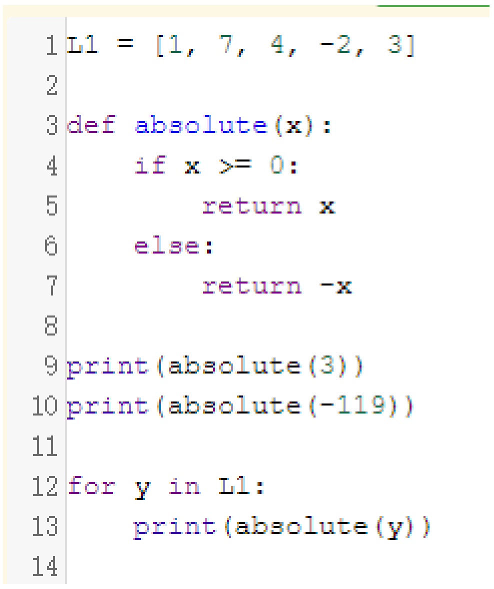
## 2.5 sorting

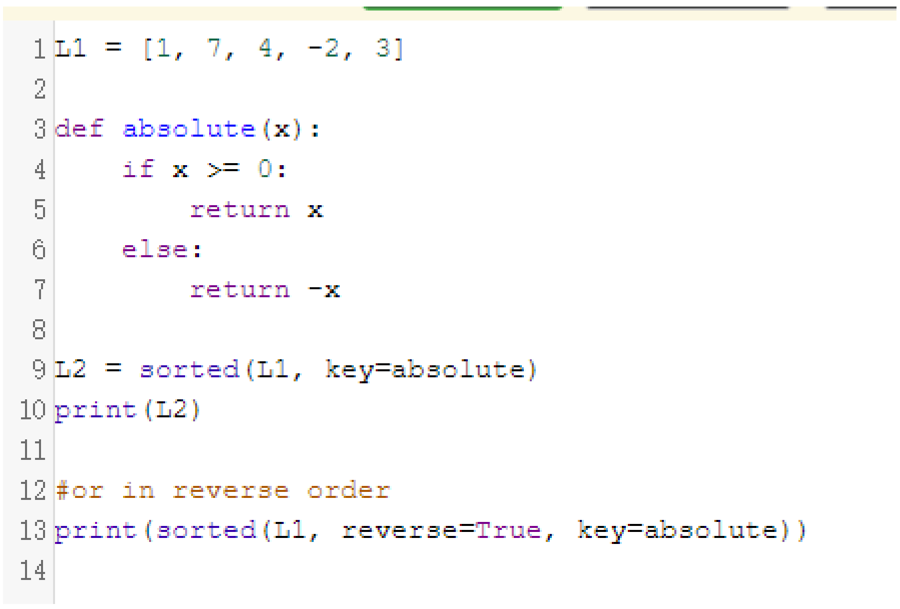
- sorting with sort and sorted
  - sort 改变了顺序之后就不能再变回来了
    - 单纯的对一个list去sort
      - L1 = [1, 7, 4, -2, 3]
      - L2 = ["Cherry", "Apple", "Blueberry"]
      - L1.sort()
      - print(L1)
      - L2.sort()
      - print(L2)
  - sorted 是一个function 通过将列表作为参数传递到括号内而在列表上调用的，而不是将列表放在句点之前。更重要的是，良序不会改变原始列表。
    - L2 = ["Cherry", "Apple", "Blueberry"]
    - L3 = sorted(L2)
    - print(L3)
    - print(sorted(L2))
    - print(L2) # unchanged 这里出来L2没有被改变
    - print("----")
    - L2.sort()
    - print(L2)
    - print(L2.sort()) #return value is None return出来是None
  - 有一个对比
    - print(sorted("Apple"))
    - 输出['A'，'e', 'l', 'p', 'p']
    - "Apple".sort()
    - 输出是error 因为是个string 无法更改
- Optional reverse parameter
  - 例子
    - L2 = ["Cherry", "Apple", "Blueberry"]
    - print(sorted(L2, reverse=True))
    - 输出
    - ['Cherry', 'Blueberry', 'Apple']
- optional key parameter

- 例子
  - 自己先定义了一个absolute function 但实际上python有内置的abs



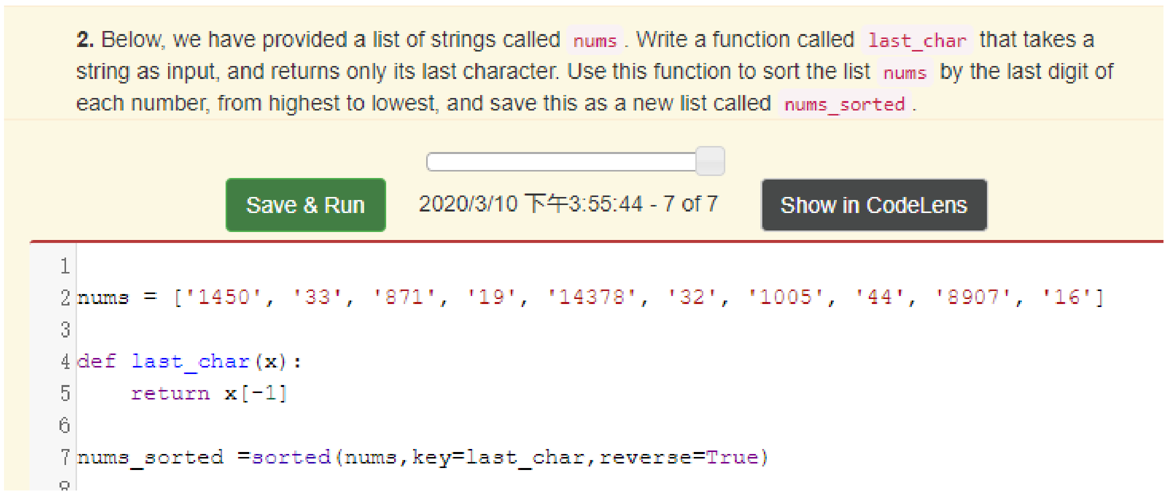
```
 1 L1 = [1, 7, 4, -2, 3]
 2
 3 def absolute(x):
 4     if x >= 0:
 5         return x
 6     else:
 7         return -x
 8
 9 print(absolute(3))
10 print(absolute(-119))
11
12 for y in L1:
13     print(absolute(y))
14
```

  - 用absolute去sorted list 实际上这里第九行的东西就是 absolute是一个function了



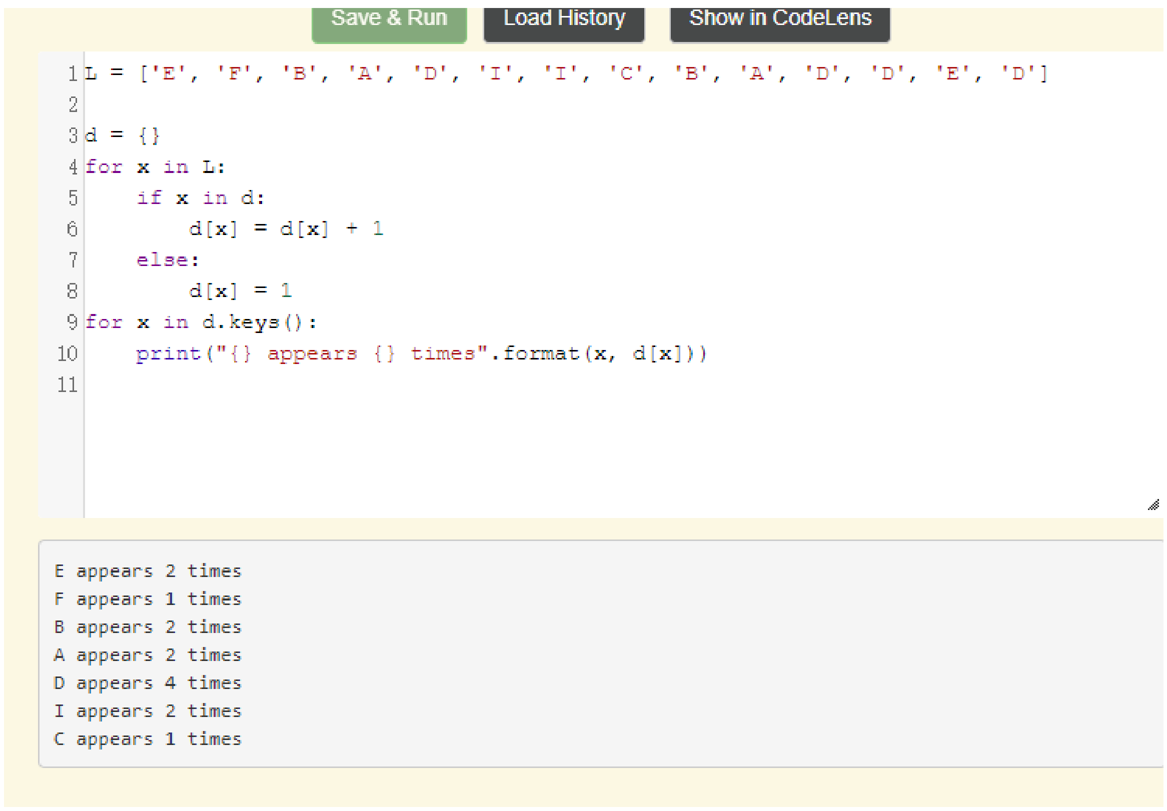
```
 1 L1 = [1, 7, 4, -2, 3]
 2
 3 def absolute(x):
 4     if x >= 0:
 5         return x
 6     else:
 7         return -x
 8
 9 L2 = sorted(L1, key=absolute)
10 print(L2)
11
12 #or in reverse order
13 print(sorted(L1, reverse=True, key=absolute))
14
```

- 有一个互相改写的案例
  - 用lambda改写的话就是 nums_sorted_lambda =sorted(nums,key= lambda x:x[-1],reverse=True)


**2.** Below, we have provided a list of strings called `nums`. Write a function called `last_char` that takes a string as input, and returns only its last character. Use this function to sort the list `nums` by the last digit of each number, from highest to lowest, and save this as a new list called `nums_sorted`.

```python

nums = ['1450', '33', '871', '19', '14378', '32', '1005', '44', '8907', '16']

def last_char(x):
    return x[-1]

nums_sorted =sorted(nums,key=last_char,reverse=True)
```

- sort a dictionary
  - 例子
    - 从一个list中去数每个出现了几次 然后print

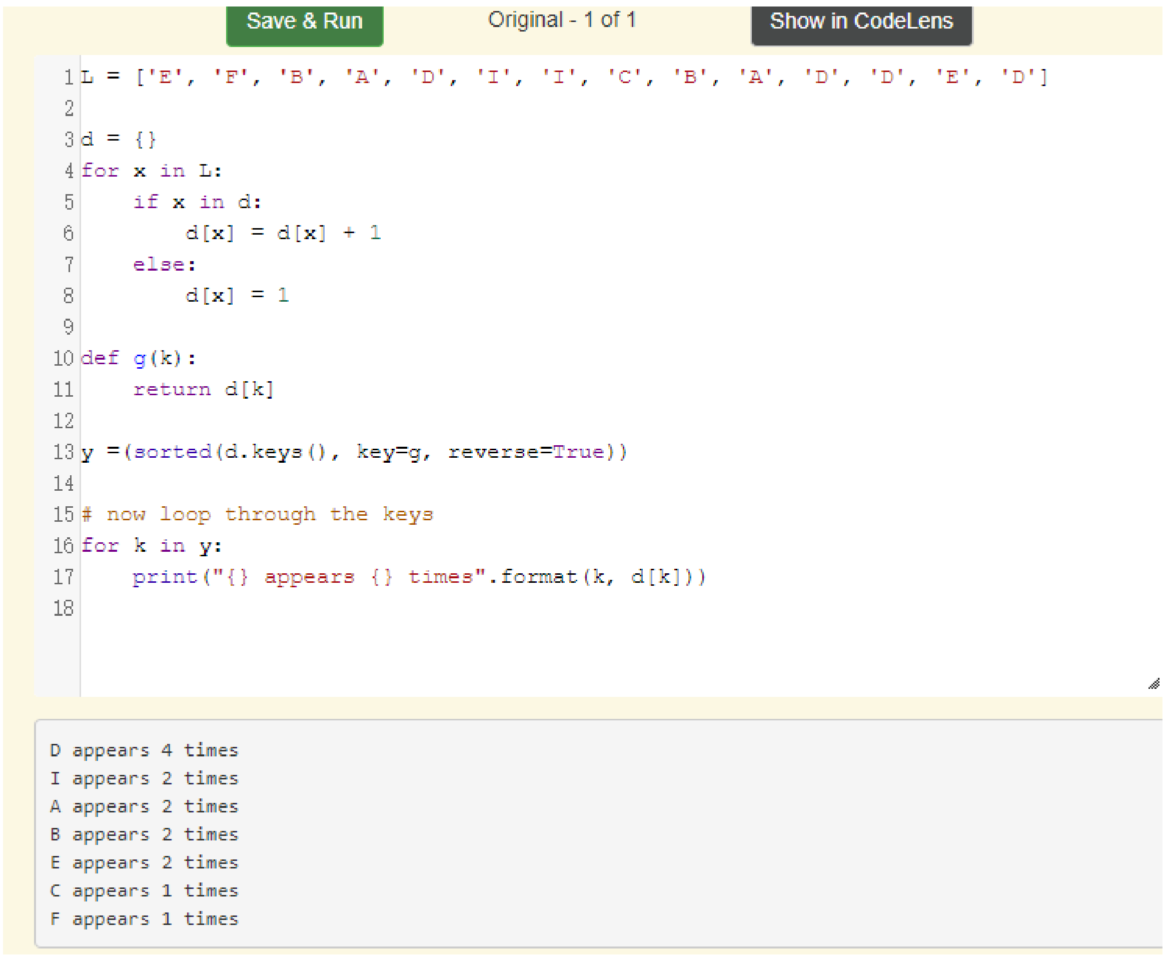


```python
L = ['E', 'F', 'B', 'A', 'D', 'I', 'I', 'C', 'B', 'A', 'D', 'D', 'E', 'D']

d = {}
for x in L:
    if x in d:
        d[x] = d[x] + 1
    else:
        d[x] = 1
for x in d.keys():
    print("{} appears {} times".format(x, d[x]))

```

```
E appears 2 times
F appears 1 times
B appears 2 times
A appears 2 times
D appears 4 times
I appears 2 times
C appears 1 times
```

    - 但是dic的key是没有固定的顺序的 所以需要自己sort
    - 比如
    - y = sorted(d.keys())
    - for k in y:
    - print("{} appears {} times".format(k, d[k]))
    - 用lambda改写的话就是
    - y = sorted(d.keys(), key=lambda k: d[k], reverse=True)   或者一个比较有经验的prgrammer直接会把前面的keys省略 直接d

- 这里要注意两个key是不同的含义 第一个是dictionary的key 第二个是在用于lambda function中需要的key=...的格式
- 如果想要用出现的次数排序
- key=lambda x: d[x]是格式！！！

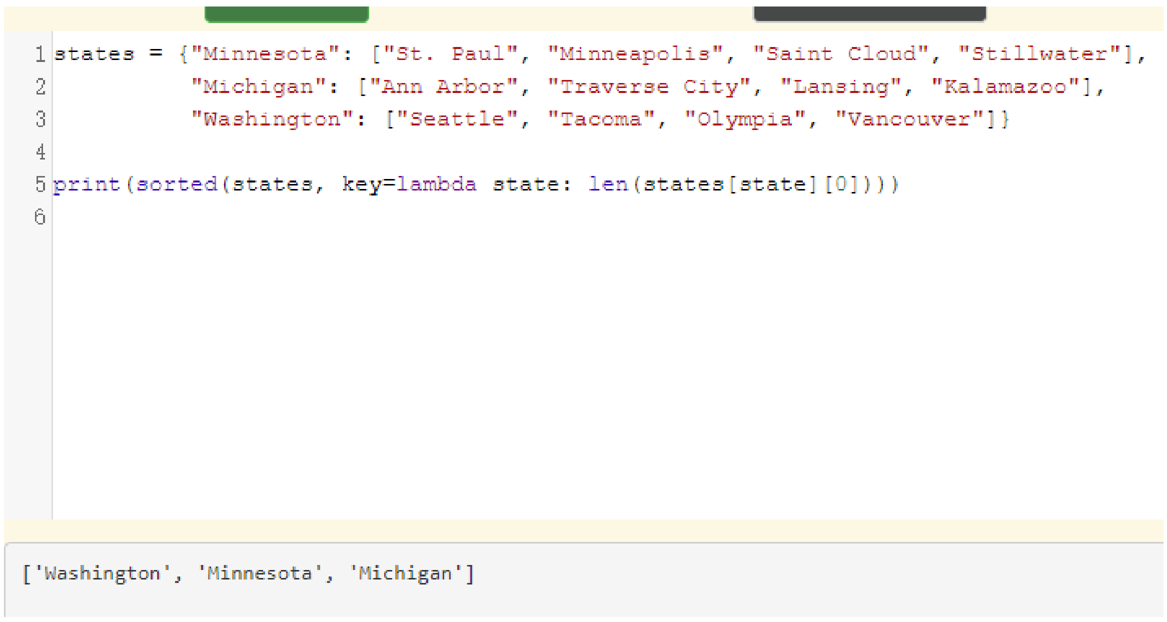


```python
L = ['E', 'F', 'B', 'A', 'D', 'I', 'I', 'C', 'B', 'A', 'D', 'D', 'E', 'D']

d = {}
for x in L:
    if x in d:
        d[x] = d[x] + 1
    else:
        d[x] = 1

def g(k):
    return d[k]

y =(sorted(d.keys(), key=g, reverse=True))

# now loop through the keys
for k in y:
    print("{} appears {} times".format(k, d[k]))

```

```
D appears 4 times
I appears 2 times
A appears 2 times
B appears 2 times
E appears 2 times
C appears 1 times
F appears 1 times
```

- Breaking ties
  - 例子
    - tups = [('A', 3, 2),
    - ('C', 1, 4),
    - ('B', 3, 1),
    - ('A', 2, 4),
    - ('C', 1, 2)]
    - for tup in sorted(tups):
    -     print(tup)
    - 输出
    - ('A', 2, 4)
    - ('A', 3, 2)
    - ('B', 3, 1)
    - ('C', 1, 2)
    - ('C', 1, 4)

  - 例子
    - fruits = ['peach', 'kiwi', 'apple', 'blueberry', 'papaya', 'mango', 'pear']
    - new_order = sorted(fruits, key=lambda fruit_name: (len(fruit_name), fruit_name))
    - for fruit in new_order:
    -     print(fruit)
    - 输出
    - kiwi
    - pear
    - apple
    - mango
    - peach
    - papaya
    - blueberry
    - 如果想要倒过来就加reverse=True但是同时alphabet的顺序也是倒过来的
    - 所以如果想要数字从大到小又a-z在len前面加个负号
- 什么时候用lambda
  - 如果lambda short和simple 用
  - 例子
    - 并不特别复杂



```
states = {"Minnesota": ["St. Paul", "Minneapolis", "Saint Cloud", "Stillwater"],
          "Michigan": ["Ann Arbor", "Traverse City", "Lansing", "Kalamazoo"],
          "Washington": ["Seattle", "Tacoma", "Olympia", "Vancouver"]}

print(sorted(states, key=lambda state: len(states[state][0])))
```

```
['Washington', 'Minnesota', 'Michigan']
```

    - 比较复杂

```python
def s_cities_count(city_list):
    ct = 0
    for city in city_list:
        if city[0] == "S":
            ct += 1
    return ct

states = {"Minnesota": ["St. Paul", "Minneapolis", "Saint Cloud", "Stillwater"],
          "Michigan": ["Ann Arbor", "Traverse City", "Lansing", "Kalamazoo"],
          "Washington": ["Seattle", "Tacoma", "Olympia", "Vancouver"]}

print(sorted(states, key=lambda state: s_cities_count(states[state])))
```

```
['Michigan', 'Washington', 'Minnesota']
```