Exercise 3-1:
Add a (lower level) hierarchy under **myfirstpackage** directory, using <ex32.py> as the lower level module so that it can be called/used/imported as myfirstpackage.test.ex32.
Hints: create a directory and remember what files need to be inside that directory (2 files are required)

Add `<module1.py> and <another_module.py>` that you created earlier under this package so that they can be called/used as myfirstpackage.module1 and myfirstpackage.another_module

test by importing ex32 and execute/call its"multiply" function from a file <exercise3-1.py>
test module1 and another_module the same way from <exercise3-1.py>

Zip the whole package and files then submit the zip file on SFS

Exercise 3-2:
For **Calorie and Expense Tracker** (In the presentation slide)
Implement/write the required codefor the functions required(intake, expenses, release), then just like Exercise 3-1 make them into a package, call the package calorietracker.
As decsribed in the presentation materials call the python file/module inside this package <calexp.py>

Optional:
Add one more lower level of hierarchy. You can add/use any function(s) you want as this lower level hierarchy.

Test by importing and executing the functions (intake, expenses, release)

Zip the whole package and files and submit the zip file on

SFS. Bonus point: (do your research)
add this line
if __name__ == '__main__':
for assigning “default values”

The expected format/type of the parameter of the functions are dictionary
e.g
`{'rice': 2, 'chicken':1, 'beef': 1, 'veggies': 2, 'ramen': 1 }` for consumption in these functions: intake , expense

`{'rest': 12, 'class': 5, 'walk': 2, 'bike': 1 }` for activities in this function: release

For example:

```
intake({'rice':   2,   'chicken':1,   'beef':   1,
'veggies': 2, 'ramen': 1 })

release({'rest':   12,   'class':   5,   'walk':   2,
'bike': 1 })
```

Hints:
#1
Use dictionary (Lesson 02) to create the required these data structures (dictionary) as inputs, for example(just an example, item name does not match <calexp.py> in teaching materials!-you need to create a proper/correct one based on <calexp.py> in teaching materials!

```
meal_data = {   'burger':(400,250), 'sandwich':(300,200),
'donut':(200,300) }
```

with 400 being the calory for burger per serve and 250 being the price per serve and so on

```
activity_data = { 'jogging': 150, 'swimming': 50, 'hiking':
200}
```

with 150 being the calories burned for jogging and so on

#2
To perform accumulation (total) you can use this format
`a+=b` (this means a=a+b)

#3
To loop through the dictionary you could use this format (Lesson 02)
```
for act, qty in activities.iteritems():
```

#4

intake
Because consumption is specified as a dictionary,
go through the dictionary's key and values and perform the required calculation :

calories = calories+ (meal_data's_first_element * quantity_consumed)

return the calories!

expense
Because activities is specified as a dictionary,
go through the dictionary's key and values and perform the required calculation :

total_expense = total_expense+ (meal_data's _second_element * quantity_consumed)

return the total_expense

release
Because consumption is specified as a dictionary,
go through the dictionary's key and values and perform the required calculation :

calories = calories+ (activities_data)

return the calories

IMPORTANT HINT
try these

```
meal_data = {    'burger':(400,250), 'sandwich':(300,200),
'donut':(200,300) }
```

meal_data['burger']
meal_data['burger'][0]
meal_data['burger]'[1]

try the same for meal_data['sandwich'] and so on

EXPECTED functionality:
In your top level python file (call it exercise3-2.py), after creating the package above you should be able to execute these lines. (submit the top level file along /zip everything)

```
import calorieracker.calexp
a=calorietracker.calexp.intake({'rice':         2,
'chicken':1, 'beef': 1, 'veggies': 2, 'ramen': 1
})
b=calorietracker.calexp.expense({'rice':        2,
'chicken':1, 'beef': 1, 'veggies': 2, 'ramen': 1
})
c=calorietracker.calexp.release({'rest':       12,
'class': 5, 'walk': 2, 'bike': 1 })

print a,b,c
```