Lecture 4

Some Standard Libraries
BioPython

Prof. Steven Ludtke
N420, sludtke@bcm.edu

Homework Review

Write a simplified amortization program, that is, a program that keeps track of how much you still owe on a loan. We will simplify the math a bit: Assume that each month, the amount increases by the balance times 1/12 the interest rate and decreases by the amount of the fixed monthly payment. You should ask the user for the amount of the loan, the annual percentage interest rate, and the payment amount. For each month, print the payment number, interest for the month, and the remaining balance on the loan after the payment. Continue to write out new months until the loan is payed off.

Homework Review

- How do we represent the data?
  - balance - Balance at the end of each month
  - rate - monthly fractional interest rate
  - payment - amount of monthly payment
- Break the code into small pieces:
  - ask user for values
  - convert rate to monthly fraction (/1200)
  - if balance*rate > payment then raise error
  - loop until balance <=0
    - compute interest = balance*rate
    - balance = balance + interest - payment
    - print values

Homework Review

- How do we represent the data ?
- Break the task into small pieces
- Code each of the pieces

Amortization

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

Simple Examples

3 columns with automatic formatting of an integer, a floating point number and a string

"{}\t{}\t{}".format(int1,float1,string1)

named keywords, which can be reused

"The tree is {tree} ft tall and the house is {house} ft high, the tree is taller at {tree} ft".format(tree=60,house=30)

formatting numbers

"{num:0.5f}, {num:0.3f} and {num:0.5e} are all the same value, but represented in different ways".format(num=123.45678)

fixed width

"{:8s}{:8s}{:8.2f}".format("abc","testing",98.2)

"{:>8s}{:8s}{:8.2f}".format("abc","testing",98.2)

Amortization

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

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

print("year\tmonth\tbalance\tinterest\tpayment")
month=0
while (balance>0):
   print("{}\t{}\t{:1.2f}\t{:1.2f}\t{:1.2f}". format(month//
12+1,month%12+1,balance,balance*rate,payment))
   balance+=rate*balance-payment
   month+=1
```

Amortization

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

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

print("year month     balance interest payment")
month=0
while (balance>0):
   print("{:4d} {:5d}   {:9.2f} {:8.2f} {:1.2f}".
format(month//12+1,month%12+1,balance,balance*rate,
min(balance,payment)))
   balance+=rate*balance-payment
   month+=1
```

Homework Review

Start with the simple DNA -> Protein translation program we wrote in class today (you can download it from the class site). Let's assume that we've dealt with identifying a promotor, etc, and that the sequence we're getting is within a few residues of being the start of a coding region of DNA. However, the exact frame hasn't been identified, and clearly if we start with a frame shift we'll get the wrong sequence. Modify the program to identify the correct frame by assuming the first ATG we find represents the beginning of the coding region, then translate only until a stop codon is found. example: if your program were given 'gatggcagct aaagacgtaa aatgaaaa' it should produce ‘maakdvk'

Homework Review

```
import urllib
seq=urllib.request.urlopen("https://www.ebi.ac.uk/ena/data/view/
M11717&display=text&download=txt&filename=M11717.txt").read()
dna=str(seq,"utf-8").split("\n")

for i,ln in enumerate(dna):
    if ln[:2]=="SQ" : break
dna=" ".join(dna[i+1:])

dna=dna.translate(str.maketrans("CAGT","cagt","/0123456789 \t\n\r"))

xlate={"ttt":"f","ttc":"f", …}

out=[]
for i in range(0,len(dna),3):
  triplet=dna[i:i+3]
  try: amino=xlate[triplet]
  except:
    print("Unknown triplet: ",triplet)
    break
  out.append(amino)

out="".join(out)
print(out)
```

Homework Review

```
import urllib
seq=urllib.request.urlopen("https://www.ebi.ac.uk/ena/data/view/
M11717&display=text&download=txt&filename=M11717.txt").read()
dna=str(seq,"utf-8").split("\n")

for i,ln in enumerate(dna):
    if ln[:2]=="SQ" : break
dna=" ".join(dna[i+1:])

dna=dna.translate(str.maketrans("CAGT","cagt","/0123456789 \t\n\r"))

xlate={"ttt":"f","ttc":"f", …}

start=dna.find("atg")
out=[]
for i in range(start,len(dna),3):
  triplet=dna[i:i+3]
  try: amino=xlate[triplet]
  except:
    print("Unknown triplet: ",triplet)
    break
  out.append(amino)

out="".join(out)
print(out)
```

Homework Review

```
import urllib
seq=urllib.request.urlopen("https://www.ebi.ac.uk/ena/data/view/
M11717&display=text&download=txt&filename=M11717.txt").read()
dna=str(seq,"utf-8").split("\n")

for i,ln in enumerate(dna):
    if ln[:2]=="SQ" : break
dna=" ".join(dna[i+1:])

dna=dna.translate(str.maketrans("CAGT","cagt","/0123456789 \t\n\r"))
dna=dna[dna.find("atg"):]

xlate={"ttt":"f","ttc":"f", …}

out=[]
for i in range(0,len(dna),3):
  triplet=dna[i:i+3]
  try: amino=xlate[triplet]
  except:
    print("Unknown triplet: ",triplet)
    break
  out.append(amino)

out="".join(out)
print(out)
```

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 urllib
seq=urllib.request.urlopen("https://www.ebi.ac.uk/ena/data/view/
M11717&display=text&download=txt&filename=M11717.txt").read()
data=str(seq,"utf-8").split("\n")

for i,ln in enumerate(data):
    if ln[:2]=="SQ" : break
dna=" ".join(data[i+1:])

dna=dna.translate(str.maketrans("CAGT","cagt","/0123456789 \t\n\r"))
dna=dna[dna.find("atg"):]

xlate={"ttt":"f","ttc":"f", …}

out=[]
for i in range(0,len(dna),3):
  triplet=dna[i:i+3]
  try: amino=xlate[triplet]
  except:
    print("Unknown triplet: ",triplet)
    break
  if amino=="0" : break
  out.append(amino)

out="".join(out)
print(out)
```

import

- import module
- from module import name,name2,name3
- from module import *
- import module as othername

A Few Standard Libraries

- sys - System-specific parameters
- os - Operating system functions
- time - Delays, formatting time
- datetime - Manipulate dates/times
- random - Random numbers
- pprint - Pretty printing

PyPi

- http://pypi.python.org
- Note that many packages also have installers available for Windows
- pip
  - included as part of Python 2.7.9 and later
  - should be set up in Anaconda
- If using Anaconda, may consider "conda" instead of pip (only a subset will be available). Any packages available with conda will be easier to install. "conda search", "conda 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

- http://biopython.org/DIST/docs/tutorial/Tutorial.html

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 = "sludtke@bcm.edu"
handle = Entrez.esearch(db="pubmed", term="Ludtke SJ[Author]",
retmax=500)
record = Entrez.read(handle)
handle.close()
print(record)
ids=record["IdList"]

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

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]		

Scope

What will this produce ?

```
def f(x):
    y=x*10
    return y

x=5
y=6

print(y)
print(f(x))
print(y)
```

Scope

How about this ?

```
def f():
    y=x*10
    return y

x=5
y=6

print(y)
print(f())
print(y)
```

Scope

- Local scope
  - Variables defined within a function, exist only within the function
- Global scope
  - Variables defined at the “top level” in the program or module.
  - Variables declared using the “global” keyword
- Built-in names
  - Built in functions and variable names

Scope

What will this produce ?

```
def f(x):
    global y
    y=x*10
    return y

x=5
y=6

print(y)
print(f(x))
print(y)
```

Homework 4

- Install BioPython (http://biopython.org) on your computer, and make sure you can import it successfully before lab next Friday. If you have Anaconda set up as you should, you should just be able to type:

  conda install biopython

  If it is installed properly, the following python code should not raise an error:

  from Bio import SeqIO

- Start thinking about what you might want to do for a class project, I will be asking for your plan soon.

Homework 4

1) Use BioPython to write a program that prompts the user for a protein sequence, runs a BLAST on the sequence, then prints the "RecName" for the top 5 hits (if there are that many). For example, the BLAST we did in class should produce:

```
Full=60 kDa chaperonin 1
Full=60 kDa chaperonin 1
Full=60 kDa chaperonin 1
Full=60 kDa chaperonin 2
Full=60 kDa chaperonin 2
```

2) Use BioPython to write a program that prompts the user for a PubMed query (like the "Ludtke SJ[Author]" from lecture). Print the number of matches, and the Title, First Author Name, Last Author Name

for the first 5 returned results