alacazam')

In [114]: a
Out[114]: {'a', 'b', 'c', 'd', 'r'}

In [115]: b
Out[115]: {'a', 'c', 'l', 'm', 'z'}

In [116]: a - b
Out[116]: {'b', 'd', 'r'}

In [117]: a | b
Out[117]: {'a', 'b', 'c', 'd', 'l', 'm', 'r', 'z'}

In [118]: a.union(b)
Out[118]: {'a', 'b', 'c', 'd', 'l', 'm', 'r', 'z'}

In [119]: a & b
Out[119]: {'a', 'c'}

In [120]: a.intersection(b)
Out[120]: {'a', 'c'}
```

```
In [123]: a|b - a&b
Out[123]: {'a', 'b', 'c', 'd', 'l', 'm', 'r', 'z'}

In [124]: (a|b) - (a&b)
Out[124]: {'b', 'd', 'l', 'm', 'r', 'z'}

In [125]: a^b
Out[125]: {'b', 'd', 'l', 'm', 'r', 'z'}

In [126]: (a|b) - (a&b)
Out[126]: {'b', 'd', 'l', 'm', 'r', 'z'}

In [127]: c = list(set(a&b))

In [128]: c
Out[128]: ['a', 'c']

In [129]: print c[0]
a
```

Advanced: modules

```
In [130]: def add(tmp_a, tmp_b):
     ...:     return tmp_a+tmp_b
     ...:

In [131]: add(20,50)
Out[131]: 70

In [132]: def raw_and_add(tmp_a, tmp_b):
     ...:     print tmp_a
     ...:     print tmp_b
     ...:     return tmp_a,tmp_b,tmp_a+tmp_b
     ...:

In [133]: add(20,50)
Out[133]: 70

In [134]: raw_and_add(20,50)
20
50
Out[134]: (20, 50, 70)

In [135]: in_a, in_b, in_a_and_b = raw_and_add(20,50)
20
50

In [136]: in_a
Out[136]: 20

In [137]: in_b
Out[137]: 50

In [138]: in_a_and_b
Out[138]: 70
```

Flow control: if… elif … else …

(comparison: ==, !=, >, <, >=, <=)

```
In [146]: def rock_sissors_paper(tmp):
     ...:     if tmp == 'rock':
     ...:         return 'paper'
     ...:     elif tmp == 'sissor':
     ...:         return 'rock'
     ...:     elif tmp == 'paper':
     ...:         return 'sissor'
     ...:     else:
     ...:         return "I don't understand."
     ...:

In [147]: rock_sissors_paper('rock')
Out[147]: 'paper'

In [148]: rock_sissors_paper('sissor')
Out[148]: 'rock'

In [149]: rock_sissors_paper('paper')
Out[149]: 'sissor'

In [150]: rock_sissors_paper('xenopus')
Out[150]: "I don't understand."
```

```
In [151]: def starts_with_X(tmp):
     ...:     if tmp.startswith('X'):
     ...:         return True
     ...:     else:
     ...:         return False
     ...:

In [152]: starts_with_X('Xenopus')
Out[152]: True

In [153]: starts_with_X('Zebrafish')
Out[153]: False

In [154]: if starts_with_X('Xenopus'):
     ...:     print "Awesome!"
     ...:
Awesome!

In [155]: if starts_with_X('Zebrafish'):
     ...:     print "Awesome!"
     ...: else:
     ...:     print "Boo~"
     ...:
Boo~
```

Flow control: for

```
In [164]: for animal in ['cat','frog','dog','lion']:
     ...:     if len(animal) > 3 :
     ...:         print animal,"is cool"
     ...:     else:
     ...:         print animal,"is not cool"
     ...:
cat is not cool
frog is cool
dog is not cool
lion is cool

In [165]: sum = 0

In [166]: for i in [1,2,3,4,5,6,7,8,9,10]:
     ...:     sum = sum + i
     ...:     print sum
     ...:
1
3
6
10
15
21
28
36
45
55
```

```
In [169]: for i in range(1,11):
     ...:     if i % 2 == 0:
     ...:         print "Even ",i
     ...:         continue
     ...:     if i == 7:
     ...:         break
     ...:     print i
     ...:
1
Even  2
3
Even  4
5
Even  6
```

```
In [171]: for n in range(2,10):
     ...:     for x in range(2,n):
     ...:         if n % x == 0 :
     ...:             print n, 'equals', x, '*', n/x
     ...:             break
     ...:     else:
     ...:         print n, 'is a prime number'
     ...:
2 is a prime number
3 is a prime number
4 equals 2 * 2
5 is a prime number
6 equals 2 * 3
7 is a prime number
8 equals 2 * 4
9 equals 3 * 3
```

Input & Output

- First, you need to open a file by “open()”
  - open(<filename>,**’r’**) for reading.
- Then, read contents by “read()”
  - Or use the iterator (see next slide)
- Then, close the file with “close()”

- First, you need to open a file by “open()”
  - open(<filename>,**’w’**) for writing.
- Then, write stuff by “write()”
  - f.write(“%d\n”%(my_integer))
  - sys.stdout.write() → same as print()
  - sys.stderr.write()
- Then, close the file with “close()”

Codes I have used almost everyday

```
#!/usr/bin/env python
Import os
Import sys

filename_fa = sys.argv[1]

seqlen = dict()
seq_h = ‘’
f_fa = open(filename_fa,’r’)
for line in f_fa:
    if( line.startswith(‘>’) ):
        seq_h = line.strip().lstrip(‘>’)
        seq_len[seq_h] = 0
    else:
        seq_len[seq_h] += len(line.strip())
f_fa.close()
```

```
#!/usr/bin/env python
Import os
Import sys

filename_tsv = sys.argv[1]

f_tsv = open(filename_tsv,’r’)
f_out = open(‘results.txt’,’w’)
for line in f_tsv:
    if( line.startswith(‘#’) ):
        continue
    tokens = line.strip().split(“\t”)
    if( tokens[0].upper().find(‘BMP4’) > 0 ):
        f_out.write(‘%s\t%s\n’%(tokens[0],tokens[2]))
f_tsv.close()
f_out.close()
```

REALLY advanced: regular expression

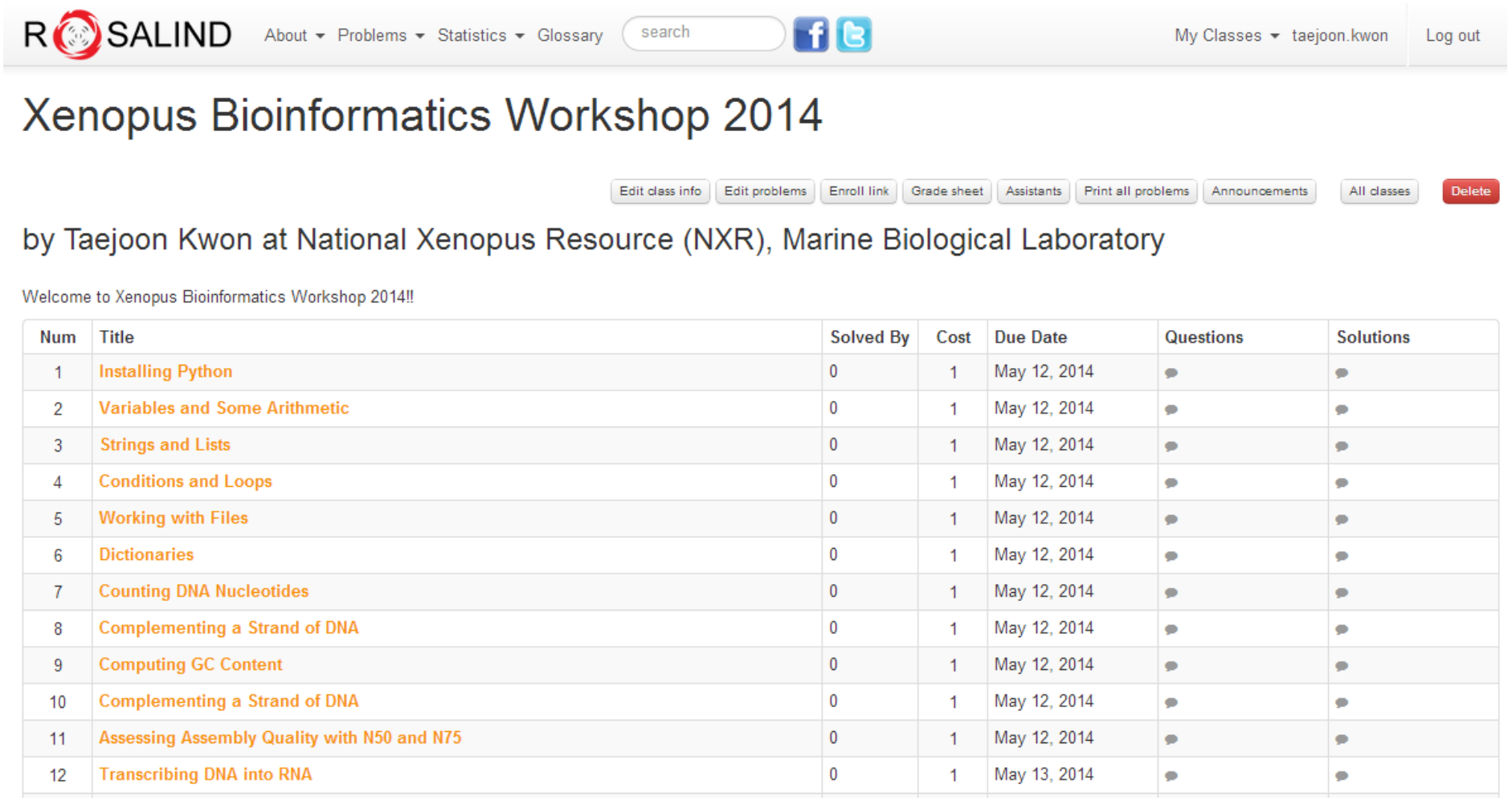
- The way to perform ‘pattern matching’ with strings.
- If built-in function of string is not enough for your job…
  - split(), replace(), strip(), startswith(), endswith(), …
- Google’s python course is good place to start (see below URL).
- Don’t cry if you don’t understand what they are talking about; it is not easy to get it at first sight.

```
## Search for pattern 'iii' in string 'piiig'.
## All of the pattern must match, but it may appear anywhere.
## On success, match.group() is matched text.
match = re.search(r'iii', 'piiig') =>  found, match.group() == "iii"
match = re.search(r'igs', 'piiig') =>  not found, match == None

## . = any char but \n
match = re.search(r'..g', 'piiig') =>  found, match.group() == "iig"

## \d = digit char, \w = word char
match = re.search(r'\d\d\d', 'p123g') =>  found, match.group() == "123"
match = re.search(r'\w\w\w', '@@abcd!!') =>  found, match.group() == "abc"
```

https://developers.google.com/edu/python/regular-expressions

http://www.oyemagazine.org/sites/default/files/imagecache/blog_slideshow/article/Time-to-practice-what-you-learned.gif

Rosalind – “there is a prize!”



Num	Title	Solved By	Cost	Due Date	Questions	Solutions
1	Installing Python	0	1	May 12, 2014		
2	Variables and Some Arithmetic	0	1	May 12, 2014		
3	Strings and Lists	0	1	May 12, 2014		
4	Conditions and Loops	0	1	May 12, 2014		
5	Working with Files	0	1	May 12, 2014		
6	Dictionaries	0	1	May 12, 2014		
7	Counting DNA Nucleotides	0	1	May 12, 2014		
8	Complementing a Strand of DNA	0	1	May 12, 2014		
9	Computing GC Content	0	1	May 12, 2014		
10	Complementing a Strand of DNA	0	1	May 12, 2014		
11	Assessing Assembly Quality with N50 and N75	0	1	May 12, 2014		
12	Transcribing DNA into RNA	0	1	May 13, 2014		

http://rosalind.info/classes/129/

To enroll: http://rosalind.info/classes/enroll/42fa27a979/

Problem #1 – Calculate N50 of genome

- Input
  - A FASTA file of *X. laevis* genome scaffolds (JGI 7.1)
  - A FASTA file of *X. tropicalis* genome (JGI 8.0)
- Output
  - The length of concatenated scaffolds
  - The length of longest scaffolds
  - N50 of scaffolds
  - The number of ‘N’s
- You may need
  - Variables: dictionary, string
  - open(), startswith(), dict(), split(‘’), sort(), len(), int(), print()

Problem #2 – Orthologs of *X. tropicalis*

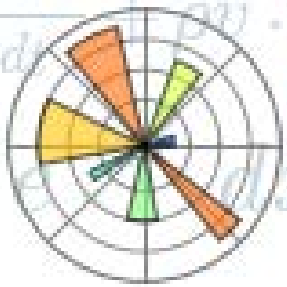
- Input
  - Orthology table of EnsEMBL BioMART (provided)
  - Or you can make it by yourself at http://www.ensembl.org/biomart/martview
- Output
  - Number of orthologous genes per ortholog type (i.e. one-to-one, one-to-many, etc) between human and *X. tropicalis*
- You may need
  - open(), startswith(), split(), dict()

Problem #3 – Extract Dev. Stage expression

- Input
  - Developmental stage expression data (Yanai, 2011;provided)
  - Gene of interest (EnsID or gene name)
    - “How can I apply Problem #3 solution here?”
- Output
  - Expression signals of your interesting gene
  - “How to make it generalized?”

    ./show-me-exp.py Ago2 → show Ago2 expression pattern

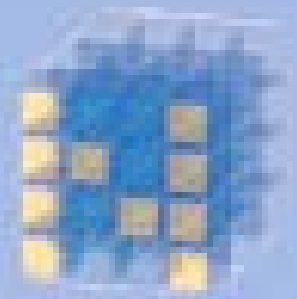
Tomorrow morning – Visualization, etc



matplotlib

SciPy.org

Sponsored By ENTHOUGHT



NumPy