# Lecture 4

Standard Libraries
BioPython

**Prof. Steven Ludtke**
**N410.07, sludtke@bcm.edu**

# Homework Review

```
import sys

xlate={"ttt":"f" ... "ggg":"g"}

fsp=sys.argv[1]
dna=file(fsp,"r").read()
dna=dna.translate(None,"0123456789 \t\n\r").lower()
out=(fsp+".prot","w")

for i in xrange(0,len(dna),3):
 triplet=dna[i:i+3]
 try: amino=xlate[triplet]
 except:
  print "Unknown triplet: ",triplet
  sys.exit(1)
 out.write(amino)

out.write("\n")
```

# Homework Review

```
import sys

xlate={"ttt":"f" ... "ggg":"g"}

fsp=sys.argv[1]
dna=file(fsp,"r").read()
dna=dna.translate(None,"0123456789 \t\n\r").lower()
out=(fsp+".prot","w")
dna=dna[dna.find("atg"):]

for i in xrange(0,len(dna),3):
 triplet=dna[i:i+3]
 try: amino=xlate[triplet]
 except:
  print "Unknown triplet: ",triplet
  sys.exit(1)
 out.write(amino)

out.write("\n")
```

# Represent as Dict

```
xlate={   "ttt":"f","ttc":"f","tta":"l","ttg":"l",
"ctt":"l","ctc":"l","cta":"l","ctg":"l","att":"i",
"atc":"i","ata":"i","atg":"m","gtt":"v","gtc":"v",
"gta":"v","gtg":"v","tct":"s","tcc":"s","tca":"s",
"tcg":"s","cct":"p","ccc":"p","cca":"p","ccg":"p",
"act":"t","acc":"t","aca":"t","acg":"t","gct":"a",
"gcc":"a","gca":"a","gcg":"a","tat":"y","tac":"y",
"taa":"0","tag":"0","cat":"h","cac":"h","caa":"q",
"cag":"q","aat":"n","aac":"n","aaa":"k","aag":"k",
"gat":"d","gac":"d","gaa":"e","gag":"e","tgt":"c",
"tgc":"c","tga":"0","tgg":"w","cgt":"r","cgc":"r",
"cga":"r","cgg":"r","agt":"s","agc":"s","aga":"r",
"agg":"r","ggt":"g","ggc":"g","gga":"g","ggg":"g"}
```

# Homework Review

```
import sys

xlate={"ttt":"f" ... "ggg":"g"}

fsp=sys.argv[1]
dna=file(fsp,"r").read()
dna=dna.translate(None,"0123456789 \t\n\r").lower()
out=(fsp+".prot","w")
dna=dna[dna.find("atg"):]

for i in xrange(0,len(dna),3):
 triplet=dna[i:i+3]
 try: amino=xlate[triplet]
 except:
  print "Unknown triplet: ",triplet
  sys.exit(1)
 if amino=="0" : break
 out.write(amino)

out.write("\n")
```

# Amortization

```
import sys
balance=float(raw_input("Amount of loan:"))
rate=float(raw_input("Rate as a %:"))/1200.0
payment=float(raw_input("Monthly payment:"))

if rate*balance>payment :
    print "Insufficient payment !"
    sys.exit(1)

month=1
while (balance>0):
    print month,") “,balance,”+",rate*balance,"-",payment,"=", balance+rate*\
balance-payment
    balance+=rate*balance-payment
    month+=1
```

# String Formatting

- {[field][:format]}

- format ::= [[fill]align][sign][#][0][width][,][.precision][type]
- fill ::= <any character>
- align ::= "<" | ">" | "=" | "^"
- sign ::= "+" | "-" | " "
- width ::= integer
- precision ::= integer
- type ::= "b" | "c" | "d" | "e" | "E" | "f" | "F" | "g" | "G" | "n" | "o" | "s" | "x" | "X" | "%"

# Functions

A function is used when some action needs to be completed in different parts of a program, or re-used in multiple programs. It allows code to be grouped in a self-contained block, and can also make debugging easier.

Generally it is not good practice to make functions that are called only one time strictly for organizational purposes. Use comments instead.

# Examples

```
def middle(x): return int(str(x)[1:-1])


def between(lo,val,hi):
   """Checks to see if val is between lo and hi"""
   if lo<val and val<hi : return True
   else: return False


def cmp(a,b):
   """Compare the second element of list a to list b for
   use with sort(), returns -1, 0 or 1"""
   return a[1]-b[1]
```

# A Few Standard Libraries

- sys - System-specific parameters
- os - Operating system functions
- string - String manipulation
- time - Delays, formatting time
- datetime - Manipulate dates/times
- pprint - Pretty printing
- urllib2 - Easy web access

# File Manipulation

- os.getcwd() - the current working directory (folder)
- os.chdir() - change the current working directory
- os.listdir - Lists files in a particular folder
- os.stat - info about a file
- os.rename - rename (mv) a file
- os.mkdir - create a folder
- os.remove - delete a file
- os.rmdir - remove a directory
- os.system - execute a command (mostly mac/linux)

# PyPi

- http://pypi.python.org
- Note that many packages also have installers available for Windows
- easy_install
  - Comes with Mac
  - May be in a package called python_setuptools on linux
- pip
  - newer alternative, but not standard on mac

# BioPython

- easy_install biopython
- Sequence format conversion
- Sequence maniplulation
- Interface to common programs/databases
  - BLAST, Clustalw, EMBOSS, SCOP, SwissProt, ...
- PubMed & Medline access
- Simple GUI programs
- BioSQL integration

# Simple SwissProt Example

```
from Bio import ExPASy
from Bio import SeqIO
handle = ExPASy.get_sprot_raw("A0LR17")
seq_record = SeqIO.read(handle, "swiss")
handle.close()
```

Note: help() comes in handy here…

# BLAST

```
from Bio.Blast import NCBIWWW
from Bio.Blast import NCBIXML

result=NCBIWWW.qblast("blastp","swissprot",
"MAKMIAMADEAARRALERGMNQLADAVKVTLGPKGRNVVLEK
KWGAPTITNDGVSIAKEIELEDPYEKIGAELVKEVAKK")
blast_record = NCBIXML.read(result)
result.close()
blast_record.alignments
```

# Pubmed

```
from Bio import Entrez
from Bio import Medline

# Always tell NCBI who you are
Entrez.email = "A.N.Other@example.com"
handle = Entrez.esearch(db="pubmed", term="Ludtke SJ[Author]",
retmax=500)
record = Entrez.read(handle)
print record
ids=record["IdList"]

handle=Entrez.efetch(db="pubmed",id="22696402",rettype="medline")
records = list(Medline.parse(handle))
```

# Medline Terms

| | | |
|---|---|---|
| Affiliation [AD] | Investigator [IR] | Pharmacological Action [PA] |
| Article Identifier [AID] | ISBN [ISBN] | Place of Publication [PL] |
| All Fields [ALL] | Issue [IP] | PMID [PMID] |
| Author [AU] | Journal [TA] | Publisher [PUBN] |
| Author Identifier [AUID] | Language [LA] | Publication Date [DP] |
| Book [book] | Last Author [LASTAU] | Publication Type [PT] |
| Comment Corrections | Location ID [LID] | Secondary Source ID [SI] |
| Corporate Author [CN] | MeSH Date [MHDA] | Subset [SB] |
| Create Date [CRDT] | MeSH Major Topic [MAJR] | Supplementary Concept[NM] |
| Completion Date [DCOM] | MeSH Subheadings [SH] | Text Words [TW] |
| EC/RN Number [RN] | MeSH Terms [MH] | Title [TI] |
| Editor [ED] | Modification Date [LR] | Title/Abstract [TIAB] |
| Entrez Date [EDAT] | NLM Unique ID [JID] | Transliterated Title [TT] |
| Filter [FILTER] | Other Term [OT] | UID [PMID] |
| First Author Name [1AU] | Owner | Version |
| Full Author Name [FAU] | Pagination [PG] | Volume [VI] |
| Full Investigator Name [FIR] | Personal Name as Subject [PS] | |
| Grant Number [GR] | | |

# Homework 4

- Start thinking about a topic for your class projects. Next homework you will need to tell me what your planned project is.

- Rewrite the sequence translation assignment from the last homework to use biopython to represent the sequences and perform the translation for you, instead of using your own dictionary.

- Write a program using biopython to perform a pubmed query to retrieve reference citations for all of the papers published by your mentor (or anyone else with at least 5 publications). Use the citation list to identify all of the coauthors he/she has published with, and count the number of joint articles with each coauthor. Print the list of coauthors/counts. Turn in the program and the output of the program for the person you ran it on. You do not need to do error checking for things like the same coauthor with slight name variations.