# Ch08-1-Lists

September 10, 2025

## 1 Lists

- http://openbookproject.net/thinkcs/python/english3e/lists.html

### 1.1 Topics

- list data structure
- syntax to create lists
- methods or operations provided to list objects
- list operators
- list traversal
- list applications and problems

### 1.2 List

- a type of sequence or container
- ordered collection of values called elements or items
- lists are similar to strings (ordered collections of characters) except that elements of a list can be of any type

### 1.3 Creating lists

- several ways; the simplest is to enclose the elements in square brackets `[ ]`

[1]:
```
alist = [] # an empty list
```

[2]:
```
blist = list() # use list constructor
```

[3]:
```
type(alist)
```

[3]:
```
list
```

[4]:
```
# creating lists with some elements of same type
list1 = [10, 20, 30, 40]
list2 = ['spam', 'bungee', 'swallow']
```

[5]:
```
# lists with elements of different types
list3 = ["hello", 2.0, 10, [10, ('hi', 'world'), 3.5], (1, 'uno')]
```

[6]:
```
# print list
print(list3)
```

```
['hello', 2.0, 10, [10, ('hi', 'world'), 3.5], (1, 'uno')]
```

[7]:
```
# quickly create a list of range of numbers between 1 and 19
list4 = list(range(1, 20, 1))
```

[8]:
```
print(list4)
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19]
```

[9]:
```
# print multiple lists
print(alist, list1, list2, list3)
```

```
[] [10, 20, 30, 40] ['spam', 'bungee', 'swallow'] ['hello', 2.0, 10, [10, ('hi',
'world'), 3.5], (1, 'uno')]
```

[10]:
```
# Exercise: create a list of even numbers between 1 and 20 inclusive
```

[11]:
```
# Exercise: create a list of odd numbers between 1 and 20 inclusive
```

[12]:
```
# Exercise: create a list of numbers from 20 to 1 inclusive
```

### 1.4 Accessing elements

- same syntax for accessing characters of a string
- use the index operator: `listName[index]`

[13]:
```
# let's see what elements are in list1
list1
```

[13]:
```
[10, 20, 30, 40]
```

[14]:
```
# access an element, which one?
list1[0]
```

[14]:
```
10
```

[15]:
```
list3
```

[15]:
```
['hello', 2.0, 10, [10, ('hi', 'world'), 3.5], (1, 'uno')]
```

[16]:
```
list3[3][0]
```

[16]:
```
10
```

[17]:
```
# list index can be variable as well
index = 0
print(list3[index])
```

```
hello
```

[18]:
```
# can you use float value as an index?
list3[1.0]
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-18-1fc6b2fe2f36> in <module>
      1 # can you use float value as an index?
----> 2 list3[1.0]

TypeError: list indices must be integers or slices, not float
```

### 1.5 Lenght of list

- use `len(listObject)` to find length or number of elements in a list

[19]:
```
# how many elements are there in list3?
len(list3)
```

[19]:
```
5
```

[20]:
```
# what happens if you access an index equal to the size of the list
list3[5]
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
<ipython-input-20-207cd7e1c880> in <module>
      1 # what happens if you access an index equal to the size of the list
----> 2 list3[5]

IndexError: list index out of range
```

[21]:
```
list3
```

[21]:
```
['hello', 2.0, 10, [10, ('hi', 'world'), 3.5], (1, 'uno')]
```

[22]:
```
# Exercise: access and print the last element of list3
```

[23]:
```
# Can we use negative index?
# Can you guess the output of the following code?
print(list3[-1])
```

```
(1, 'uno')
```

[24]:
```
# Exercise - access and print 'world' in list3
```

### 1.6 Checking for membership

- checking if some data/object is a member/element in the given list
- `in` and `not in` boolean operators let's you check for membership

[25]:
```
horsemen = ["war", "famine", "pestilence", ["death"]]
'death' in horsemen
```

[25]:
```
False
```

[26]:
```
'War' not in horsemen
```

[26]:
```
True
```

[27]:
```
["death"] in horsemen
```

[27]:
```
True
```

### 1.7 Traversing lists

- for or while loop can be used to traverse through each element of a list

[28]:
```
list3
```

[28]:
```
['hello', 2.0, 10, [10, ('hi', 'world'), 3.5], (1, 'uno')]
```

[29]:
```
# common technique; use for loop
for item in list3:
    print(item)
```

```
hello
2.0
10
[10, ('hi', 'world'), 3.5]
(1, 'uno')
```

[30]:
```
for item in list3:
    if isinstance(item, list) or isinstance(item, tuple):
        for l in item:
            print(l)
    else:
        print(item)
```

```
hello
2.0
10
10
('hi', 'world')
3.5
```

```
1
uno
```

[31]:
```
horsemen = ["war", "famine", "pestilence", "death"]
for i in [0, 1, 2, 3]:
    print(horsemen[i])
# better way to do the same thing?
```

```
war
famine
pestilence
death
```

[32]:
```
print("traversing using indices")
for i in range(len(horsemen)):
    print(horsemen[i])
```

```
traversing using indices
war
famine
pestilence
death
```

[33]:
```
print('traversing each element')
for ele in horsemen:
    print(ele)
```

```
traversing each element
war
famine
pestilence
death
```

### 1.8 list operators

- + operator concatenates two lists and gives a bigger list
- * operator repeats a list elements a given number of times

[34]:
```
list2
```

[34]:
```
['spam', 'bungee', 'swallow']
```

[35]:
```
list3
```

[35]:
```
['hello', 2.0, 10, [10, ('hi', 'world'), 3.5], (1, 'uno')]
```

[36]:
```
list4 = list2 + list3
```

[37]:
```
list4
```

[37]:
```
['spam',
 'bungee',
 'swallow',
 'hello',
 2.0,
 10,
 [10, ('hi', 'world'), 3.5],
 (1, 'uno')]
```

[38]:
```
[0]*10
```

[38]:
```
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
```

[39]:
```
a = [1, 2, 3]*4
```

[40]:
```
a
```

[40]:
```
[1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3]
```

[41]:
```
b = [a]*3
```

[42]:
```
b
```

[42]:
```
[[1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3],
 [1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3],
 [1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3]]
```

[43]:
```
# 2-D list or matrix
matrix = [[1, 2], [3, 4], [5, 6]]
```

[44]:
```
print(matrix)
```

```
[[1, 2], [3, 4], [5, 6]]
```

[45]:
```
matrix
```

[45]:
```
[[1, 2], [3, 4], [5, 6]]
```

[46]:
```
# How do you replace 5 with 50 in matrix?
matrix[2][0] = 50
```

[47]:
```
matrix
```

[47]:
```
[[1, 2], [3, 4], [50, 6]]
```

### 1.9 Slicing lists

- all the slice operations that work with strings also work with lists
- syntax: [startIndex : endIndex : step]

- startIndex is inclusive; endIndex is exclusive; step is optional by default 1

[48]:
```
# create a list of lower-case alphabets
alphas = ['a', 'b', 'c', 'd', 'e', 'f', 'g'] # add the rest...
```

[49]:
```
alphas
```

[49]:
```
['a', 'b', 'c', 'd', 'e', 'f', 'g']
```

[50]:
```
# there's better way to create lists of all lowercase ascii
import string
alphas = list(string.ascii_lowercase)
```

[51]:
```
print(alphas[:])
```

```
['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p',
'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z']
```

[52]:
```
print(alphas[::3])
```

```
['a', 'd', 'g', 'j', 'm', 'p', 's', 'v', 'y']
```

[53]:
```
print(alphas[1:3])
```

```
['b', 'c']
```

[54]:
```
print(alphas[::-1])
```

```
['z', 'y', 'x', 'w', 'v', 'u', 't', 's', 'r', 'q', 'p', 'o', 'n', 'm', 'l', 'k',
'j', 'i', 'h', 'g', 'f', 'e', 'd', 'c', 'b', 'a']
```

### 1.10 Lists and strings

- match made in heaven - work together really well :)

[55]:
```
# convert string to list of characters
alphaList = list(string.ascii_lowercase)
```

[56]:
```
alphaList
```

[56]:
```
['a',
 'b',
 'c',
 'd',
 'e',
 'f',
 'g',
 'h',
 'i',
 'j',
```

```
 'k',
 'l',
 'm',
 'n',
 'o',
 'p',
 'q',
 'r',
 's',
 't',
 'u',
 'v',
 'w',
 'x',
 'y',
 'z']
```

[57]:
```
# convert list to string by joining pairs of chars with a delimiter
alphaStr = '|'.join(alphaList)
```

[58]:
```
alphaStr
```

[58]:
```
'a|b|c|d|e|f|g|h|i|j|k|l|m|n|o|p|q|r|s|t|u|v|w|x|y|z'
```

### 1.11 lists are mutable

- we can change/replace/update list elements in place

[59]:
```
names = ["john", "David", "Alice"]
names[0] = "jake"
```

[60]:
```
names
```

[60]:
```
['jake', 'David', 'Alice']
```

[61]:
```
# How to correct spelling of jake?
names[0][0]
```

[61]:
```
'j'
```

[62]:
```
names[0][0] = 'J'
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-62-0442e9474c4b> in <module>
----> 1 names[0][0] = 'J'
```

```
TypeError: 'str' object does not support item assignment
```

[63]:
```
names[0] = 'Jake'
```

[64]:
```
names
```

[64]:
```
['Jake', 'David', 'Alice']
```

[65]:
```
alphas
```

[65]:
```
['a',
 'b',
 'c',
 'd',
 'e',
 'f',
 'g',
 'h',
 'i',
 'j',
 'k',
 'l',
 'm',
 'n',
 'o',
 'p',
 'q',
 'r',
 's',
 't',
 'u',
 'v',
 'w',
 'x',
 'y',
 'z']
```

[66]:
```
alphas[:4] = ['A', 'B', 'C', 'D']
```

[67]:
```
alphas
```

[67]:
```
['A',
 'B',
 'C',
 'D',
 'e',
 'f',
```

```
 'g',
 'h',
 'i',
 'j',
 'k',
 'l',
 'm',
 'n',
 'o',
 'p',
 'q',
 'r',
 's',
 't',
 'u',
 'v',
 'w',
 'x',
 'y',
 'z']
```

[68]:
```
alphas[:4] = []
```

[69]:
```
alphas
```

[69]:
```
['e',
 'f',
 'g',
 'h',
 'i',
 'j',
 'k',
 'l',
 'm',
 'n',
 'o',
 'p',
 'q',
 'r',
 's',
 't',
 'u',
 'v',
 'w',
 'x',
 'y',
 'z']
```

### 1.12 Deleting list elements

- del statement removes an element from a list given its index

[70]:
```
alphas
```

[70]:
```
['e',
 'f',
 'g',
 'h',
 'i',
 'j',
 'k',
 'l',
 'm',
 'n',
 'o',
 'p',
 'q',
 'r',
 's',
 't',
 'u',
 'v',
 'w',
 'x',
 'y',
 'z']
```

[71]:
```
del alphas[0]
```

[72]:
```
alphas
```

[72]:
```
['f',
 'g',
 'h',
 'i',
 'j',
 'k',
 'l',
 'm',
 'n',
 'o',
 'p',
 'q',
 'r',
 's',
 't',
```

```
 'u',
 'v',
 'w',
 'x',
 'y',
 'z']
```

[73]:
```
del alphas[26]
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
<ipython-input-73-80e1a19bb44c> in <module>
----> 1 del alphas[26]

IndexError: list assignment index out of range
```

[74]:
```
alphas.index('z')
```

[74]:
```
20
```

[75]:
```
alphas.index(alphas[-1])
```

[75]:
```
20
```

[76]:
```
del alphas[1:3]
```

[77]:
```
alphas
```

[77]:
```
['f',
 'i',
 'j',
 'k',
 'l',
 'm',
 'n',
 'o',
 'p',
 'q',
 'r',
 's',
 't',
 'u',
 'v',
 'w',
 'x',
 'y',
```

```
 'z']
```

[78]:
```
indexOfZ = alphas.index('z')
del alphas[indexOfZ]
```

[79]:
```
print(alphas)
```

```
['f', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w',
'x', 'y']
```

### 1.13 Objects and references

- **is** operator can be used to test if two objects are referencing the same memory location
  - meaning they're essentially the same object with the same values

[80]:
```
# even though a and b are two separate objects is still evaluates to True
a = 'apple'
b = 'apple'
a is b
```

[80]:
```
True
```

[81]:
```
# even though c and d are two separate objects is still evaluates to True
c = 10
d = 10
c is d
```

[81]:
```
True
```

[82]:
```
# what about tuple?
e = (1, 2)
f = (1, 2)
print(e == f)
print(e is f)
```

```
True
False
```

[83]:
```
# What about lists?
l1 = [1, 2, 3]
l2 = [1, 2, 3]
print(l1 == l2)
print(l1 is l2)
```

```
True
False
```

### 1.14 Copying lists (Shallow copy vs Deep copy)

- see PythonTutor.com to visualize aliasing

- assignment = operator does shallow copy

[84]:
```
a = [1, 2, 3]
b = a
print(a is b)
print(a == b)
```

```
True
True
```

[85]:
```
b[0] = 10
print(a)
print(b)
```

```
[10, 2, 3]
[10, 2, 3]
```

[86]:
```
# How do you actually clone lists - do a deep copy?
c = a[:] # easy way shallow copy
d = a.copy() # shallow copy
import copy
e = copy.deepcopy(b)
```

[87]:
```
c is a
```

[87]:
```
False
```

[88]:
```
d is a
```

[88]:
```
False
```

[89]:
```
b is e
```

[89]:
```
False
```

### 1.15 List methods

- list objects have a bunch methods that can be invoked to work with list
- run help(list)

[90]:
```
help(list)
```

```
Help on class list in module builtins:

class list(object)
 |  list(iterable=(), /)
 |
 |  Built-in mutable sequence.
 |
 |  If no argument is given, the constructor creates a new empty list.
```

```
|  The argument must be an iterable if specified.
|
|  Methods defined here:
|
|  __add__(self, value, /)
|      Return self+value.
|
|  __contains__(self, key, /)
|      Return key in self.
|
|  __delitem__(self, key, /)
|      Delete self[key].
|
|  __eq__(self, value, /)
|      Return self==value.
|
|  __ge__(self, value, /)
|      Return self>=value.
|
|  __getattribute__(self, name, /)
|      Return getattr(self, name).
|
|  __getitem__(…)
|      x.__getitem__(y) <==> x[y]
|
|  __gt__(self, value, /)
|      Return self>value.
|
|  __iadd__(self, value, /)
|      Implement self+=value.
|
|  __imul__(self, value, /)
|      Implement self*=value.
|
|  __init__(self, /, *args, **kwargs)
|      Initialize self.  See help(type(self)) for accurate signature.
|
|  __iter__(self, /)
|      Implement iter(self).
|
|  __le__(self, value, /)
|      Return self<=value.
|
|  __len__(self, /)
|      Return len(self).
|
|  __lt__(self, value, /)
|      Return self<value.
```

```
|
|  __mul__(self, value, /)
|      Return self*value.
|
|  __ne__(self, value, /)
|      Return self!=value.
|
|  __repr__(self, /)
|      Return repr(self).
|
|  __reversed__(self, /)
|      Return a reverse iterator over the list.
|
|  __rmul__(self, value, /)
|      Return value*self.
|
|  __setitem__(self, key, value, /)
|      Set self[key] to value.
|
|  __sizeof__(self, /)
|      Return the size of the list in memory, in bytes.
|
|  append(self, object, /)
|      Append object to the end of the list.
|
|  clear(self, /)
|      Remove all items from list.
|
|  copy(self, /)
|      Return a shallow copy of the list.
|
|  count(self, value, /)
|      Return number of occurrences of value.
|
|  extend(self, iterable, /)
|      Extend list by appending elements from the iterable.
|
|  index(self, value, start=0, stop=9223372036854775807, /)
|      Return first index of value.
|
|      Raises ValueError if the value is not present.
|
|  insert(self, index, object, /)
|      Insert object before index.
|
|  pop(self, index=-1, /)
|      Remove and return item at index (default last).
|
```

```
 |      Raises IndexError if list is empty or index is out of range.
 |
 |  remove(self, value, /)
 |      Remove first occurrence of value.
 |
 |      Raises ValueError if the value is not present.
 |
 |  reverse(self, /)
 |      Reverse *IN PLACE*.
 |
 |  sort(self, /, *, key=None, reverse=False)
 |      Sort the list in ascending order and return None.
 |
 |      The sort is in-place (i.e. the list itself is modified) and stable (i.e.
the
 |      order of two equal elements is maintained).
 |
 |      If a key function is given, apply it once to each list item and sort
them,
 |      ascending or descending, according to their function values.
 |
 |      The reverse flag can be set to sort in descending order.
 |
 |  ----------------------------------------------------------------------
 |  Static methods defined here:
 |
 |  __new__(*args, **kwargs) from builtins.type
 |      Create and return a new object.  See help(type) for accurate signature.
 |
 |  ----------------------------------------------------------------------
 |  Data and other attributes defined here:
 |
 |  __hash__ = None
```

[91]:
```
a = []
a.append(1)
a.append(2)
a.append([2, 3])
```

[92]:
```
a
```

[92]:
```
[1, 2, [2, 3]]
```

[93]:
```
a.extend([3, 4])
```

[94]:
```
a
```

[94]:
```
[1, 2, [2, 3], 3, 4]
```

[95]:
```
a.append([5, 6])
```

[96]:
```
a
```

[96]:
```
[1, 2, [2, 3], 3, 4, [5, 6]]
```

[97]:
```
a.insert(0, 'hi')
```

[98]:
```
a
```

[98]:
```
['hi', 1, 2, [2, 3], 3, 4, [5, 6]]
```

[99]:
```
a.reverse()
```

[100]:
```
a[0].reverse()
```

[101]:
```
a
```

[101]:
```
[[6, 5], 4, 3, [2, 3], 2, 1, 'hi']
```

[102]:
```
a.sort()
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-102-2ed0d7de6146> in <module>
----> 1 a.sort()

TypeError: '<' not supported between instances of 'int' and 'list'
```

[103]:
```
# let's create a list of numbers in descending order to sort
blist = list(range(10, 0, -1))
```

[104]:
```
blist
```

[104]:
```
[10, 9, 8, 7, 6, 5, 4, 3, 2, 1]
```

[105]:
```
blist.sort()
```

[106]:
```
print(blist)
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

[107]:
```
# sort blist in reverse/descending order
blist.sort(reverse=True)
```

[108]:
```
blist
```

[108]:
```
[10, 9, 8, 7, 6, 5, 4, 3, 2, 1]
```

[109]:
```
m = max(blist)
mI = blist.index(m)
```

[110]:
```
print(mI)
```

```
0
```

[111]:
```
min(blist)
```

[111]:
```
1
```

[112]:
```
# how many 100s are in blist?
print(blist.count(100))
```

```
0
```

## 1.16 List applications

### 1.16.1 convert a string to list of integers

[1]:
```
nums = input('Enter 5 numbers separated by space: ')
```

```
Enter 5 numbers separated by space: 10 15 1 3 4
```

[2]:
```
nums
```

[2]:
```
'10 15 1 3 4'
```

[3]:
```
nums = nums.split(' ')
```

[4]:
```
nums
```

[4]:
```
['10', '15', '1', '3', '4']
```

[5]:
```
intNums = []
for n in nums:
    intN = int(n)
    intNums.append(intN)
```

[6]:
```
intNums
```

[6]:
```
[10, 15, 1, 3, 4]
```

[7]:
```
intNums.sort(reverse=True)
```

[8]:
```
intNums
```

[8]:
```
[15, 10, 4, 3, 1]
```

### 1.16.2 convert list of integers to string

[9]:
```
' '.join(intNums)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-9-0dd63a44d5ec> in <module>
----> 1 ' '.join(intNums)

TypeError: sequence item 0: expected str instance, int found
```

[10]:
```
strNum = []
for n in intNums:
    strNum.append(str(n))
```

[11]:
```
strNum
```

[11]:
```
['15', '10', '4', '3', '1']
```

[12]:
```
strNum = ' '.join(strNum)
```

[13]:
```
strNum
```

[13]:
```
'15 10 4 3 1'
```

## 1.17 Passing list to function - passed-by-reference

- mutable types such as list are passed-by-reference
- a reference/location is passed instead of a copy of the data

[14]:
```
def getData(someList):# someList is formal parameter
    for i in range(5):
        a = int(input('enter a number: '))
        someList.append(a)
```

[16]:
```
alist = []
getData(alist) # alist is actual parameter/argument
```

```
enter a number: 100
enter a number: 99
enter a number: 75
enter a number: 33
enter a number: 13
```

[17]:
```
# when formal parameter is updated, actual parameter is also updated
alist
```

[17]:
```
[100, 99, 75, 33, 13]
```

### 1.17.1 visualize - pass-by-reference with pythontutor.com

## 1.18 return list from functions

- lists can be returned from functions

[18]:
```
def getMaxMin(alist):
    m = max(alist)
    minVal = min(alist)
    return [m, minVal]
```

[19]:
```
alist = range(-1000, 2000000)
print(getMaxMin(alist))
```

```
[1999999, -1000]
```

[20]:
```
assert getMaxMin(alist) == [1999999, -1000]
```

## 1.19 Casting list into tuple and back

- since tuples are immutable it may be benefitial to cast them into lists and update
- can convert list back to tuple again

[21]:
```
atuple = (1, 2, 3)
alist = list(atuple)
print(alist)
```

```
[1, 2, 3]
```

[22]:
```
btuple = tuple(alist)
```

[23]:
```
print(btuple)
```

```
(1, 2, 3)
```

[24]:
```
atuple == btuple
```

[24]:
```
True
```

[25]:
```
# eventhough the elements are equal; the types of two objects are not
print(atuple, alist)
print(atuple == alist)
```

```
(1, 2, 3) [1, 2, 3]
False
```

## 1.20 Exercises

1. Practice with the rest of the methods of list
2. Draw memory state snapshot for a and b after the following Python code is executed:

```
a = [1, 2, 3]
b = a[:]
b[0] = 5
```

3. Lists can be used to represent mathematical vectors. Write a function `add_vectors(u, v)` that takes two lists of numbers of the same length, and returns a new list containing the sums of the corresponding elements of each. The following test cases must pass once the add_vectors is complete.

[26]:
```
def add_vectors(a, b):
    pass
```

[27]:
```
# test cases
assert add_vectors([1, 1], [1, 1]) == [2, 2], 'vectors not added correctly'
assert add_vectors([1, 2], [1, 4]) == [2, 6], 'vectors not added correctly'
assert add_vectors([1, 2, 1], [1, 4, 3]) == [2, 6, 4], 'vectors not added␣
 ↪correctly'
```

```
---------------------------------------------------------------------------
AssertionError                            Traceback (most recent call last)
<ipython-input-27-f13d12d55331> in <module>
      1 # test cases
----> 2 assert add_vectors([1, 1], [1, 1]) == [2, 2], 'vectors not added␣
 ↪correctly'
      3 assert add_vectors([1, 2], [1, 4]) == [2, 6], 'vectors not added␣
 ↪correctly'
      4 assert add_vectors([1, 2, 1], [1, 4, 3]) == [2, 6, 4], 'vectors not␣
 ↪added correctly'

AssertionError: vectors not added correctly
```

## 1.21 Kattis problems

- the following Kattis problems can be solved using list

1. Dice Game - https://open.kattis.com/problems/dicegame
2. Height Ordering - https://open.kattis.com/problems/height
3. What does the fox say? - https://open.kattis.com/problems/whatdoesthefoxsay
4. Army Strength (Easy) - https://open.kattis.com/problems/armystrengtheasy
5. Army Strength (Hard) - https://open.kattis.com/problems/armystrengthhard
6. Black Friday - https://open.kattis.com/problems/blackfriday

### 1.21.1 List sorting with two keys

1. Roll Call - https://open.kattis.com/problems/rollcall
2. Cooking Water - https://open.kattis.com/problems/cookingwater

[ ]:
```

```