Code Quest: My Tips, Tricks, and Strategy

Samuel Shuster

February 2025

Index & Content

List Operations

list()

Convert your input into a list. Very simple but very useful, may be used to turn a string into a list of chars,
e.g. list(“hello”) -> [‘h’, ‘e’, ‘l’, ‘l’ ‘o’]
Or to convert one of the next functions back into a usable list.

map()

Quick way to run a function on every element of a list. Very helpful for converting strings to integers e.g.
input is say “12 9 2 1”
list(map(int, input()))-> [12, 9, 2, 1]
We can use this later to sort, sum etc...

filter()

How to make sure everything in a list passes a function. Use full in conjunction with a lambda inline functionE.g:
list(filter(lambda x: x%2==0, [1,2,3,4]
-> [2, 4]

lambda x:

How to do a quick inline function for a map, filter or any function that takes in another function. E.g:
list(map(lambda x: “#”+str(x*2), [1,2,3])) -> [“#2”, “#4”, “#6”]

More List Stuff

_ in _

Easy way to tell if something is in another thing (chr or substr in str or element in array). Only works for one level

```
letter = “i”
if letter in “hijklm”:
    print(“true”) # true (this is executed)
```

sorted()

Sorts a list alphabetically or numerically, you can give it a key=lambda x: to sort by the value given, or pass reverse=True to sort descending (greatest first).

```
list(sorted([4, 2, 1, 3]) # [1, 2, 3, 4]
```

reversed()

Reverses a list. May need to turn back into a list object like others. E.g.

```
list(reversed([1, 2, 3, 4])) # 4, 3, 2, 1
```

.append()

Adds an element to the end of the list

```
[1,2,3].append(4) # [1,2,3,4]
```

```
L = ["c", "a", "b"]

print(list(sorted(L)))
# ["a", "b", "c"]

print(list(map(lambda x: "{" + x + "}", L)))
# ['{c}', '{a}', '{b}']

L = list(filter(lambda x: x != "c", L))

print(list(sorted(L)))
# ['a', 'b']

list(map(lambda x: "#"+str(x*2), [1,2,3]))
# ["#2", "#4", "#6"]
```

Ord

Take a character and give its unicode number representation. E.g:
ord(“a”) = 97
ord(“A”) = 65
ord(“B”) = 66
ord(“Y”) = 89
ord(“Z”) = 90 (65+25)

Chr

Inverse of ord; takes a unicode number and gives a character. E.g:
chr(97) = “a”a
chr(65) = “A”
chr(ord(“A”)+1) = “B”
chr(ord(“C”)-2) = “A”

Ord & Chr

Used by many challenges

Many ASCII challenges may be done quickly using ord and chr.

Remember for strings and characters there are many methods like

“abc”.isalpha()
“1”.isdigit()
“1”.isnumeric()
“l”.islower()
“L”.isupper()

Use your IDE to see all available methods

Strings

Now you can write awful code.

```
for _ in range(int(input())):
  print( sum( list(map(lambda x: ord(x) - ord("a") + 1, list(input()) ))) )
```

This solves a CQA challenge

```python
for _ in range(int(input())):
  print(
    sum(
      list(  # convert the map object to a list so that sum can use it
        map(  # do this function on every element
          lambda x: ord(x) - ord("a") + 1,# convert each letter to a its difference to “a” so a is 1, b is 2
          list(input())  # split the input into a list of chars
        )
      )
    )
  )
```

Decimal

Use the built-in decimal package whenever floats are involved

```
import decimal
def round_normal(number, digits=0):
   decimal.getcontext().rounding =
decimal.ROUND_HALF_UP
   return round(decimal.Decimal(str(number)), digits)
```

Quarks
Decimals can be (or round to) -0, which is unacceptable

round(decimal.Decimal('0.5')) is 0, as context is ignored without specifying the number of digits

Convert str's into Decimals instead of floats

A = decimal.Decimal("0.1")
B = decimal.Decimal("0.2")

Then, do math as normal

C = A + B
print(str(C)) # 0.3 exactly, no floating point inaccuracies
print(str(B/A)) # (0.2/0.1=) 2 exactly

Print, round or str(), just like an int

print(A) # 0.1 [automatically calls str(A)]
print("number: " str(A)) # number: 0.1
print(round(A)) # 0
print(round(A, 1)) # 0.1

Set decimal rounding context for proper rounding

decimal.getcontext().rounding = decimal.ROUND_HALF_UP
round(decimal.Decimal('0.5'), 0) # 1

The CodeQuest provided tests are not good.
In my experience, if once I get my code to pass the tests, I get it wrong, it’s unlikely I’ll get the challenge later on. Maybe try to fix a program once or twice but otherwise just move on and try a new challenge. It’s easy to waste a lot of time on one challenge when you might be able to solve another problem.

SKIP AND MOVE ON

GET SOLVING