|  |  |
| --- | --- |
| def **find\_duplicates**(lst\_nums):  duplicates = set()  seen = set()  for num in lst\_nums:  if num in seen:  duplicates.add(num)  seen.add(num)  return duplicates |  |
| def **removeDuplicate**(test\_list):  res = []  for i in test\_list:  if i not in res:  res.append(i)  return "".join(res) |  |
| # Reverse string using recursion  def **reverse**(string):  if len(string) <= 1:  return string  return **reverse**(string[1:]) +string[0]  **reverse**("hello") |  |
| **#** word count in a sentence.  sentence = "my name is Robin is is are"  def **word\_count**(sente):  words = sente.split()  return {i: words.count(i) for i in words} |  |
| # Reading from line by line  f\_name = "file1.txt"  **try:**  **content = []**  with open(f\_name) as f:  lines = f.readlines()  for line in lines:  content.append(line)  print(content[:5]  **except** **FileNotFoundError**:  msg = f" Can't find file: {f\_name}." |  |
| **#** extract digits from given string  string = "dfdjnjd8439dnjvndj"  result = "\n".**join**(  **filter**(**lambda** idx : idx.isdigit(), string))  print(result) |  |
| **# Lambda Function**  add5 = lambda x : x+5  print(add5(7)) |  |
| **# map function**  def **addition**(n):  return n + n  numbers = (1, 2, 3, 4)  result = map(**addition**, numbers)  print(list(result)) |  |
| # modify item at index  # access items around index  lst = [11, 18, 9, 12, 23, 4, 17]  lost = []  for idx, val in enumerate(lst):  if val > 15:  lost.append(val)  lst[idx] = 15  print(lst, lost) |  |
| def **iterator1**():  lst = []  for i in range(10000):  lst.append(i)  return lst  iterator1() |  |
| # Generator objects iterates  # only once  def **generator1**():  for i in range(5):  yield i  a = **generator1**()  for j in a:  print(j) |  |
| **# Combine two list into tuples**  a = ['a','b','c'] b = [1,2,3]  [(k,v) for k,v in zip(a,b)] |  |
| Instance methods : accept self parameter and relate to a specific instance of the class.  Class methods : accept cls parameter and can modify the class itself  class CoffeeShop:  specialty = 'espresso'    def \_\_init\_\_(self, coffee\_price):  self.coffee\_price = coffee\_price    # instance method  def make\_coffee(self):  print(f'{self.specialty} for ${self.coffee\_price}')    @staticmethod  def check\_weather():  print('Its sunny')    @classmethod  def change\_specialty(cls, specialty):  cls.specialty = specialty  print(f'changed to {specialty}')  coffee\_shop\_obj = CoffeeShop('5')  coffee\_shop.make\_coffee() |  |