8 Lists and Dictionaries

Monday, October 28, 2024 8:05 AM

- LISTS

  Sequence of elements.
  - size is not fixed
  - elements can be of different types
  - can be nested.

- Mutable
  - = not a copy but an "alias"
  - passed "by reference" to functions.

- operators

  list(x)

  l[i] both read and write.

  x in l search

  + concatenate

- lists can be sliced.

  l[s : e]

- methods.
  - append (x)
  - extend (l)
  - insert (i, x)
  - remove (x)
  - pop()
  - pop (count)

  - reverse ( )
  - sort ( )
  - index (x)
  - count (x)

- Functions

  max (l)
  min (l)
  sum (l)

  len(l)

  all(l) :- True if all elements are non-zero
  any (l) :- True if any element is non-zero

  sorted(l)

  0 False [ ] are all "zero"

- Iterating over a list

```
for item in list:
    block.
```

- Modify a list in a loop:

  for item in list

  item is a temporary variable.
  changes to item may not reflect later.

- adding or removing elements from the iterating list.

  ```
  for item in list :
  ```
  ↳ strange things can happen if you modify list in the middle of the loop.

- List Comprehensions.

  a "Turbocharged" way of creating lists.

Syntax:

[expr for loop.var in iterable $\underbrace{\text{if cond}}_{\text{optional.}}$]

⇕

```
l = [ ]
for loop.var in iterable :
    if cond:
        l. append ( expr )
```

List comprehensions can be nested !!

e.g. Matrix initialization

```
[ [ 0 for j in range(3) ] for i in range(3) ]
```

Dictionaries

Def: an unordered collection of ⟨key, value⟩ pairs.

- keys are unique
- designed to access elements by key.

Literal

{ key1 : val1 , key2 : val2 , key3 : val3, ...}

NOTE: keys can only be immutable values.

Constructor:

dict(x)

Operators:

dict[key]   del dict[key]   key in dict

Operators:

dict[key]　　del dict[key]　　key in dict

methods:

- clear()
- get(key)
- get(key, def)
- setdefault(def)
- pop(key, def)
- update(dic2)

Iterating over a dictionary:

- default: by keys:
- items()
- values()

- Dictionaries can be nested
  - very common technique for data organization:

Problem: letter frequencies.
Given a string, create a table of how many times each letter appears on the string.

idea 1.-

```
a_counter = 0
b-counter = 0
c-counter = 0
d-counter = 0
⋮
```

idea 2.- list of 26 entries.

idea 3.- use a dictionary to store the counters.

$$\langle \underset{\text{letter}}{\text{key}}, \underset{\text{count}}{\text{value}} \rangle$$

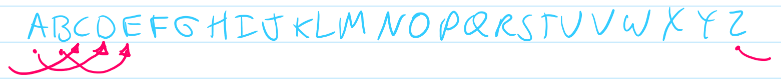
Given string s

#1 1.- look at each character $\overset{c}{\vee}$ in s
2.- increment the counter for c
3.- print table.

#2.-

```
t is an empty dictionary
for c in s:
    if c not in t:
        t[c] = 1;
    else
        t[c] = t[c] + 1
print(t)
```

Problem: Caesar Cypher
Given a message in a string, encrypt the message using the Caesar Cypher



ATTACK
DWWDFN

A	D
B	E
C	F
D	G
E	H
⋮	
X	A
Y	B
Z	C

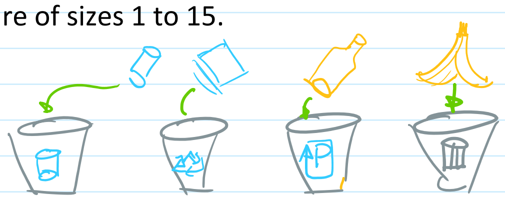
Place table in dictionary!!

Draft:

```
1) initialize table:
      add character pairs to dictionary
2) for each character c in message
      add cypher value of c to cypher-message.
3) return cypher message.
```

Problem: Counting Sort
Given a list of strings, sort the list by string length.

- All strings are of sizes 1 to 15.



$$\begin{tikzcd} & 1 \\ & 2 \\ S \arrow[ur] \arrow[uur] \arrow[r] \arrow[dr] \arrow[ddr] \arrow[dddr] \arrow[ddddr] & 3 \\ & \\ & \\ & \\ & 15 \end{tikzcd}$$

Spit strings into buckets.
^ by string length.

alpha
beta
cat
dog
duck

```
1 : [ ]
2 : [ ]
3 : [cat, dog]
4 : [duck, beta]
5 : [alpha]
6
.
```

1.- create buckets in a dictionary

2.- for each string w in the

duck

5 : [alpha]
6
⋮
15 :

2.- for each string w in the list
    place w in it's bucket

3.- join all the buckets togather
    by lenght size.

EOF