```
# -ProblemSet2.py *- coding: utf-8 -*-
"""
Each problem will be a function to write.

Remember that you can execute just the code between the #%% signs by clicking
somewhere in that space and the using Ctrl-Enter (Cmd-Enter on Mac). An
alternative is to use the second toolbar green triangle or Menu>Run>Run cell.

On loops especially, you can make an error that causes the program to run
forever. If you don't get immediate response, then this is probably happening.
In that case, try Ctrl-C. If that doesn't stop it click your IPython console
away and open a new one. Look over you code and see why the termination
condition can't be met and fix it. Then run again.

"""
"""
Problem 2_1:
Write a function 'problem2_1()' that sets a variable lis = list(range(20,30)) and
does all of the following, each on a separate line:
(a) print the element of lis with the index 3
(b) print lis itself
(c) write a 'for' loop that prints out every element of lis. Recall that
    len() will give you the length of such a data collection if you need that.
    Use end=" " to put one space between the elements of the list lis.  Allow
    the extra space at the end of the list to stand, don't make a special case
    of it.
"""
#%%
def problem2_1():
    pass # replace this pass (a do-nothing) statement with your code








#%%
"""
Test run:

problem2_1()
23
[20, 21, 22, 23, 24, 25, 26, 27, 28, 29]
20 21 22 23 24 25 26 27 28 29

"""
"""
Problem 2_2:
Write a function 'problem2_2()' that takes a list and does the following to it.
Actually, I've started the function for you below. Your function should do all
of the following, each on a separate line. Recall that lists start numbering
with 0.
0) print the whole list (this doesn't require a while or for loop)
1) print the item with index 0
2) print the last item in the list
3) print the items with indexes 3 through 5 but not including 5
4) print the items up to the one with index 3 but not including item 3
5) print the items starting at index 3 and going through the end.
6) print the length of the list ( use len() )
7) Use the append() method of a list to append the letter "z" onto a list.
   Print the list with z appended.

Make sure that your function also works with blist below.  For this to work,
you cannot use alist as a variable inside your function.
"""
#%%

alist = ["a","e","i","o","u","y"]
blist = ["alpha", "beta", "gamma", "delta", "epsilon", "eta", "theta"]

def problem2_2(my_list):
    pass # replace this pass (a do-nothing) statement with your code













#%%
"""
Test run, two of them. The same function should work with either list. The
grader function will use different lists.

problem2_2(alist)
['a', 'e', 'i', 'o', 'u', 'y']
a
y
['o', 'u']
['a', 'e', 'i']
['o', 'u', 'y']
6
['a', 'e', 'i', 'o', 'u', 'y', 'z']

problem2_2(blist)
['alpha', 'beta', 'gamma', 'delta', 'epsilon', 'eta', 'theta']
alpha
theta
['delta', 'epsilon']
['alpha', 'beta', 'gamma']
['delta', 'epsilon', 'eta', 'theta']
7
['alpha', 'beta', 'gamma', 'delta', 'epsilon', 'eta', 'theta', 'z']


"""

"""
Problem 2_3:
Write a function problem2_3() that should have a 'for' loop that steps
through the list below and prints the name of the state and the number of
letters in the state's name. You may use the len() function.
Here is the output from mine:
In [70]: problem2_3(newEngland)
Maine has 5 letters.
New Hampshire has 13 letters.
Vermont has 7 letters.
Rhode Island has 12 letters.
Massachusetts has 13 letters.
Connecticut has 11 letters.

The function is started for you.  The grader will not use the list newEngland
so don't use the variable newEngland inside your function.
"""
```

```
#%%
newEngland = ["Maine","New Hampshire","Vermont", "Rhode Island",
"Massachusetts","Connecticut"]

def problem2_3(ne):
    pass # replace this pass (a do-nothing) statement with your code


#%%
"""
Problem 2_4:
random.random() generates pseudo-random real numbers between 0 and 1. But what
if you needed other random reals? Write a program to use only random.random()
to generate a list of random reals between 30 and 35. This is a simple matter
of multiplication and addition. By multiplying you can spread the random numbers
out to cover the range 0 to 5. By adding you can shift these numbers up to the
required range from 30 to 35.  Set the seed in this function to 70 so that
everyone generates the same random numbers and will agree with the grader's
list of random numbers. Print out the list (in list form).
"""
#%%
import random

def problem2_4():
    """ Make a list of 10 random reals between 30 and 35 """
    random.seed()
    pass # replace this pass (a do-nothing) statement with your code

#%%
"""
COMMENT: Note that this uses a pseudorandom number generator.  That means
that the list will be different for each person.  We issue the command
random.seed(70) inside the function problem2_4() to insure that we generate the
same numbers that the grader expects. If you do this problem correctly, you
should get the list of random numbers below.

Test run:

problem2_4()
[34.54884618961936, 31.470395203793395, 32.297169396656095, 30.681793552717807,
 34.97530360173135, 30.773219981037737, 33.36969776732032, 32.990127772708405,
 33.57311858494461, 32.052629620057274]
""""""
Problem 2_5:
Let's do a small simulation. Suppose that you rolled a die repeatedly. Each
time that you roll the die you get a integer from 1 to 6, the number of pips
on the die. Use random.randint(a,b) to simulate rolling a die 10 times and
printout the 10 outcomes. The function random.randint(a,b) will
generate an integer (whole number) between the integers a and b inclusive.
Remember each outcome is 1, 2, 3, 4, 5, or 6, so make sure that you can get
all of these outcomes and none other. Print the list, one item to a line so that
there are 10 lines as in the example run.  Make sure that it has 10 items
and they are all in the range 1 through 6.  Here is one of my runs. In
the problem below I ask you to set the seed to 171 for the benefit of the
auto-grader. In this example, that wasn't done and so your numbers will be
different.  Note that the seed must be set BEFORE randint is used.

problem2_5()
4
5
3
1
4
3
5
```

```
1
6
3

"""
"""
Problem 2_5:
"""
import random

def problem2_5():
    """ Simulates rolling a die 10 times."""
    # Setting the seed makes the random numbers always the same
    # This is to make the auto-grader's job easier.
    random.seed(171)  # don't remove when you submit for grading
    pass # replace this pass (a do-nothing) statement with your code

#%%
"""
Problem 2_6:
Let's continue with our simulation of dice by rolling two of them. This time
each die can come up with a number from 1 to 6, but you have two of them. The
result or outcome is taken to be the sum of the pips on the two dice. Write a
program that will roll 2 dice and produce the outcome. This time let's roll
the two dice 100 times. Print the outcomes one outcome per line.
"""
#%%
import random

def problem2_6():
    """ Simulates rolling 2 dice 100 times """
    # Setting the seed makes the random numbers always the same
    # This is to make the auto-grader's job easier.
    random.seed(431)  # don't remove when you submit for grading
    pass # replace this pass (a do-nothing) statement with your code


#%%
"""
Test run with seed 82, but make sure that you submit with the seed 431:

problem2_6()
6
8
4
9
3
8
6
5
7
5
7
6
5
6
3
9
4
8
11
'
'
'
9
```

```
6
7
10
4

"""

"""
Problem 2_7:
Heron's formula for computing the area of a triangle with sides a, b, and c is
as follows. Let s = .5(a + b + c) --- that is, 1/2 of the perimeter of the
triangle. Then the area is the square root of s(s-a)(s-b)(s-c). You can compute
the square root of x by x**.5 (raise x to the 1/2 power). Use an input
statement to get the length of the sides. Don't forget to convert this input
to a real number using float(). Adjust your output to be just like what you
see below. Here is a run of my program:

problem2_7()

Enter length of side one: 9

Enter length of side two: 12

Enter length of side three: 15
Area of a triangle with sides 9.0 12.0 15.0 is 54.0

"""
#%%

def problem2_7():
    """ computes area of triangle using Heron's formula. """
    pass # replace this pass (a do-nothing) statement with your code

#%%
"""
Problem 2_8:
The following list gives the hourly temperature during a 24 hour day. Please
write a function, that will take such a list and compute 3 things: average
temperature, high (maximum temperature), and low (minimum temperature) for the
day.  I will test with a different set of temperatures, so don't pick out
the low or the high and code it into your program. This should work for
other hourly_temp lists as well. This can be done by looping (interating)
through the list. I suggest you not write it all at once. You might write
a function that computes just one of these, say average, then improve it
to handle another, say maximum, etc. Note that there are Python functions
called max() and min() that could also be used to do part of the jobs.
"""
#%%
hourly_temp = [40.0, 39.0, 37.0, 34.0, 33.0, 34.0, 36.0, 37.0, 38.0, 39.0, \
               40.0, 41.0, 44.0, 45.0, 47.0, 48.0, 45.0, 42.0, 39.0, 37.0, \
               36.0, 35.0, 33.0, 32.0]
#%%
def problem2_8(temp_list):
    pass # replace this pass (a do-nothing) statement with your code


#%%
"""
Sample run using the list hourly_temp. Note that the grader will use a
different hourly list.  Be sure that you function works on this list and test
it on at least one other list of your own construction.
Note also, that the list the grader uses may not have the same number of items
as this one.

problem2_8(hourly_temp)
```

```
Average: 38.791666666666664
High: 48.0
Low: 32.0
"""
```