```
"""
Code Challenge
  Name:
    generator
  Filename:
    generator.py
  Problem Statement:
    This program accepts a sequence of comma separated numbers from
user
    and generates a list and tuple with those numbers.
  Input:
    2, 4, 7, 8, 9, 12
  Output:
    List : ['2', ' 4', ' 7', ' 8', ' 9', '12']
    Tuple : ('2', ' 4', ' 7', ' 8', ' 9', '122')
"""


"""
Code Challenge
  Name:
    weeks
  Filename:
    weeks.py
  Problem Statement:
    Write a program that adds missing days to existing tuple of days
  Input:
    ('Monday', 'Wednesday', 'Thursday', 'Saturday')
  Output:
    ('Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday',
'Saturday', 'Sunday')
"""




"""
Code Challenge
  Name:
    Supermarket
  Filename:
    supermarket.py
  Problem Statement:
    You are the manager of a supermarket.
    You have a list of items together with their prices that consumers
bought on a particular day.
    Your task is to print each item_name and net_price in order of its
first occurrence.
    Take Input from User
  Hint:
    item_name = Name of the item.
    net_price = Quantity of the item sold multiplied by the price of
each item.
    try to use new class for dictionary : OrderedDict
  Input:
    BANANA FRIES 12
    POTATO CHIPS 30
    APPLE JUICE 10
    CANDY 5
    APPLE JUICE 10
    CANDY 5
    CANDY 5
    CANDY 5
    POTATO CHIPS 30
  Output:
    BANANA FRIES 12
    POTATO CHIPS 60
    APPLE JUICE 20
    CANDY 20

"""



"""
Code Challenge
  Name:
    Teen Calculator
  Filename:
    teen_cal.py
  Problem Statement:
    Take dictionary as input from user with keys, a b c, with some
integer
    values and print their sum. However, if any of the values is a teen
--
    in the range 13 to 19 inclusive -- then that value counts as 0,
except
    15 and 16 do not count as a teens. Write a separate helper "def
    fix_teen(n):"that takes in an int value and returns that value
fixed for
    the teen rule. In this way, you avoid repeating the teen code 3
times
  Input:
    {"a" : 2, "b" : 15, "c" : 13}
  Output:
    Sum = 17
"""



"""
Code Challenge
  Name:
    Character Frequency
  Filename:
    frequency.py
  Problem Statement:
    This program accepts a string from User and counts the number of
characters (character frequency) in the input string.
  Input:
    www.google.com
  Output:
    {'c': 1, 'e': 1, 'g': 2, 'm': 1, 'l': 1, 'o': 3, '.': 2, 'w': 3}
"""
```

```
"""
Code Challenge
  Name:
    Digit Letter Counter
  Filename:
    digit_letter_counter.py
  Problem Statement:
    Write a Python program that accepts a string from User and
calculate the number of digits and letters.
  Hint:
    Store the letters and Digits as keys in the dictionary
  Input:
    Python 3.2
  Output:
    Digits 2
    Letters 6
"""


"""
Two words are anagrams if you can rearrange the letters of one to spell
the second.
For example, the following words are anagrams:

 ['abets', 'baste', 'bates', 'beast', 'beats', 'betas', 'tabes']

Hint: How can you tell quickly if two words are anagrams?
Dictionaries allow you to find a key quickly.

"""



"""
Code Challenge
  Name:
    Intersection
  Filename:
    Intersection.py
  Problem Statement:
    With two given lists [1,3,6,78,35,55] and [12,24,35,24,88,120,155]
    Write a program to make a list whose elements are intersection of
the above given lists.
"""



"""
Code Challenge
  Name:
    Duplicate
  Filename:
    duplicate.py
  Problem Statement:
    With a given list [12,24,35,24,88,120,155,88,120,155]
    Write a program to print this list after removing all duplicate 
values
    with original order reserved
"""


"""
Code Challenge
  Name: 
    Mailing List
  Filename: 
    mailing.py
  Problem Statement:
  I recently decided to move a popular community  mailing list (3,000 
subscribers, 
  60-80 postings/day) from my server to Google Groups. 
  I asked people to joint he Google-based list themselves, 
  and added many others myself, as the list manager. 
  However, after nearly a week, only half of the list had been moved. 
  I somehow needed to learn which people on the old list hadn't yet 
l  signed up for the new list.


  Fortunately, Google will let you export a list of members of a group 
to 
  CSV format. 
  Also, Mailman (the list-management program I was using on 
  my server) allows you to list all of the e-mail addresses being used 
  for a list. Comparing these lists, I think, offers a nice chance to 
look 
  at several different aspects of Python, and to consider how we can 
  solve this real-world problem in a "Pythonic" way.


  The goal of this project is thus to find all of the e-mail addresses 
on 
  the old list that aren't on the new list. The old list is in a file 
  containing one e-mail address per line, as in:

  Hint: 
      Refer to mailing.txt for the new and old list of email addresses.

"""
```