A Small Python Reference

```
# This is a comment
# Whitespace is meaningful in python!
# (and tabs != spaces)
```

Types:

```
x = 3  # variable x is an int w/ value 3
y = 4.2  # float
s = 'Hello!'  # string
t = "Hello!"  # strings can be made with ' or "
b = True  # boolean values are True and False
```

Operators:

```
**  # exponent (3**2 evaluates to 9)
*  # multiply
/  # divide
%  # modulo (remainder, 7 % 2 evaluates to 1)
+  # add
-  # subtract
```

Equivalence operators:

```
==  # equals
!=  # not equals
>=  # greater than or equal to
<=  # less than or equal to
>  # greater than
<  # less than
```

Conditional statements:

```
if test:
      # runs if test is true
elif test2:
      # runs if test is false
      # and test2 is true
      # you can't have an elif branch
      # without an if!
else:
      # runs if none of the
      # first branches ran

# keywords like 'and', 'or' and 'not'
# can be used in tests:
if  x < y and not y < 7:
      print 'y is < 7 but more than x!'

if x % 2 == 1 or y % 2 == 1:
      print 'We have an odd number!'
```

Loops:

```
# prints numbers 0 through 4
for i in range(5):
      print i

# loop through a collection
for element in collection:
      # do things with element
```

(Loops, cont'd)

```
# loop until a condition is met:
i = 0
while i < 105:
      i = i + random.randint(1, 99)
      print i
```

Collections:

```
l = [1, 8, -5]  # a list
# access list elements:
l[0]  # 1
l[1]  # 8
l[2]  # -5
# get the size of a list
len(l)  # 3

# a dictionary: maps keys to values
d = {'a' : 1, 'b' : 2}
# access the dictionary elements:
d['a']  # 1
d['b']  # 2
d.keys()  # ['a', 'b']
```

Functions:

```
# we can define functions like this:
def my_func():
      print 'You called my_func!'

# and call them like this
my_func()

# a function can take parameters
def math_func(x, y):
      result = x + y - (x * y)
      return result  # and return results

# the variable answer now has value -47
answer = math_func(7, 9)
```

Objects:

```
# we define objects like:
class Cat:
      # all methods in a class take
      # 'self' as their first argument!
      def purr(self):
            print 'purrrrrrrr!'

      def meow(self, times):
            for i in range(times):
                  print 'meow!'

# and use objects like:
mr_tibbles = Cat()
mr_tibbles.purr()  # prints purrrrrrrr!
mr_tibbles.meow(3)  # prints meow! 3x
```

Some Example Problems (for funsies!)

1. Write a function called `stars` that produces the output:

```
*****
```

2. Write a function called `rocket` that produces the output:

```
 ^
| |
|_|
/ \
 *
 *
 *
 *
 *
```

3. Add an argument, `height`, to your `rocket` function that determines how high your rocket goes!

```
rocket(3)
   ^
  | |
  |_|
  / \
   *
   *
   *
```

4. Write a function `snack` that takes a string argument `kind`. If `kind` is `'sweet'`, you should print the name of a sweet treat, if it is `'salty'` you should print the name of a salty treat, and if it is anything else you should print `'I don't understand!'`.

5. Write a function `secret_identity` that takes two arguments, a string `name` and a dictionary `secret_key`, which maps characters to characters. `secret_identity` should replace all the letters in `name` that are keys in `secret_key` with the value that they map to and print the final name. You might find the function `string.replace(old, new)` useful. For example, `'lollipop'.replace('l', 'r')` will return `'rorripop'`. Some examples:

   `secret_identity('Geraldina', {'e' : 'o', 'd' : 'f'})` should print `'Goralfina'`
   `secret_identity('Billy Bob', {'B' : 'P'})` should print `'Pilly Pob'` (note that case matters!)
   `secret_identity('Totoro', {'T' : 'F', 't' : 'l', 'o' : 'a'})` should print `'Falara'`

6. Write a function `categorize` that takes a list of ints, `numbers`, as an arguments, and, for each element of the list prints the number and whether it is even or odd. Some examples:
   `categorize([1, 2, 3])` should print:
   ```
   1: odd
   2: even
   3: odd
   ```
   `categorize(['7', '-10', '4000', '5'])` should print:
   ```
   7: odd
   -1: even
   4000: even
   5: odd
   ```

7. Write a function `wheres_waldo` that takes as input a string, `text`, and returns the index location of the string `'waldo'` in that text. If `'waldo'` doesn't occur, your function should return -1. If `'waldo'` occurs more than once, return the index of the **last** occurrence. This function should be case insensitive. You may find the functions `string.find(substring)` and/or `string.rfind(substring)` useful. Both these functions can take additional arguments of an index to start looking at `string.find(substring, start)` and end at `string.find(substring, start, end)`.

Some Example Solutions (for looking at after doing them!)

1.
```
def stars():
      print ‘*****’
```

2.
```
def rocket():
      print ‘ ^’
      print ‘| |’
      print ‘|_|’
      print ‘/ \’
      for i in range(5):
            print ‘ *’
```
3.
```
def rocket(height):
      print ‘ ^’
      print ‘| |’
      print ‘|_|’
      print ‘/ \’
      for i in range(height):
            print ‘ *’
```
4.
```
def snack(kind):
      if kind == ‘sweet’:  # this is case sensitive. How do you make it not?
            print ‘donuts!’
      elif kind == ‘salty’:
            print ‘extra salty peanuts with extra salt’
      else:
            print ‘I don\’t understand!’  # escape the ‘
```

5.
```
def secret_identity(name, secret_key):
      secret_name = name
      for key in secret_key: # equivalent to ‘for key in secret_key.keys()’
            secret_name = secret_name.replace(key, secret_key[key])
      print secret_name
```

6.
```
def categorize(numbers):
      for n in numbers:
            if n % 2 == 0:
                  # equivalent with formatted strings: print ‘%i: even’ % n
                  print str(n) + ‘: even’
            else:
                  print str(n) + ‘: odd’
```
7.
```
# solution only using find()
def wheres_waldo(text):
      tlower = text.lower()
      location = tlower.find(‘waldo’)
      while location >= 0:
            next = tlower.find(‘waldo’, location + 1)
            if next == -1:
                  return location
            location = next
      return location

# solution with rfind()
def wheres_waldo(text):
      return text.lower().rfind(‘waldo’)
```