Python 3 Cheatsheet: Sorting & Functional

Sort

`sorted(iterable, key=key, reverse=reverse)` vs `list.sort()` (in-place).

```
# Basic
sorted(['Ford', 'BMW', 'Volvo'])  # ['BMW', 'Ford', 'Volvo']
cars.sort(key=len)                # Sort by length

# Custom Keys (Lambda)
# Sort by 2nd element, then 1st
nested = [(5,10), (2,20), (2,3)]
sorted(nested, key=lambda x: (x[1], x[0]))

# Sort Dict by Value
d = {'a':3, 'b':0}
sorted(d, key=lambda k: d[k])     # ['b', 'a']
```

Zip

Combine iterables.

```
list(zip([1,2], ['a','b']))       # [(1, 'a'), (2, 'b')]
dict(zip(keys, values))           # Create dict

# Parallel Iteration
for l, n in zip(letters, numbers):
    print(l, n)

# Unzip / Transpose Matrix
matrix = [[1,2],[3,4],[5,6]]
list(zip(*matrix))                # [(1, 3, 5), (2, 4, 6)]
# Rotate Matrix 90 deg clockwise
[list(t) for t in zip(*matrix[::-1])]
```

Map, Filter, Reduce

```
# Map: Apply function to all
list(map(str.upper, ['a', 'b']))  # ['A', 'B']

# Filter: Keep if True
list(filter(lambda x: x > 5, nums))

# Reduce: Accumulate
```

```
from functools import reduce
reduce(lambda x, y: x+y, [1,2,3]) # 6
```

Itertools

```
import itertools
# Accumulate (Prefix Sum)
list(itertools.accumulate([1,2,3])) # [1, 3, 6]

# GroupBy
for k, v in itertools.groupby("aabbbc"):
    print(k, list(v))             # a [a,a], b [b,b,b], c [c]
```

Any / All

```
any([False, True, False])         # True
all([True, True, True])           # True
```