DATA 520
Lecture 10
Using Lists

Storing Collections of Data

Lists

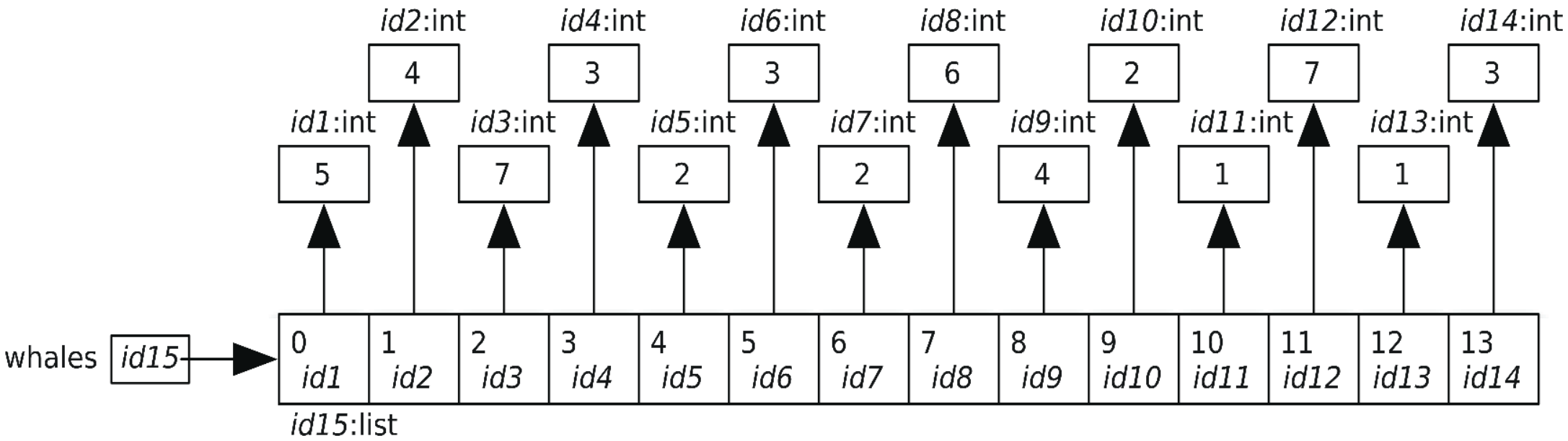
Lists store data sequentially

Lists use brackets

whales contains daily counts of whale sightings

```
whales = [5, 4, 7, 3, 2, 3, 2, 6, 4, 2, 1, 7, 1, 3]
```

each piece of data is stored in memory



Lists

Lists start counting with zero (like most Python things)

```
whales[0] # the first in the list
5
# there are 14 items, so the last is n-1
whales[13] # the last one in this list
3
# notice that a number is returned.
whales[14]
Traceback (most recent call last):
  File "<pyshell#7>", line 1, in <module>
    whales[14] # the last one in this list
IndexError: list index out of range
```

Lists

Lists start counting with zero (like most Python things)

```
>>> whales[0:4] # returns the first ... 5? in the list
[5, 4, 7, 3]
it returns the first FOUR!!!!!!!!!
- so Python uses the list index like this: [>= low, < high]
# notice that a list type is returned
# 14 elements
>>> whales[0:13]
[5, 4, 7, 3, 2, 3, 2, 6, 4, 2, 1, 7, 1]
>>> whales[0:14]
[5, 4, 7, 3, 2, 3, 2, 6, 4, 2, 1, 7, 1, 3]
>>> whales[0:100]
[5, 4, 7, 3, 2, 3, 2, 6, 4, 2, 1, 7, 1, 3]
# wow, returns all with index <= 100
```

Lists

Lists start counting with zero (like most Python things)

```
third_day_cnt = whales[2]
# sets it as a number (integer)

# one way to print (easy)
print('There were', third_day_cnt , 'whales sighted on the third day.')
There were 7 whales sighted on the third day.

# a bit more control this way:
print('There were ' + str(third_day_cnt) + ' whales sighted on the third day.')
There were 7 whales sighted on the third day.
```

Lists

Lists start counting with zero (like most Python things)

```
# - and a list can be empty
Lake_Erie_Whales = []
# you can get the variable
Lake_Erie_Whales
[]
# but it has no elements
Lake_Erie_Whales[0]
Traceback (most recent call last):
  File "<pyshell#19>", line 1, in <module>
    Lake_Erie_Whales[0]
IndexError: list index out of range
Lake_Erie_Whales[0:100]
[]
```

Lists

Lists can be heterogeneous (they can contain different data types)

```
# element = name, symbol, melting point, boiling point in Celsius
>>> krypton = ['Krypton', 'Kr', -157.2, -153.4]
>>> krypton[1] # returns string
'Kr'
>>> krypton[2] # returns float
-157.2
>>> krypton[-1] # last item in list
-153.4
```

Great for organizing data, but functions need to be thought out

```
max(krypton)
sum(krypton)
max(krypton[2:4]) # number indices are 2 and 3; OR: max(krypton[-1],krypton[-2])
-153.4
```

Lists

Lists can easily be edited

```
>>> nobles = ['helium', 'none', 'argon', 'krypton', 'xenon', 'radon']

# oops , I wanted 'neon'
>>> nobles[1] = 'neon'
>>> nobles
['helium', 'neon', 'argon', 'krypton', 'xenon', 'radon']
>>> max(nobles)
'xenon'
>>> nobles[4] = 'Xenon'

>>> max(nobles) # watch capitalization!
'radon'
```

File Edit View History Bookmarks Tools Help
apple laptop battery - Best Buy | vw 2011 air conditioner com... | Pandora One - Listen to Pan... | Amazon.com: Konnecting...: ... | Chemical Elements.com - Krypto...
bourbon peppercorn mushroom sauc
www.chemicalelements.com/elements/kr.html

Home

About This Site
Comments
Help
Links

Window Version

Show Table With:
Name
Atomic Number
Atomic Mass
Electron Configuration
Number of Neutrons
Melting Point
Boiling Point
Date of Discovery
Crystal Structure

Element Groups:
Alkali Metals
Alkaline Earth Metals
Transition Metals
Other Metals
Metalloids
Non-Metals
Halogens
Noble Gases
Rare Earth Elements

Periodic Table: Krypton

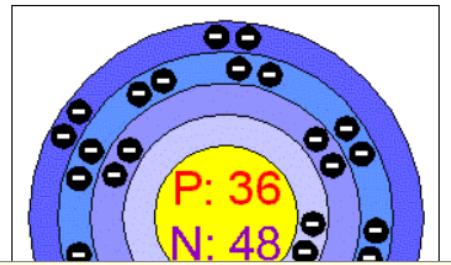
At Chemical Elements.com

Basic Information | Atomic Structure | Isotopes | Related Links | Citing This Page

Basic Information

Name: Krypton
Symbol: Kr
Atomic Number: 36
Atomic Mass: 83.8 amu
Melting Point: -157.2 °C (115.950005 K, -250.95999 °F)
Boiling Point: -153.4 °C (119.75001 K, -244.12 °F)
Number of Protons/Electrons: 36
Number of Neutrons: 48
Classification: Noble Gas
Crystal Structure: Cubic
Density @ 293 K: 3.74 g/cm^3
Color: colorless gas

Atomic Structure



Number of Energy Levels: 4

First Energy Level: 2
Second Energy Level: 8

Scripts Currently Forbidden | <SCRIPT>: 9 | <OBJECT>: 0
Options...

Lists

Gries page 129: **"**Unlike the other types you've learned about, lists can be modified."

Gries page 134: "In contrast to lists, numbers and strings are *immutable*. You cannot, for example, change a letter in a string."

```
# careful, all variables are mutable
name = 'Darwin'
name = str.lower(name)
name = upper(name)
name = name[0].upper() + str.lower(name[1]) + str.upper(name[2]) + name[3].lower() + str.upper(name[4]) + str.lower(name[5])
name = name[5].upper() + str.lower(name[3]) + str.upper(name[0]) + name[1].lower() + str.upper(name[0]) + str.lower(name[2])
BUT:
name[3] = 'Z'
Traceback (most recent call last):
  File "<pyshell#46>", line 1, in <module>
    name[3] = 'Z'
TypeError: 'str' object does not support item assignment
```

Lists

Useful List functions (they are not methods)

Function	Description
len(L)	Returns the number of items in list L
max(L)	Returns the maximum value in list L
min(L)	Returns the minimum value in list L
sum(L)	Returns the sum of the values in list L
sorted(L)	Returns a copy of list L where the items are in order from smallest to largest

Lists

A list of the half-lives of plutonium isotopes Pu-238, Pu-239, Pu-240, Pu-241, and Pu-242:

```
half_lives_Pu = [887.7, 24100.0, 6563.0, 14, 373300.0]
len(half_lives_Pu)
5
max(half_lives_Pu)
373300.0
min(half_lives_Pu)
14
sum(half_lives_Pu)
404864.7
sorted(half_lives_Pu)
[14, 887.7, 6563.0, 24100.0, 373300.0]
half_lives_Pu  # unchanged
[887.7, 24100.0, 6563.0, 14, 373300.0]
```

Lists

Lists can be added to other lists easily

```
original = ['H', 'He', 'Li']
final = original + ['Be']  # notice brackets , making it a list.
final
['H', 'He', 'Li', 'Be']
original = original + ['Be'] # or [2,4,5,6,7]
original
['H', 'He', 'Li', 'Be']
original = original + [5]
original
['H', 'He', 'Li', 'Be', 5]
# will not work without brackets
original = original + 5
original = original + 'He'
```

Lists

Lists can be multiplied easily - but the elements are duplicated

```
original = original * 3
original
```

```
['H', 'He', 'Li', 'Be', 5, 'H', 'He', 'Li', 'Be', 5, 'H', 'He', 'Li', 'Be', 5]
```

Items in lists can be removed easily

```
del original[3:5]  # delete 4th and 5th item !!!
original
```

```
['H', 'He', 'Li', 'H', 'He', 'Li', 'Be', 5, 'H', 'He', 'Li', 'Be', 5]
```

Items in lists can be found easily: in

```
# nobles = ['helium', 'neon', 'argon', 'krypton', 'Xenon', 'radon']
gas = 'Xenon'
gas in nobles #  boolean
```

```
True
```

Lists

Items in lists can be found easily

```
>>> gas = input('Enter a gas: ')

>>> if gas in nobles:
    print('{} is noble.'.format(gas))
```

"in" finds one thing at a time

```
[1, 2] in [0, 1, 2, 3]
False
```

But lists can have lists in them

```
[1, 2] in [0, 1, 2, 3, [1,2]]
True
```

Lists

Lists can be copied, in whole or part (C. elegans)

```
>>> celegans_phenotypes = ['Emb', 'Him', 'Unc', 'Lon', 'Dpy', 'Sma']
>>> useful_markers = celegans_phenotypes[0:4]
>>> useful_markers  # (items 1 through 4)
['Emb', 'Him', 'Unc', 'Lon']
>>> part_celegans_phenotypes = celegans_phenotypes [4:]
>>> part_celegans_phenotypes # items 5 and above
['Dpy', 'Sma']
>>> part_celegans_phenotypes = celegans_phenotypes [:3]
>>> part_celegans_phenotypes # items 2 and below (< upper)
['Emb', 'Him', 'Unc']
```

Duplicate a list

```
>>> dupe_celegans_phenotypes = celegans_phenotypes[:]
>>> dupe_celegans_phenotypes
['Emb', 'Him', 'Unc', 'Lon', 'Dpy', 'Sma']
```

Lists

Lists can be aliased - a bit strange

```
>>> dupe_celegans_phenotypes = celegans_phenotypes[:]
>>> dupe2_celegans_phenotypes = celegans_phenotypes
# change a value
celegans_phenotypes[5] = 'Lvl5'

dupe_celegans_phenotypes
['Emb', 'Him', 'Unc', 'Lon', 'Dpy', 'Sma']

dupe2_celegans_phenotypes
['Emb', 'Him', 'Unc', 'Lon', 'Dpy', 'Lvl5']

# what happened? Does order matter? No.
aliasing works when simple duplication used (=), without colon
(refer to same location in memory)
```

List Methods

Appending an item to a list modifies it directly: append

```
L.append(v)  Appends value v to list L (adds to end)
celegans_phenotypes.append('SHH')
celegans_phenotypes
['Emb', 'Him', 'Unc', 'Lon', 'Dpy', 'Lv15', 'SHH']
```

Count methods

```
L.count(v)   Returns the number of occurrences of v in list L
celegans_phenotypes.count('SHH')
1
celegans_phenotypes.count('n')  # will it find the substring?
0
# no, it looks for complete string matches
```

List Methods

Adding multiple items to a list: extend

```
L.extend(v)  Appends the items in LIST v to L  = L.extend(L2)
celegans_phenotypes.extend(['g123','h333'])
celegans_phenotypes
['Emb', 'Him', 'Unc', 'Lon', 'Dpy', 'Sma', 'SHH', 'g123', 'h333']

celegans_phenotypes.extend(whales)
['Emb', 'Him', 'Unc', 'Lon', 'Dpy', 'Sma', 'SHH', 'g123', 'h333', 5, 4, 7, 3, 2, 3, 2,
6, 4, 2, 1, 7, 1, 3]
```

List Methods

List methods can find items in lists - similar to str.find

L.index(v) Returns the index of the first occurrence of v in L

L.index(v, beg) Returns the index of the first occurrence of v at or after index

begL.index(v, beg, end)

Returns the index of the first occurrence of v between index beg (inclusive) and end (exclusive) in L;

An error is raised if v doesn't occur in that part of L.

```
celegans_phenotypes.index('SHH')
6
celegans_phenotypes.index(2) # find 2
13
celegans_phenotypes.index(2,14) # find 2 after index 14
15
celegans_phenotypes.index(2,16,19) # find 2 at index >= 16 and < 19.
18
```

List Methods

Inserting an item into a list

L.insert(i, v) Inserts value v at index i in list L, shifting later items

```
celegans_phenotypes.insert(2,55) # insert integer 4 at pos 2 (the third one)
celegans_phenotypes
```

```
['Emb', 'Him', 55, 'Unc', 'Lon', 'Dpy', 'Sma', 'SHH', 'g123', 'h333', 5, 4, 7, 3, 2,
3, 2, 6, 4, 2, 1, 7, 1, 3]
```

```
celegans_phenotypes.insert(2,'k12') # will shift all later items
celegans_phenotypes
```

```
['Emb', 'Him', 'k12', 55, 'Unc', 'Lon', 'Dpy', 'Sma', 'SHH', 'g123', 'h333', 5, 4, 7,
3, 2, 3, 2, 6, 4, 2, 1, 7, 1, 3]
```

List Methods

Remove an item from a list

```
L.remove(v) Removes the first occurrence of value v from list L
>>> celegans_phenotypes.remove(2) # will remove integers too
>>> celegans_phenotypes.remove(2)
>>> celegans_phenotypes.remove(2)
>>> celegans_phenotypes.remove(2)
Traceback (most recent call last):
  File "<pyshell#44>", line 1, in <module>
    celegans_phenotypes.remove(2)
ValueError: list.remove(x): x not in list

# to remove without errors
>>> if 2 in celegans_phenotypes:
    celegans_phenotypes.remove(2)
```

List Methods

Sort a list

L.sort() Sorts the values in list L in descending order (for strings with the same letter case, it sorts in reverse alphabetical order)

L.reverse() sorts the other way

```
celegans_phenotypes.sort()
Traceback (most recent call last):
  File "<pyshell#119>", line 1, in <module>
    celegans_phenotypes.sort()
TypeError: '<' not supported between instances of 'int' and 'str'
celegans_phenotypes
['Emb', 'Him', 'k12', 4, 'Unc', 'Lon', 'Dpy', 'Sma', 'SHH', 'g123', 'h333', 5, 4, 7, 3, 3, 6, 4, 1, 7, 1, 3]
celegans_phenotypes.remove(4)  ?? should I do it in this order?
del celegans_phenotypes[11:-1]???
```

List Methods

Removing items from a list

```
# remove the later ones first, index not modified for earlier one
del celegans_phenotypes[11:-1]
celegans_phenotypes.remove(4)
['Emb', 'Him', 'k12', 'Unc', 'Lon', 'Dpy', 'Sma', 'SHH', 'g123', 'h333']

celegans_phenotypes.sort() # changes underlying data (it is a method)
celegans_phenotypes
['Dpy', 'Emb', 'Him', 'Lon', 'SHH', 'Sma', 'Unc', 'g123', 'h333', 'k12']

celegans_phenotypes.reverse() # changes underlying data (it is a method)
celegans_phenotypes
['k12', 'h333', 'g123', 'Unc', 'Sma', 'SHH', 'Lon', 'Him', 'Emb', 'Dpy']
```

Lists

A function to remove the last item in a list

```
def remove_last_item(L):
    """ (list) -> list
    Return list L with the last item removed.

    Precondition: len(L) >= 0

    remove_last_item([1, 3, 2, 4])
    [1, 3, 2]
    """
    del L[-1]
    # return L  # either one will do, return or print
    print(L) # save time typing; Python version 3 uses print ()
```

Lists

A function to remove the last item in a list

```
krypton = ['Krypton', 'Kr', -157.2, -153.4]
>>> remove_last_item(krypton)
['Krypton', 'Kr', -157.2]
>>> remove_last_item(krypton)
['Krypton', 'Kr']
>>> remove_last_item(krypton)
['Krypton']
>>> remove_last_item(krypton)
[]
>>> remove_last_item(krypton)
Traceback (most recent call last):
  File "<pyshell#12>", line 1, in <module>
    remove_last_item(krypton)
  File "C:/Users/sousley/AppData/Local/Programs/Python/Python35-32/001.py", line 11, in remove_last_item
    del L[-1]
IndexError: list assignment index out of range
```

Lists

Lists can contain lists

```
krypton = ['Krypton', 'Kr', -157.2, -153.4, ['A','B','C'] ]
remove_last_item(krypton)
['Krypton', 'Kr', -157.2, -153.4]

# country, life expectancy (an ordered list, similar to a data table)
lifeexp = [['Canada', 76.5], ['United States', 75.5], ['Mexico', 72.0]]
canada = lifeexp[0]
canada
['Canada', 76.5]
canada[1]
76.5
```

Homework 8 due before class Wednesday

Exercises: Gries 8.9, page

1, 3, 4, 7