

Dictionaries, Functions

Lists and Array Reminders

To create a list of items, use the []

```
genes = ['SOD1', 'CDC11', 'YFG1']
print(genes)
print(genes[1])
print(genes[1:]) # everything after slot 1 (incl 1)
print(genes[:1]) # everything before slot 1
print(len(genes))
```

```
['SOD1', 'CDC11', 'YFG1']
CDC11
['CDC11', 'YFG1']
['SOD1']
3
```

Some built-in list functions

- [range\(\)](#) - *range(start, stop[, step])*

```
>>> range(5,10,1)
[5, 6, 7, 8, 9]
>>> range(5,-1,-1)
[5, 4, 3, 2, 1, 0]
```

- [map\(\)](#) - lets you update a list with a function

```
l = [ 'a', 100, 12/3.3 ]
# ",".join(l) # this throws an error
";".join(map(str,l)) # have to cast numbers as string
print( ";".join(map(str,l)) )
l = [1,2,3,4]
squares = map(lambda x: x**2,l)
print(squares)
```

```
['a', 100, 3.6363636363636367]
[1, 2, 3, 4]
[1, 4, 9, 16]
```

Reverse a list

- [reversed\(\)](#) - iterate in reverse order of an array/string

```
l = ['zzz', 'yyy', 'a']  
print(list(reversed(l)))  
for n in reversed(l):  
    print(n)
```

```
['a', 'yyy', 'zzz']  
a  
yyy  
zzz
```

More array functions

See more details here <https://docs.python.org/3/tutorial/datastructures.html>

- `list.append(x)` - Add an item to the end of the list;
- `list.pop([i])` - Remove the item at the given position in the list;
- `list.extend(L)` - Extend the list by appending all the items in the given list;
- `list.insert(i, x)` - Insert an item at a given position. The first argument is the index of the element before which to insert, so `a.insert(0, x)` inserts at the front of the list, and `a.insert(len(a), x)` is equivalent to `a.append(x)`.
- `list.remove(x)` - Remove the first item from the list whose value is `x`. It is an error if there is no such item. `list`, and return it. If no index is specified, `a.pop()` removes and returns the last item in the list.
- `list.index(x)` - Return the index in the list of the first item whose value is `x`. It is an error if there is no such item.
- `list.count(x)` - Return the number of times `x` appears in the list.
- `list.sort(cmp=None, key=None, reverse=False)` - Sort the items of the list in place
- `list.reverse()` - Reverse the order of the items in the list

Sorting Lists

The `LIST.sort()` function on a list or the `sorted(LIST)`

<https://docs.python.org/3/howto/sorting.html>

```
#!/usr/bin/env python3
genes = ['SOD1', 'CDC11', 'YFG1']
print(genes)
sort_genes = sorted(genes)
print(sort_genes)
numbers = [141, 7, 90, 3, 13]
print("unsorted", numbers)
numbers.sort()
print("sorted", numbers)
print("reversed", sorted(numbers, reverse=True))

alphanumbers = ['141', '7', '90', '3', '13']
print("Alphanumeric strings", alphanumbers)
print("Alpha sorted numbers", sorted(alphanumbers))
print("Numeric sorted", sorted(alphanumbers, key=int))
```

Dates and times

See <https://docs.python.org/3/library/datetime.html>

```
from datetime import datetime

dates = ['3-Jan-2016', '4-Mar-2015', '2-Aug-1999', '1-May-2000']
print(dates)
dates.sort()
print(dates)

#newdates = [ datetime.strptime(d,"%d-%b-%Y") for d in dates ]
newdates = []
for str in dates:
    newdates.append(datetime.strptime(str, '%d-%b-%Y'))
print(newdates)
newdates.sort()
print(newdates)

for n in newdates:
    print(datetime.strftime(n, "%Y-%b-%d"), " OR ",
          datetime.strftime(n, "%Y-%m-%d"), " OR ",
          datetime.strftime(n, "%A, %b %d, %Y"), " OR ",
          datetime.strftime(n, "%c")
          )
```

Iterate on Strings/Arrays in the same way

```
lst = [ 'BRCA1', 'SOD1', 'PTEN' ]
for gene in sorted(lst):
    print("gene is",gene)

DNA='AAAACCGTAG'
for let in DNA:
    print(let)

for let in reversed(DNA):
    print(let)
```

BRCA1
PTEN
SOD1

A
A
A
...

G
A
T
...

Dictionaries

Initialize a dictionary, Dictionaries are key and value pairs

```
things = {}          # an empty dictionary
listofstuff = []    # an empty array
print(things)
things = {'diane': 10, 'jack': 13}
print(things)
things['diane']
things['billy'] = 15 # assign a new key/value pair
# if you have a list of pairs of things
strangerthings = dict([('Will', 12), ('Jim', 44), ('Joyce', 45), ('Eleven',
strangerthings['Eleven']
```

```
{}
```

```
{'diane': 10, 'jack': 13}
```

```
10
```

```
11
```

Iterate through a dictionary

Using the for loop and the items() function

```
for key,value in strangerthings.items():  
    print("key is", key,"value is",value)
```

```
key is Will value is 12  
key is Jim value is 44  
key is Joyce value is 45  
key is Eleven value is 11  
key is Lucas value is 10
```

Functions

These are blocks of code that can be called repeatedly. Simplify tool development.

Might have subroutine to read a sequence file. Or compute a statistic.

Uses indentation just like loops.

```
def ROUTINENAME(ARGUMENTS):  
    CODE HERE
```

Read Fasta code part 1

<https://drj11.wordpress.com/2010/02/22/python-getting-fasta-with-itertools-groupby/>

See

[https://github.com/biodataprogram/code_templates/blob/master/Lists Dictionaries/fasta_parser.py](https://github.com/biodataprogram/code_templates/blob/master/Lists%20Dictionaries/fasta_parser.py)

```
import itertools
import sys
import re

# based on post here
# https://drj11.wordpress.com/2010/02/22/python-getting-fasta-with-itertools-

# define what a header looks like in FASTA format
def isheader(line):
    return line[0] == '>'
```

Read Fasta code part 2

```
# this function reads in fasta file and returns pairs of data
# where the first item is the ID and the second is the sequence
# it isn't that efficient as it reads it all into memory
# but this is good enough for our project
def aspairs(f):
    seq_id = ''
    sequence = ''
    for header,group in itertools.groupby(f, isheader):
        if header:
            line = next(group)
            seq_id = line[1:].split()[0]
        else:
            sequence = ''.join(line.strip() for line in group)
            yield seq_id, sequence
```

Read Fasta example code Part 3

```
# here is my program
# get the filename from the cmdline
filename = sys.argv[1]
with open(filename,"r") as f:
    seqs = dict(aspairs(f))

# iterate through the sequences
n=0
for k,v in seqs.items():
    print( "id is ",k,"seq is",v)
    n += 1

print(n,"sequences")
```

```
id is Q0142 seq is MTGSGTPPSREVNTYYMTMTMTMTMIMIMTMTMNIHFNNNNNNNINMNSRRMYLFIL*M
id is Q0143 seq is MGLWISFGTPPSYTYLLIMNHKLLLINNNNLTEVHTYFNININIDKMYIH*
```

Dictionaries For Unique Lists

Dictionaries are useful ways to generate a unique list

```
dna = 'AAGAGAGGATACA'  
bases = {'A':0, 'C':0, 'G':0, 'T':0 }  
for l in dna:  
    bases[l] += 1  
  
print(bases)
```

```
{'A': 7, 'C': 1, 'G': 4, 'T': 1}
```

Class problem

Write a script to translate DNA into Protein

- Use dictionary to lookup codons in a FASTA file and convert CDS to amino acid