```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Example 2: pure python

@author Austin Downey
"""

# clears everything, including IPython
import IPython as IP
IP.get_ipython().magic('reset -sf')

# import modules native to Python, use the "as" to not get confused
import random as random # random number generator
import sys as sys # import systems functions
import os as os # import os functions

# <-- makes a comment

''' <-- makes a multi
line
comment '''

#%% makes a section, means nothing in the code, helpful for splitting up code


print('hello world') # prints a string to the console.
print("hello world") # prints the same to the console, but you have to hit the shift key.
print(' "hello world" ' ) # allows you to print the quotation marks to the console.

# print will retrun a string, form anything it can.
print('string')
print(10) # number
print([10]) # string
# but it cant print two typbes of variables unless seperated by a ,
print('string',10,[10])
# or concantated into a string
print('string '+ str(10)+ ' ' + str([10]))



#%% arithmetic operations
# There are seven different arithmetic operations
'''
+ addition - yields the sum of its arguments
- subtraction - yields the difference of its arguments
* multiplication -  yields the product of its arguments
/ division  -  yields  the quotient of its arguments
% modulo - yields the remainder from the division of the first argument by the second.
** exponential -  yields the power (or exponentiation) of its arguments
// floor division - yield the quotient of its arguments with the 'floor' function
 applied to the result
'''

print('5 + 2 = ', 5+2)
print('5 - 2 = ', 5-2)
print('5 * 2 = ', 5*2)
print('5 / 2 = ', 5/2)
print('5 % 2 = ', 5%2)
print('5 ** 2 = ', 5**2)
print('5 +// 2 = ', 5//2)

# order of operation matters in Python,
print('1 + 2 - 3 * 2 = ',1 + 2 - 3 * 2 )
print('(1 + 2 - 3) * 2 = ',(1 + 2 - 3) * 2 )

```

```
#%% variables.

# variables have to start with a number and can have numbers and underscores

name = 'Austin'
last_name = 'Downey'

''' there are lots of data types (classes) in Python, we will use seven main ones.
integers (class int)
floats (class float)
booleans (class bool)
strings (class str)
lists (class list)
tuples (class tuple)
dictionaries  (class dict)
'''

ii = 10
ff = 10/2
bb = False
ss = 'Austin'
ll = [1,2,'3',ss,bb] # a list of anything.
# a list will copy, print list, change bb to True, reprint list.
tt = (6,7,8,9,10) # a list of anything that you can not change.
dd = {'ii':ii,
      'ff':ff,
      'bb':bb,
      'ss':ss,
      'll':ll,
      'tt':tt}

#%% control flow statements.

''' The Python for statement iterates over the members of a sequence in order,
executing the block each time. '''

for i in range (0,3):
    print("We're on time %d" % (i))

''' the ''while'' loop, used when a condition needs to be checked each iteration
or to repeat a block of code forever. '''

x = 1
while x<100:  #True:
    print("To infinity and beyond! We're getting close, on %d now!" % (x))
    x += 1


''' The while loop and the for loop can be made to exit before the given object
is finished. This is done using the break statement, which will immediately drop
 out of the loop and continue execution at the first statement after the block. '''

for x in range(3):
    if x == 1:
        break

''' You can also have an optional else clause, which will run should the for loop
exit cleanly - that is, without breaking. '''

for x in range(3):
    print(x)
else:
    print('Final x = %d' % (x))


''' Lastly, you can directly use an "if statement", which will allow you to cycle
```

```
through statements to check. '''

i='d'
if i == 'a':
    print('the variable is ' + i)
elif i == 'b':
    print('the variable is ' + i)
elif i == 'c':
    print('the variable is ' + i)
elif i == 'd':
    print('the variable is ' + i)
else:
    print('the variable is a mystery')


```