

## Queues

```
Priority Queue
from queue import PriorityQueue

customers = PriorityQueue()
customers.put((2, "Harry"))

while customers:
    print(customers.get())
```

## Heaps

```
import heapq

heapq.heapify(li)    transform array "li" into min-heap
heapq._heapify_max(li)  transform array "li" into max-heap
heapq.heappush(li,4)  push element in the heap
heapq.heappop(li)    pop the smallest element
heapq.nlargest(3, li1)  get the 3 largest numbers
heapq.nsmallest(3, li1)  get the 3 smallest numbers
heapq.nsmallest(3, tuple_list, key=lambda x: x[1])  if we want 3 smallest tuple_list second dimension
```

## Deque

```
deque sized window = deque(maxlen=3)
```

## String

```
Split
string.split(" ", 1) //split first occ

Replace
str.replace('e', ")

Replace first occurrence
str.replace('e', ",1)

Remove symbols from phrase
paragraph = ".join([c.lower() if c.isalnum() else ' ' for c in paragraph])
```

## String (cont)

```
check array if matches
banned_words = set(banned)
for word in toCount:
    if word not in banned_words:
        ...
```

## Arrays

```
array of set capacity
LRUCache = [] * capacity

sort to new
new_list = sorted(old_list, key=..., reverse=...)

sort in place
list.sort(key=..., reverse=...)

sort by len
list.sort(key=len)

sort column
def takeSecond(elem):
    return elem[1]
list.sort(key=takeSecond)
```

## sort on objects

```
>>> class Student:
...     def __init__(self, name, grade, age):
...         self.name = name
...         self.grade = grade
...         self.age = age
...     def __repr__(self):
...         return repr((self.name, self.grade, self.age))
>>> student_objects = [
...     Student('john', 'A', 15),
...     Student('jane', 'B', 12),
...     Student('dave', 'B', 10),
... ]
>>> sorted(student_objects, key=lambda student: student.age)
[('dave', 'B', 10), ('jane', 'B', 12), ('john', 'A', 15)]
```

## Dict

```
create dict = {}

dict

return dict.keys()

keys arr

return dict.values()

values arr

order a dict by value
from operator import itemgetter
sorted_keys = sorted(toReduce.items(), key=itemgetter(1))

orderdict
from collections import OrderedDict
d = OrderedDict()
d.move_to_end('key')
```

## Dict (cont)

```
orderdict by value
from collections import OrderedDict
from operator import itemgetter
OrderedDict(sorted(d.items(), key = itemgetter(1), reverse = True))

put in first position
d.move_to_end('key',last=False)

comprehension
dict_variable = {key:value*2 for (key,value) in dictionary.items()}

dict with array as key
dictionary[tuple(arr)] = val
```

## Classes

```
initialise __init__(self, var1, var2)
create iterable __iter__(self)
```

## Files

```
Read file wfile = open(wordFile, ' r')

split line
for line in wfile:
    word = line[:-1]
```

