



Welcome to  
**Python 2**  
Session #6

Michael Purcaro and the GSBS Bootstrappers

February 2014

[michael.purcaro@umassmed.edu](mailto:michael.purcaro@umassmed.edu)

# Problem 7: RNA Splicing

- After identifying the exons and introns of an RNA string, we only need to delete the introns and concatenate the exons to form a new string ready for translation.
- **Given:** A DNA string  $s$  and a collection of substrings of  $s$  acting as introns. (All strings are given in FASTA format.)
- **Return:** A protein string resulting from transcribing and translating the exons of  $s$ .
- **Note:** Only one solution will exist for the dataset provided.

# Building Blocks: list comprehension

- Quick way to build lists under certain situations

```
v = []
```

```
for i in range(10):
```

```
    v.append(i)
```

```
v = [x for x in range(10)]
```



# Building Blocks: list comprehension

- Quick way to build certain kinds of lists

```
a = [x for x in range(10)]  
print a
```

```
b = [x*x for x in range(10)]  
print b
```

```
c = [str(x) for x in range(10)]  
print c
```

```
q = 5  
d = [x+q for x in range(10)]  
print d
```

# Building Blocks: map

- Another way to build a list
- Applies a function to every element in a list

```
a = map(lambda x: x, range(10))  
print a
```

```
b = map(lambda x: x*x, range(10))  
print b
```

```
c = map(str, range(10))  
print c
```

```
q = 5  
d = map(lambda x: x+q, range(10))  
print d
```

# Building Blocks: map

- Another way to build a list
- Applies a function to every element in a list

```
a = map(lambda x: x, range(10))  
print a
```

“anonymous” function!  
➔ Function with no name

```
b = map(lambda x: x*x, range(10))  
print b
```

```
c = map(str, range(10))  
print c
```

For examples on this slide, map  
assumes the function takes only  
1 argument

```
q = 5  
d = map(lambda x: x+q, range(10))  
print d
```

# Building Blocks: map

```
chrs = map(lambda x: int(x[3:]),  
            ["chr1", "chr2", "chr3"])  
  
print chrs
```

# Regular Expressions

- How would we extract 4-digit number from this sequence?

`abcd1234dsfasdfa5643asdfasef`

Try extracting the numbers; make sure the code also works for:

`sadfaweqwe1235lkaf9843asdfaefaeef`



# Regular Expressions

- An alternative way to extract the 4-digit number is to use regular expressions
  - We specify a **pattern** that we wish to match

# Regular Expressions

- To match a letter (a-z and A-Z)

`\D`

- To match any two letters

`\D\D`

- To match a digit (0-9)

`\d`

- To match any 1 letter and any 1 digit (0-9)

`\D\d`

# Regular Expressions

- To match any character

.

- To match 0 or more times, add

\*

- To match 1 or more times, add

+

- Example: to match any number of characters, then a digit

.\*\d

# Regular Expressions

To find out if a string matches a pattern, you can do:

```
import re  
if re.search(r"<your pattern", variableName):  
    # Successful match  
else:  
    # Match attempt failed
```

For example, to see if the string has two letters, two digits, and then two letters:

```
if re.search(r"\D\D\d\d\D\D", "aa22aa"):  
    print "matched 2 letters, 2 digits, 2 letters"  
else:  
    print "could not match the pattern"
```

# Regular Expressions

- To denote what to copy out, surround with parenthesis

`\D\D(\d\d)\D\D`

**Extract the number:**

```
match = re.search(r"\D\D(\d\d)\D\D", "aa22aa")
if match:
    print "found number!", int(match.group(1))
else:
    print "no number found!"
```

# Regular Expressions

```
import re
```

```
s = "abcd1234dsfasdfa5643asdfasef"
```

Write the code to extract two numbers from the above string

# Regular Expressions

```
import re
```

```
s = "the1234next5643word"
```

Write the code to extract the words from the above string

# Regular Expressions

- Many other patterns possible!
  - See <https://docs.python.org/2/library/re.html>



# Other Exercises and Info

# Static Class Methods

- Methods that live in a class, but don't need any access to data (via `self`) in that class
- One use: organize miscellaneous functions together

**class** **Utils**:

**@staticmethod**

**def** **mkdir\_p**(path):

...

**@staticmethod**

**def** **get\_file\_if\_size\_diff**(url, path):

Called without using an object!

`Utils.mkdir_p(path)`

`Utils.get_file_if_size_diff(url, path)`

# Extended Exercise 7

- Change the ChipseqData class to use Peak class:

```
class Peak:
    def __init__(self, line):
        toks = line.split()
        self.chr = toks[0]
        self.start = int(toks[1])
        self.end = int(toks[2])

    def length(self):
        return self.end - self.start
```

```

class ChipseqData:
    def __init__(self, paths, url):
        self.paths = paths
        self.url = url
        self.fnp =
Utils.get_file_if_size_diff(self.url,
paths.lectureFolder)

    def getPeaks(self, chr):
        peaks = []
        with open(self.fnp) as f:
            for line in f:
                peak = Peak(line)
                if chr != peak.chr:
                    continue
                peaks.append(peak)
        return peaks

```

```
def numPeaks(self, chr):  
    return len(self.getPeaks(chr))  
  
def computePercentageChromosomeCovered(self, chr,  
                                         chromosomeLength):  
    peaks = self.getPeaks(chr)  
    numBases = 0  
    for peak in peaks:  
        numBases += peak.length()  
    return "{0:.2f}%".format(float(numBases) /  
                              chromosomeLength * 100)
```

# Building Blocks: list comprehension

```
fnp = paths.makeFilePath("ENCFF002COQ.narrowPeak")

with open(fnp) as f:
    allPeaks = [Peak(x) for x in f]

print len(allPeaks)
```

# Building Blocks: list comprehension

```
fnp = paths.makeFilePath("ENCFF002COQ.narrowPeak")

with open(fnp) as f:
    allPeaks = [Peak(x) for x in f]

print len(allPeaks)

55551
```

# Building Blocks: map

```
fnp = paths.makeFilePath("ENCFF002COQ.narrowPeak")

with open(fnp) as f:
    allPeaks = map(Peak, f)

print len(allPeaks)
```



# Building Blocks: map

```
fnp = paths.makeFilePath("ENCFF002COQ.narrowPeak")
```

```
with open(fnp) as f:
```

```
    allPeaks = map(Peak, f)
```

```
print len(allPeaks)
```

```
55551
```