CS61A NOTE2 Lists,Dictionary,Sequences,Recursion,

Lists

>>> list_of_ints/bools/nested

list slicing `lst[<start index>:<end index>:<step size>]`.不特别注明则可以省略

纯文本

```
[6,5,4,3,2,1,0]
>>> lst[:3]
[6, 5, 4]
>>> lst[3:]
[3, 2, 1, 0]
>>> lst[::-1]
[0, 1, 2, 3, 4, 5, 6]
>>> lst[::2]
[6, 4, 2, 0]
```

List comprehensions 列表推导式

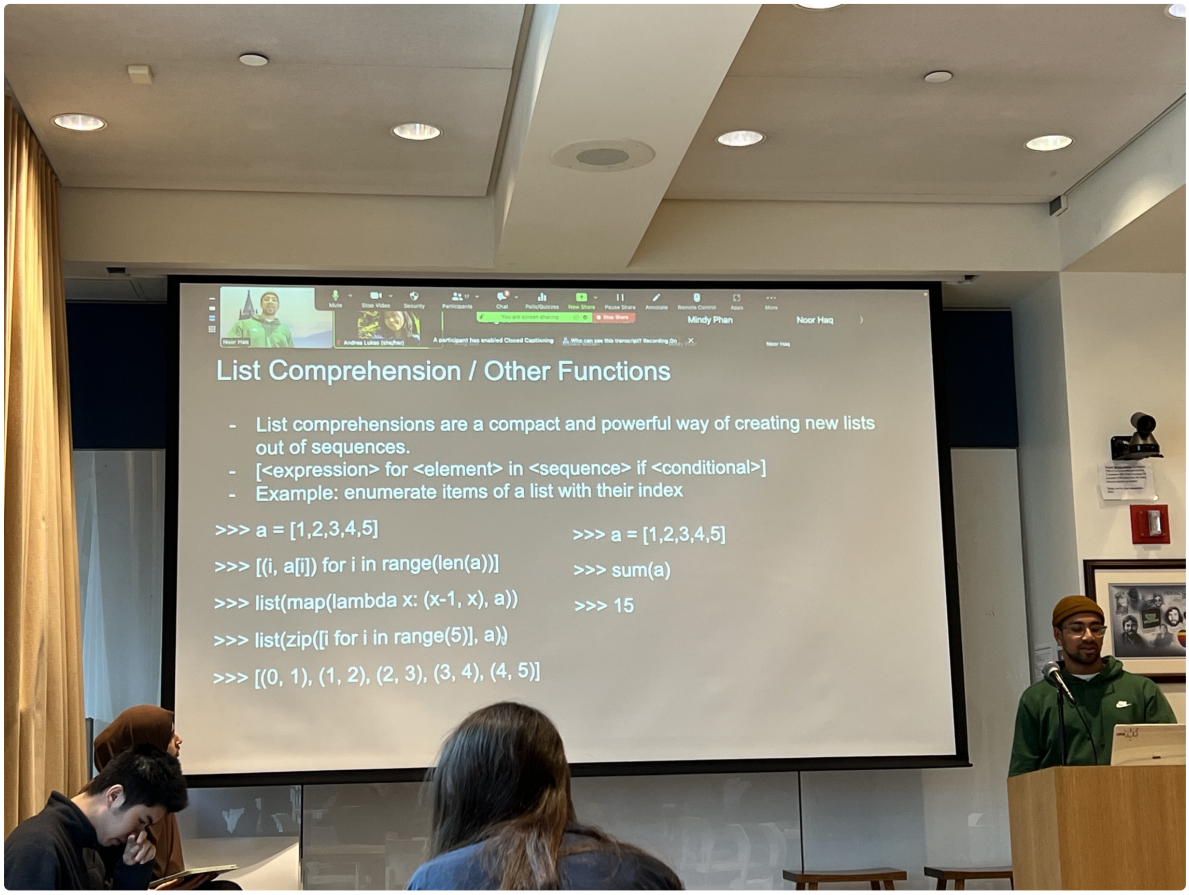
[<expression> for <element> in <sequence> if <conditional>]

纯文本

```
[i**2 for i in [1, 2, 3, 4] if i % 2 == 0] #方法一List comprehensions
[4,16]

lst = [] #方法二loop
for i in [1, 2, 3, 4]:
```

```
    if i % 2 == 0:
```



Lab04 Q4 WWPD

纯文本

```
>>> s = [7//3, 5, [4, 0, 1], 2]
>>> s[0]
2

>>> 4 in s
False

>>> 4 in s[2]
True

>>> s + [3 + 2, 9]
[2,5,[4,0,1],2,5,9] #默认计算
```

```
>>> s[2] * 2
[4,0,1,4,0,1]
```

纯文本

```
>>> x = [1, 2, 3, 4]
>>> x[1:3]
[2,3] #别多了

>>> x[:2]
[1,2]

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

>>> x[-2:3]
[3] #从倒数第2个到正数第三个，两边都易错

>>> x[-2:4]
[3,4]

>>> x[0:4:2]
[1,3]

>>> x[::-1]
[4,3,2,1]
```

disc Q4: Even weighted

Write a function that takes a list s and returns a new list that keeps only the even-indexed elements of s and multiplies them by their corresponding index.保留偶数 index 返回值乘相应的 index

纯文本

```
def even_weighted_loop(s): #方法一 用loop
    lst = []
    for i in range(0,len(s)):
        if i%2==0:
            lst += [i*s[i]]
    return lst
```

```
def even_weighted_loop(s): #方法二 用List comprehensions
    return [i*s[i] for i in range(0,len(s)) if i%2==0]
    #注意这里的range，若是判断i的话确实需要 range(0,len(s))
```

disc Q5: Max Product

Write a function that takes in a list and returns the maximum product 乘积 that can be formed using nonconsecutive 不连续 elements of the list. The input list will contain only numbers greater than or equal to 1. 两个重点：返回最大乘积，不连续，输入数字大于等于 0

纯文本

```
def max_product(s):
    if len(s)==0: #养成习惯
        return 1
    elif len(s)==1: #递归是两步
        return s[0]
    else:
        return max(s[-1] * max_product(s[:len(s)-2]),max_product(s[:len(s)-
```

Lab 5 Q1: Flatten

Write a function `flatten` that takes a list and returns a "flattened" version of it. The input list could be a deep list, meaning that there could be a multiple layers of nesting within the list.Make sure your solution does not mutate the input list.

Hint: you can check if something is a list by using the built-in `type` function. For example:

纯文本

```
>>> type([1, 2, 3]) == list
True
#无树 无skeam 无iterator 最原始方法令lst=[]
```

纯文本

```
#法一
lst=[]
    for i in s:
        if type(i)==list:
```

```
            lst+=flatten(i) #很明显需要递归，return的是【】因此不用加
        else: #因为有else不能用一行
            lst+=[i] #需要加【】
    return lst

#法二
lst=[]
    for i in range(0,len(s)): #注意range()
        if type(s[i])==list: #注意是s[i]
            lst+=flatten(s[i]) #有一点递归思想在里面
        else:
            lst+=[s[i]]
    return lst
```

Sequences 与列表类似

Sequences are ordered collections of values that support element-selection and have length. We've worked with lists, but other Python types are also sequences, including strings.

WWPD

纯文本

```
>>> x = 'Hello there Oski!'
>>> x
'Hello there Oski!'

>>> len(x)
17 #算空格

>>> x[6:]
'there Oski!'

>>> x = 'I am not Oski.'
>>> vowel_count = 0
```

```
>>> for i in range(len(x)):
...     if x[i] in 'aeiou':
...         vowel_count += 1
>>> vowel_count
5  #不区分大小写，I O也算
```

lab5 Q2: Map 映射（内置函数）

my_map takes in a one argument function fn and a sequence seq and returns a list containing fn applied to each element in seq.

一行(Hint: use a list comprehension)

```
def my_map(fn, seq):
    return [fn(i) for i in seq]
```

纯文本

lab5 Q3: Filter 选择（内置函数）

`my_filter` takes in a predicate function `pred` and a sequence `seq` and returns a list containing all elements in `seq` for which `pred` returns `True`. (A predicate function is a function that takes in an argument and returns either `True` or `False`.)

一行(Hint: use a list comprehension)

```
def my_filter(pred, seq):
    return [i for i in seq if pred(i)] #[.for.in.if.]
```

纯文本

lab5 Q4: Reduce（连续操作内置函数）

以上都是分别操作一行可写，本题连续的操作

`my_reduce` takes in a two argument function `combiner` and a non-empty 非空不用别的 if 了 sequence `seq` and combines the elements in `seq` into one value using `combiner`.

```
def my_reduce(combiner, seq): #嵌套函数套函数
    total=seq[0]     #非空给的提示
    for i in seq[1:]:     #从后面
        total=combiner(total,i)
    return total
```

纯文本

lab5 Q8: Count Palindromes 回文数

The Python library defines filter, map, and reduce, which operate on Python sequences. 前文提到

Devise a function that counts the number of palindromic words (those that read the same backwards as forwards) in a tuple of words using only lambda, basic operations on strings, the tuple constructor, conditional expressions, and the functions filter, map, and reduce. Specifically, do not use recursion or any kind of loop:不用循环只能用 list 里的 for 语句

Hint: The easiest way to get the reversed version of a string `s` is to use the Python slicing notation trick `s[::-1]`. Also, the function `lower`, when called on strings, converts all of the characters in the string to lowercase. For instance, if the variable `s` contains the string "PyThoN", the expression `s.lower()` evaluates to "python".回文不分大小写

```
def count_palindromes(L):
    return len([i for i in L if i.lower() == i[::-1].lower()])
```

纯文本

lab5 Q9: Coordinates 限制坐标

Implement a function `coords` that takes a function `fn`, a sequence `seq`, and a `lower` and `upper` bound on the output of the function. `coords` then returns a list of coordinate pairs `[x, fn(x)]`, contains only pairs whose y-coordinate is within the upper and lower bounds (inclusive)闭集

```
def coords(fn, seq, lower, upper):
    return [[i,fn(i)] for i in seq if lower<=fn(i)<=upper]
```
纯文本

Dictionary

Dictionaries are data structures which map keys to values. Dictionaries in Python are unordered, unlike real-world dictionaries --- in other words, key-value pairs are not arranged in the dictionary in any particular order. 字典是从 key 到 value 的数据结构，python 里的字典没有顺序。

The *keys* of a dictionary can be any *immutable* value, such as numbers, strings, and tuples.[1] 字典的 key 不可更改，key 和 value 都可为数字字符串元祖，数字不用加'',字符串要加''。Dictionaries themselves are mutable; we can add, remove, and change entries after creation. 字典可以在创建之后更改 There is only one value per key, however --- if we assign a new value to the same key, it overrides any previous value which might have existed.一个 key 一个值，新值覆盖旧值

To access the value of dictionary at key, use the syntax `dictionary[key]`.用 key 访问字典里的值 dictionary[key]

Element selection and reassignment work similarly to sequences, except the square brackets contain the key, not an index.操作与 sequence 类似，只是方括号 key 代替 index

To be exact, keys must be *hashable*. This means that some mutable objects, such as classes, can be used as dictionary keys.确切地说，键必须是可散列的。一些易变的对象，如类，可以作为字典的键。

```
pokemon = {'pikachu': 25, 'dragonair': 148}
```
纯文本

```
pokemon['mew'] = pokemon['pikachu']
pokemon[25] = 'pikachu'
pokemon['mewtwo'] = pokemon['mew'] * 2
>>> pokemon
{'pikachu': 25, 'dragonair': 148, 'mew': 25, 25: 'pikachu', 'mewtwo': 50}

pokemon[['firetype', 'flying']] = 146
>>> pokemon
Error: unhashable type #为什么
```

Note that the last example demonstrates that dictionaries cannot use other mutable data structures as keys. However, dictionaries can be arbitrarily deep, meaning the *values* of a dictionary can be themselves dictionaries.最后一个示例说明字典不能使用其他可变数据结构作为 key。然而字典可以是任意深的，字典的 value 本身可以是字典。