Essential Computing for Bioinformatics

Lecture 5

High-level Programming with Python

Part II: Container Objects

Bienvenido Vélez
UPR Mayaguez

Reference: How to Think Like a Computer Scientist: Learning with Python (Ch 3-6)

Outline

- Lists
- Matrices
- Tuples
- Dictionaries

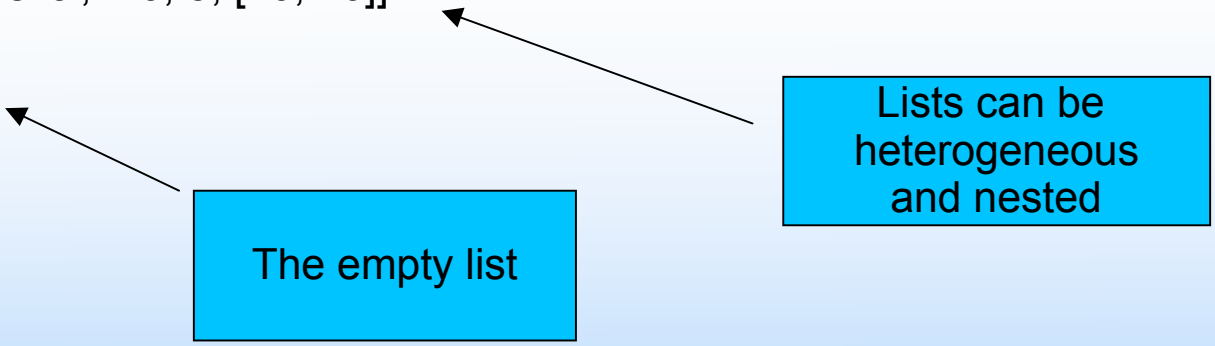
List Values

[10, 20, 30, 40]

['spam', 'bungee', 'swallow']

['hello', 2.0, 5, [10, 20]]

[]



Generating Integer Sequences

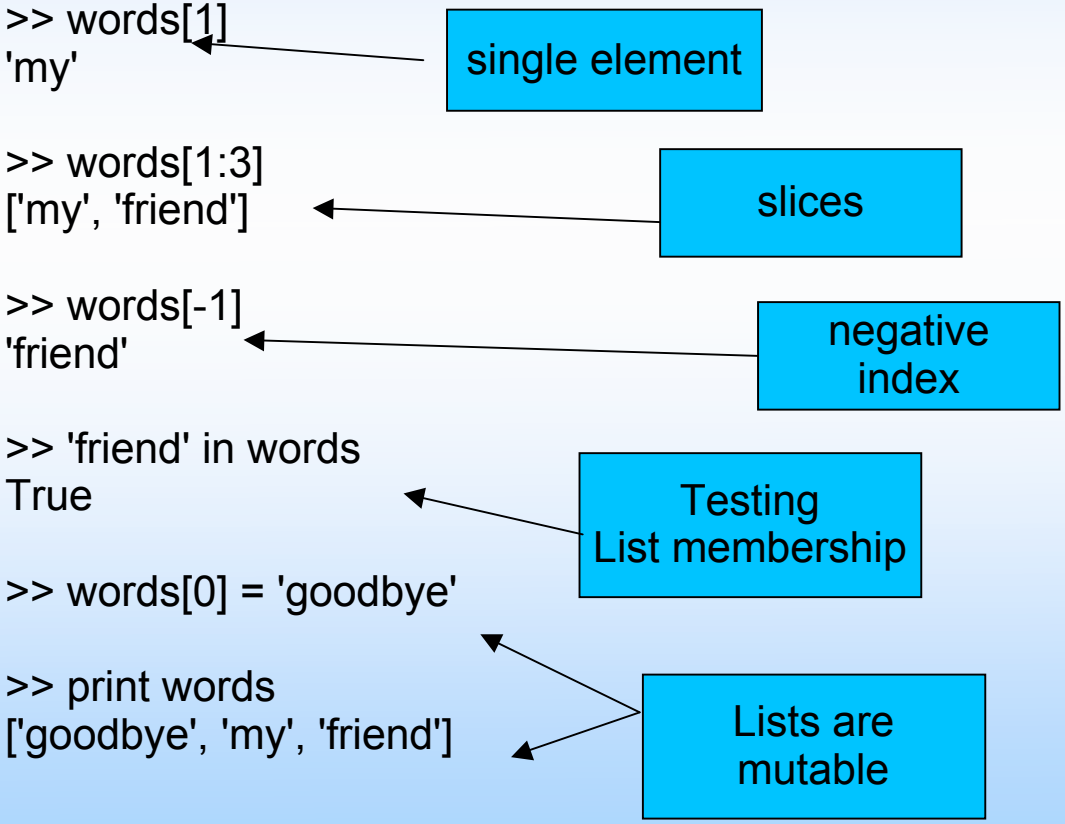
```
>>> range(1,5)
[1, 2, 3, 4]
>>> range(10)
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> range(1, 10, 2)
[1, 3, 5, 7, 9]
```

In General

range(first,last+1,step)

Accessing List Elements

```
>> words=['hello', 'my', 'friend']

>> words[1]
'my'

>> words[1:3]
['my', 'friend']

>> words[-1]
'friend'

>> 'friend' in words
True

>> words[0] = 'goodbye'

>> print words
['goodbye', 'my', 'friend']
```



More List Slices

```
>> numbers = range(1,5)

>> numbers[1:]
[1, 2, 3, 4]

>> numbers[:3]
[1, 2]

>> numbers[:]
[1, 2, 3, 4]
```

Slicing operator always returns a new list

Modifying Slices of Lists

```
>>> list = ['a', 'b', 'c', 'd', 'e', 'f']

>>> list[1:3] = ['x', 'y']

>>> print list
['a', 'x', 'y', 'd', 'e', 'f']
```

Replacing slices

```
>>> list[1:3] = []

>>> print list
['a', 'd', 'e', 'f']
```

Deleting slices

```
>>> list = ['a', 'd', 'f']

>>> list[1:1] = ['b', 'c']

>>> print list
['a', 'b', 'c', 'd', 'f']

>>> list[4:4] = ['e']

>>> print list
['a', 'b', 'c', 'd', 'e', 'f']
```

Inserting slices

Traversing Lists (2 WAYS)

```
for <VARIABLE> in <LIST>:
        <BODY>
```

```
i = 0
while i < len(<LIST>):
        <VARIABLE> = <LIST>[i]
        <BODY>
        i = i + 1
```

Which one do you prefer? Why?

Traversal Examples

```
for number in range(20):
    if number % 2 == 0:
    print number
```

```
for fruit in ['banana', 'apple', 'quince']:
    print 'I like to eat ' + fruit + 's!'
```

Python Sequence Types

Type	Description	Elements	Mutable
StringType	Character string	Characters only	no
UnicodeType	Unicode character string	Unicode characters only	no
ListType	List	Arbitrary objects	yes
TupleType	Immutable List	Arbitrary objects	no
XRangeType	return by xrange()	Integers	no
BufferType Buffer	return by buffer()	arbitrary objects of one type	yes/no

Operations on Sequences

Operator/Function	Action	Action on Numbers
[...], (...), '... '	creation	
s + t	concatenation	addition
s * n	repetition n times	multiplication
s[i]	indexation	
s[i:k]	slice	
x in s	membership	
x not in s	absence	
for a in s	traversal	
len(s)	length	
min(s)	return smallest element	
max(s)	return greatest element	

Exercises

Design and implement Python functions to satisfy the following contracts:

- Return the list of codons in a DNA sequence for a given frame
- Return the lists of restriction sites for an enzyme in a DNA sequence
- Return the list of restriction sites for a lists of enzymes in a DNA sequence

Dictionaries

Dictionaries are *mutable unordered collections* which may contain objects of different sorts. The objects can be accessed using a *key*.

A Codon -> AminoAcid Dictionary

```
>>     code =      { 'ttt': 'F', 'tct': 'S', 'tat': 'Y', 'tgt': 'C',
...                  'ttc': 'F', 'tcc': 'S', 'tac': 'Y', 'tgc': 'C',
...                  'tta': 'L', 'tca': 'S', 'taa': '*', 'tga': '*',
...                  'ttg': 'L', 'tcg': 'S', 'tag': '*', 'tgg': 'W',
...                  'ctt': 'L', 'cct': 'P', 'cat': 'H', 'cgt': 'R',
...                  'ctc': 'L', 'ccc': 'P', 'cac': 'H', 'cgc': 'R',
...                  'cta': 'L', 'cca': 'P', 'caa': 'Q', 'cga': 'R',
...                  'ctg': 'L', 'ccg': 'P', 'cag': 'Q', 'cgg': 'R',
...                  'att': 'I', 'act': 'T', 'aat': 'N', 'agt': 'S',
...                  'atc': 'I', 'acc': 'T', 'aac': 'N', 'agc': 'S',
...                  'ata': 'I', 'aca': 'T', 'aaa': 'K', 'aga': 'R',
...                  'atg': 'M', 'acg': 'T', 'aag': 'K', 'agg': 'R',
...                  'gtt': 'V', 'gct': 'A', 'gat': 'D', 'ggt': 'G',
...                  'gtc': 'V', 'gcc': 'A', 'gac': 'D', 'ggc': 'G',
...                  'gta': 'V', 'gca': 'A', 'gaa': 'E', 'gga': 'G',
...                  'gtg': 'V', 'gcg': 'A', 'gag': 'E', 'ggg': 'G'
...
...                }
>>
```

A DNA Sequence

```
>>> cds = "atgagtgaacgtctgagcattaccccgctggggccgtatatcggcgcacaaa
tttcgggtgccgacctgacgcgcccgttaagcgataatcagtttgaacagctttaccatgcggtg
ctgcgccatcaggtggtgtttctacgcgatcaagctattacgccgcagcagcaacgcgcgctggc
ccagcgttttggcgaattgcatattcaccctgtttacccgcatgccgaaggggttgacgagatca
tcgtgctggatacccataacgataatccgccagataacgacaactggcataccgatgtgacattt
attgaaacgccacccgcaggggcgattctggcagctaaagagttaccttcgaccggcggtgatac
gctctggaccagcggtattgcggcctatgaggcgctctctgttcccttccgccagctgctgagtg
ggctgcgtgcggagcatgatttccgtaaatcgttcccggaatacaaataccgcaaaaccgaggag
gaacatcaacgctggcgcgaggcggtcgcgaaaaacccgccgttgctacatccggtggtgcgaac
gcatccggtgagcggtaaacaggcgctgtttgtgaatgaaggctttactacgcgaattgttgatg
tgagcgagaaagagagcgaagccttgttaagtttttgtttgcccatatcaccaaaccggagttt
caggtgcgctggcgctggcaaccaaatgatattgcgatttgggataaccgcgtgacccagcacta
tgccaatgccgattacctgccacagcgacggataatgcatcgggcgacgatccttggggataaac
cgttttatcgggcggggtaa"
>>>
```

CDS Sequence -> Protein Sequence

```
>>> def translate(cds, code):
...     prot = ""
...     for i in range(0,len(cds),3):
...         codon = cds[i:i+3]
...         prot = prot + code[codon]
...     return prot


>>> translate(cds, code)
'MSERLSITPLGPYIGAQ*'
```

Dictionary Methods and Operations

Table 9.3. Dictionary methods and operations

Method or Operation	Action
d[key]	get the value of the entry with key key in d
d[key] = val	set the value of entry with key key to val
del d[key]	delete entry with key key
d.clear()	removes all entries
len(d)	number of items
d.copy()	makes a shallow copya
d.has_key(key)	returns 1 if key exists, 0 otherwise
d.keys()	gives a list of all keys
d.values()	gives a list of all values
d.items()	returns a list of all items as tuples (key,value)
d.update(new)	adds all entries of dictionary new to d
d.get(key [, otherwise])	returns value of the entry with key key if it exists otherwise returns otherwise
d.setdefaults(key [, val])	same as d.get(key), but if key does not exists sets d[key] to val
d.popitem()	removes a random item and returns it as tuple