CS101. A25 - Inventory Management

In this exercise, you will be managing an inventory system using Dictionaries in Python. The inventory should be organized by the item name and it should keep track of the number of items available.

You will have to handle adding items to an inventory. Each time an item appears in a given list, increase the item's quantity by 1 in the inventory. Then, you will have to handle deleting items from an inventory.

To finish, you will have to implement a function which returns all the key-value pairs in an inventory as a list of tuples.

1. Create an inventory based on a list

Implement the create_inventory() function that creates an "inventory" from a list of items. It should return a dict containing each item name paired with their respective quantity.

```
>>> create_inventory(["coal", "wood", "wood", "diamond", "diamond",
"diamond"])

{"coal":1, "wood":2 "diamond":3}
```

2. Add items from a list to an existing dictionary

Implement the add_items() function that adds a list of items to an inventory:

```
>>> add_items({"coal":1}, ["wood", "iron", "coal", "wood"])

{"coal":2, "wood":2, "iron":1}
```

3. Decrement items from the inventory

Implement the decrement_items(<items>) function that takes a list of items. The function should remove one from the available count in the inventory for each time an item appears on the list:

```
>>> decrement_items({"coal":3, "diamond":1, "iron":5}, ["diamond", "coal",
"iron", "iron"])

{"coal":2, "diamond":0, "iron":3}
```

Item counts in the inventory should not fall below 0. If the number of times an item appears on the list exceeds the count available, the quantity listed for that item should remain at 0 and additional requests for removing counts should be ignored.

```
>>> decrement_items({"coal":2, "wood":1, "diamond":2}, ["coal", "coal",
"wood", "wood", "diamond"])

{"coal":0, "wood":0, "diamond":1}
```

4. Remove an item entirely from the inventory

Implement the remove_item(<inventory>, <item>) function that removes an item and its count entirely from an inventory:

```
>>> remove_item({"coal":2, "wood":1, "diamond":2}, "coal")
```

```
{"wood":1, "diamond":2}
```

If the item is not found in the inventory, the function should return the original inventory unchanged.

```
>>> remove_item({"coal":2, "wood":1, "diamond":2}, "gold")

{"coal":2, "wood":1, "diamond":2}
```

5. Return the inventory content

Implement the list_inventory() function that takes an inventory and returns a list of (item, quantity) tuples. The list should only include the available items (with a quantity greater than zero):

```
>>> list_inventory({"coal":7, "wood":11, "diamond":2, "iron":7, "silver":0})

[('coal', 7), ('diamond', 2), ('iron', 7), ('wood', 11)]
```

Code Template (dicts.py)

```
"""
Inventory management
using dictionaries
"""

def create_inventory(items):
      """

      :param items: list - list of items to create an inventory from.
      :return:  dict - the inventory dictionary.
      """
      pass

def add_items(inventory, items):
      """

      :param inventory: dict - dictionary of existing inventory.
      :param items: list - list of items to update the inventory with.
      :return:  dict - the inventory dictionary update with the new items.
      """
      pass

def decrement_items(inventory, items):
      """

      :param inventory: dict - inventory dictionary.
      :param items: list - list of items to decrement from the inventory.
      :return:  dict - updated inventory dictionary with items decremented.
      """
      pass

def remove_item(inventory, item):
      """
      :param inventory: dict - inventory dictionary.
      :param item: str - item to remove from the inventory.
      :return:  dict - updated inventory dictionary with item removed.
      """
      pass
```

```
def list_inventory(inventory):
      """

      :param inventory: dict - an inventory dictionary.
      :return: list of tuples - list of key, value pairs from the inventory
dictionary.
      """
      pass
```

Sample output

```
Shown in the description.
```

Test Cases

Function	Inputs	Output	
create_inventory	["wood", "iron", "iron", "diamond", "diamond"]	{"wood": 1, "iron": 2, "diamond": 2}	
create_inventory	["coal", "wood", "wood", "diamond", "diamond", "diamond"])	{"coal":1, "wood":2 "diamond":3}	
add_items	{ }, ["iron", "iron", "diamond"]	{"iron": 2, "diamond": 1}	
add_items	{"coal":1}, ["wood", "iron", "coal", "wood"]	{"coal":2, "wood":2, "iron":1}	
add_items	{"wood": 2, "gold": 1, "diamond": 3}, ["wood", "gold", "gold"]	{"wood": 3, "gold": 3, "diamond": 3}	
add_items	{"iron": 1, "diamond": 2}, ["iron", "wood", "wood"]	{"iron": 2, "diamond": 2, "wood": 2}	
add_items	{"wood": 4, "iron": 2}, ["iron", "iron"]	{"wood": 4, "iron": 4}	
decrement_items	{"coal":3, "diamond":1, "iron":5}, ["diamond", "coal", "iron", "iron"]	{"coal":2, "diamond":0, "iron":3}	
decrement_items	{"coal":2, "wood":1, "diamond":2}, ["coal", "coal", "wood", "wood", "diamond"]	{"coal":0, "wood":0, "diamond":1}	
decrement_items	{"iron": 3, "diamond": 4, "gold": 2}, ["iron", "iron", "diamond", "gold", "gold"]	{"iron": 1, "diamond": 3, "gold": 0}	
decrement_items	{"wood": 2, "iron": 3, "diamond": 1}, ["wood", "wood", "wood", "iron", "diamond", "diamond"]	{"wood": 0, "iron": 2, "diamond": 0}	
remove_item	{"coal":2, "wood":1, "diamond":2}, "coal"	{"wood":1, "diamond":2}	
remove_item	{"coal":2, "wood":1, "diamond":2}, "gold"	{"coal":2, "wood":1, "diamond":2}	
remove_item	{"iron": 1, "diamond": 2, "gold": 1}, "diamond"	{"iron": 1, "gold": 1}	
remove_item	{"iron": 1, "diamond": 2, "gold": 1}, "wood"	{"iron": 1, "gold": 1, "diamond": 2}	
list_inventory	{"coal":7, "wood":11, "diamond":2, "iron":7, "silver":0}	[('coal', 7), ('diamond', 2), ('iron', 7), ('wood', 11)]	

list_inventory	{"coal": 15, "diamond": 3, "wood": 67, "silver": 0}	[("coal", 15), ("diamond", 3), ("wood", 67)]	

****

CS101. A26 - Wordy

Ref: https://github.com/rchatley/extreme_startup

Wordy requires you to parse (read a statement, understand the meaning based on the keywords) and evaluate simple math word problems returning the answer as an integer.

Iteration 0 — Numbers

Problems with no operations simply evaluate to the number given.

```
What is 5?
5
```

Iteration 1 — Addition

Add two numbers together. Handle large numbers and negative numbers.

```
What is 5 plus 13?
18
```

Iteration 2 — Subtraction, Multiplication and Division

Now, perform the following three operations.

```
What is 7 minus 5?
2
What is 6 multiplied by 4?
24
What is 25 divided by 5?
5
```

Iteration 3 — Multiple Operations

Handle a set of operations, in sequence. Since these are verbal word problems, evaluate the expression from left-to-right, *ignoring the typical order of operations.*

```
What is 5 plus 13 plus 6?
24
What is 3 plus 2 multiplied by 3?
```
`15` (i.e. not 9)

Iteration 4 — Handle Exponentials

Handle exponentials.

```
What is 2 raised to the 5th power?
32
```

Iteration 5 - Errors

The parser should reject:

- Unsupported operations ("What is 52 cubed?")

- Non-math questions ("When is 5?")
- Word problems with invalid syntax ("What is 1 plus plus 2?")

# if the question contains an unknown operation.

```
raise ValueError("unknown operation")
```

# if the question is malformed or invalid.

```
raise ValueError("syntax error")
```

Tasks to implement:

Implement the function answer(question).

Input is a question (a string datatype). Output of the function will be an integer or ValueError.

Code Template (wordy.py)

```
"""
Parse Math Q and give answers
"""

def answer(question):
    pass
```

Sample output

```
print(answer(“What is 5?”))
5
```

Test Cases

Function	Inputs	Output	
answer	"What is 5?"	5	
answer	"What is 1 plus 1?"	2	
answer	"What is 53 plus 2?"	55	
answer	"What is -1 plus -10?"	-11	
answer	"What is 123 plus 45678?"	45801	
answer	"What is 4 minus -12?"	16	
answer	"What is -3 multiplied by 25?"	-75	
answer	"What is 33 divided by -3?"	-11	
answer	"What is 1 plus 1 plus 1?"	3	
answer	"What is 1 plus 5 minus -2?"	8	
answer	“What is 20 minus 4 minus 13?"	3	
answer	"What is 17 minus 6 plus 3?"	14	

`answer`	"What is 2 multiplied by -2 multiplied by 3?"	-12	
`answer`	"What is -3 plus 7 multiplied by -2?"	-8	
`answer`	"What is -12 divided by 2 divided by -3?"	2	
`answer`	"When is 5?"	raise ValueError("unknown operation")	
`answer`	"What is 1 plus?"	raise ValueError("syntax error")	
`answer`	"What is?"	raise ValueError("syntax error")	
`answer`	"What is 1 plus plus 2?"	raise ValueError("syntax error")	
`answer`	"What is 1 plus 2 1?"	raise ValueError("syntax error")	
`answer`	"What is 1 2 plus?"	raise ValueError("syntax error")	
`answer`	"What is plus 1 2?"	raise ValueError("syntax error")	
`answer`	"What is 2 2 minus 3?"	raise ValueError("syntax error")	
`answer`	"What is 7 plus multiplied by -2?"	raise ValueError("syntax error")	

****

CS101. A27 - Saddle Points

Ref: J Dalbey's Programming Practice problems

Detect saddle points in a matrix. Example matrix:

$$\begin{array}{c|cccl} & 1 & 2 & 3 & \\ \hline 1 & 9 & 8 & 7 & \\ 2 & 5 & 3 & 2 & \text{<--- saddle point at column 1, row 2, with value 5} \\ 3 & 6 & 6 & 7 & \end{array}$$

It's called a "saddle point" because it is greater than or equal to every element in its row and less than or equal to every element in its column.

A matrix may have zero or more saddle points.

Your code should be able to provide the (possibly empty) list of all the saddle points for any given matrix.

The matrix can have a different number of rows and columns (Non square).

Your implementation assumes a regular matrix. If an Irregular matrix (https://en.wikipedia.org/wiki/Irregular_matrix) is supplied, raise a ValueError

Tasks to implement:

Implement the function saddle_points(matrix).

Input is a matrix (list of rows). Each row is a list too. Eg. `[[9, 8, 7], [5, 3, 2], [6, 6, 7]]`

Return value: List of saddle points. Each saddle point is stored in a dictionary. The output for the above input will be `[{"row": 2, "column": 1}]`. Multiple saddle points can be in any order.

Code Template (saddle.py)

```
'''
Saddle Points
'''

def saddle_points(matrix):
     pass
```

Sample output

```
print(saddle_points([[9, 8, 7], [5, 3, 2], [6, 6, 7]]))
[{'row': 2, 'column': 1}]
```

Test Cases

Function	**Inputs**	**Output**	
`saddle_points`	[[9, 8, 7], [5, 3, 2], [6, 6, 7]]	[{"row": 2, "column": 1}]	
`saddle_points`	[[1, 2, 3], [3, 1, 2], [2, 3, 1]]	[ ]	

`saddle_points`	[[4, 5, 4], [3, 5, 5], [1, 5, 4]]	[{"row": 1, "column": 2}, {"row": 2, "column": 2}, {"row": 3, "column": 2},]	
`saddle_points`	[[6, 7, 8], [5, 5, 5], [7, 5, 6]]	[{"row": 2, "column": 1}, {"row": 2, "column": 2}, {"row": 2, "column": 3},]	
`saddle_points`	[[8, 7, 9], [6, 7, 6], [3, 2, 5]]	[{"row": 3, "column": 3}]	
`saddle_points`	[[3, 1, 3], [3, 2, 4]]	[{"row": 1, "column": 3}, {"row": 1, "column": 1}]	
`saddle_points`	[[2], [1], [4], [1]]	[{"row": 2, "column": 1}, {"row": 4, "column": 1}]	
`saddle_points`	[[2, 5, 3, 5]]	[{"row": 1, "column": 2}, {"row": 1, "column": 4}]	
`saddle_points`	[[3, 2, 1], [0, 1], [2, 1, 0]]	raise ValueError("irregular matrix")	

****

CS101. A28 - grep

Ref: Conversation with Nate Foster, exercism.org

Search a file for lines matching a regular expression pattern. Return the line number and contents of each matching line.

The Unix grep command can be used to search for lines in one or more files that match a user-provided search query (known as the *pattern*).

The grep command takes three arguments:

1. The pattern used to match lines in a file.
2. Zero or more flags to customize the matching behavior.
3. One or more files in which to search for matching lines.

Your task is to implement the grep function, which should read the contents of the specified files, find the lines that match the specified pattern and then output those lines as a single string. Note that the lines should be output in the order in which they were found, with the first matching line in the first file being output first.

As an example, suppose there is a file named "input.txt" with the following contents:

```
hello
world
hello again
```

If we were to call `grep "hello" input.txt`, the returned string should be:

```
hello
hello again
```

Flags

As said earlier, the grep command should also support the following flags:

- -n Print the line numbers of each matching line.
- -l Print only the names of files that contain at least one matching line.
- -i Match line using a case-insensitive comparison.
- -v Invert the program -- collect all lines that fail to match the pattern.
- -x Only match entire lines, instead of lines that contain a match.

If we run `grep -n "hello" input.txt`, the -n flag will require the matching lines to be prefixed with its line number:

```
1:hello
3:hello again
```

And if we run `grep -i "HELLO" input.txt`, we'll do a case-insensitive match, and the output will be:

```
hello
```

hello again

The grep command should support multiple flags at once.

For example, running `grep -l -v "hello" file1.txt file2.txt` should print the names of files that do not contain the string "hello".

Tasks to implement:

Implement the function `grep(pattern, flags, files)`

Inputs.

- `pattern` is the string to be matched. Eg. `“hello”`
- `flags` are options that modify output. `flags` are strings. Eg. `"-n -i -x"`
- `files`: list of files to search for `pattern`. Eg. `[“iliad.txt”,”paradise-lost.txt”]`

Output: `grep` output as a String. Eg.: `"Of Atreus, Agamemnon, King of men.\n"`. Each line of the output is terminated by a new line character ‘\n’.

Code Template (grep.py)

```
'''
The grep program


      -n Print the line numbers of each matching line.
      -l Print only the names of files that contain at least one matching
line.
      -i Match line using a case-insensitive comparison.
      -v Invert the program -- collect all lines that fail to match the
pattern.
      -x Only match entire lines, instead of lines that contain a match.

'''

def grep(pattern, flags, files):

      pass
```

Input file - iliad.txt

```
Achilles sing, O Goddess! Peleus' son;
His wrath pernicious, who ten thousand woes
Caused to Achaia's host, sent many a soul
Illustrious into Ades premature,
And Heroes gave (so stood the will of Jove)
To dogs and to all ravening fowls a prey,
When fierce dispute had separated once
The noble Chief Achilles from the son
Of Atreus, Agamemnon, King of men.
```

Input file - midsummer-night.txt

```
I do entreat your grace to pardon me.
```

```
I know not by what power I am made bold,
Nor how it may concern my modesty,
In such a presence here to plead my thoughts;
But I beseech your grace that I may know
The worst that may befall me in this case,
If I refuse to wed Demetrius.
```

Input file - paradise-lost.txt

```
Of Mans First Disobedience, and the Fruit
Of that Forbidden Tree, whose mortal tast
Brought Death into the World, and all our woe,
With loss of Eden, till one greater Man
Restore us, and regain the blissful Seat,
Sing Heav'nly Muse, that on the secret top
Of Oreb, or of Sinai, didst inspire
That Shepherd, who first taught the chosen Seed
```

Sample output

```
grep("Agamemnon", "", ["iliad.txt"])
"Of Atreus, Agamemnon, King of men.\n"
```

Test Cases for the function `grep`

	Inputs	Output
	`"Agamemnon", "", ["iliad.txt"]`	`"Of Atreus, Agamemnon, King of men.\n"`
	`"Forbidden", "-n", ["paradise-lost.txt"]`	`"2:Of that Forbidden Tree, whose mortal tast\n"`
	`"FORBIDDEN", "-i", ["paradise-lost.txt"]`	`"Of that Forbidden Tree, whose mortal tast\n"`
	`"Forbidden", "-l", ["paradise-lost.txt"]`	`"paradise-lost.txt\n"`
	`"With loss of Eden, till one greater Man", "-x", ["paradise-lost.txt"]`	`"With loss of Eden, till one greater Man\n"`
	`"OF ATREUS, Agamemnon, KIng of MEN.", "-n -i -x", ["iliad.txt"]`	`"9:Of Atreus, Agamemnon, King of men.\n"`

	`"may", "", ["midsummer-night.txt"]`	`"Nor how it may concern my modesty,\n"`
		`"But I beseech your grace that I may know\n"`
		`"The worst that may befall me in this case,\n"`

	`"may", "-n", ["midsummer-night.txt"]`	`"3:Nor how it may concern my modesty,\n"`
		`"5:But I beseech your grace that I may know\n"`
		`"6:The worst that may befall me in this case,\n"`

	`"may", "-x", ["midsummer-night.txt"]`	“”

	`"ACHILLES", "-i", ["iliad.txt"]`	`"Achilles sing, O Goddess! Peleus' son;\n"`
		`"The noble Chief Achilles from the son\n"`

	"Of", "-v", ["paradise-lost.txt"]	"Brought Death into the World, and all our woe,\n" "With loss of Eden, till one greater Man\n" "Restore us, and regain the blissful Seat,\n" "Sing Heav'nly Muse, that on the secret top\n" "That Shepherd, who first taught the chosen Seed\n"
	"Gandalf", "-n -l -x -i", ["iliad.txt"]	""
	"Illustrious into Ades premature,", "-x -v", ["iliad.txt"]	"Achilles sing, O Goddess! Peleus' son;\n" "His wrath pernicious, who ten thousand woes\n" "Caused to Achaia's host, sent many a soul\n" "And Heroes gave (so stood the will of Jove)\n" "To dogs and to all ravening fowls a prey,\n" "When fierce dispute had separated once\n" "The noble Chief Achilles from the son\n" "Of Atreus, Agamemnon, King of men.\n"
	"Agamemnon", "", ["iliad.txt", "midsummer-night.txt", "paradise-lost.txt"]	"iliad.txt:Of Atreus, Agamemnon, King of men.\n"
	"may", "", ["iliad.txt", "midsummer-night.txt", "paradise-lost.txt"])	"midsummer-night.txt:Nor how it may concern my modesty,\n" "midsummer-night.txt:But I beseech your grace that I may know\n" "midsummer-night.txt:The worst that may befall me in this case,\n"
	"that", "-n", ["iliad.txt", "midsummer-night.txt", "paradise-lost.txt"]	"midsummer-night.txt:5:But I beseech your grace that I may know\n" "midsummer-night.txt:6:The worst that may befall me in this case,\n" "paradise-lost.txt:2:Of that Forbidden Tree, whose mortal tast\n" "paradise-lost.txt:6:Sing Heav'nly Muse, that on the secret top\n"
	"who", "-l", ["iliad.txt", "midsummer-night.txt", "paradise-lost.txt"]	"iliad.txt\n" "paradise-lost.txt\n"
	"TO", "-i", ["iliad.txt", "midsummer-night.txt", "paradise-lost.txt"]	"iliad.txt:Caused to Achaia's host, sent many a soul\n" "iliad.txt:Illustrious into Ades premature,\n" "iliad.txt:And Heroes gave (so stood the will of Jove)\n" "iliad.txt:To dogs and to all ravening fowls a prey,\n" "midsummer-night.txt:I do entreat your grace to pardon me.\n" "midsummer-night.txt:In such a presence here to plead my thoughts;\n" "midsummer-night.txt:If I refuse to wed Demetrius.\n" "paradise-lost.txt:Brought Death into the World, and all our woe,\n" "paradise-lost.txt:Restore us, and regain the blissful Seat,\n" "paradise-lost.txt:Sing Heav'nly Muse, that on the secret top\n"

	"a", "-v", ["iliad.txt", "midsummer-night.txt", "paradise-lost.txt"]	"iliad.txt:Achilles sing, O Goddess! Peleus' son;\n" "iliad.txt:The noble Chief Achilles from the son\n" "midsummer-night.txt:If I refuse to wed Demetrius.\n"
	"But I beseech your grace that I may know", "-x", ["iliad.txt", "midsummer-night.txt", "paradise-lost.txt"]	"midsummer-night.txt:But I beseech your grace that I may know\n"
	"WITH LOSS OF EDEN, TILL ONE GREATER MAN", "-n -i -x", ["iliad.txt", "midsummer-night.txt", "paradise-lost.txt"]	"paradise-lost.txt:4:With loss of Eden, till one greater Man\n"
	"Frodo", "-n -l -x -i", ["iliad.txt", "midsummer-night.txt", "paradise-lost.txt"]	“”
	"who", "-n -l", ["iliad.txt", "midsummer-night.txt", “paradise-lost.txt"]	"iliad.txt\n" "paradise-lost.txt\n"
	"Illustrious into Ades premature,", "-x -v", ["iliad.txt", "midsummer-night.txt", "paradise-lost.txt"]	"iliad.txt:Achilles sing, O Goddess! Peleus' son;\n" "iliad.txt:His wrath pernicious, who ten thousand woes\n" "iliad.txt:Caused to Achaia's host, sent many a soul\n" "iliad.txt:And Heroes gave (so stood the will of Jove)\n" "iliad.txt:To dogs and to all ravening fowls a prey,\n" "iliad.txt:When fierce dispute had separated once\n" "iliad.txt:The noble Chief Achilles from the son\n" "iliad.txt:Of Atreus, Agamemnon, King of men.\n" "midsummer-night.txt:I do entreat your grace to pardon me.\n" "midsummer-night.txt:I know not by what power I am made bold,\n" "midsummer-night.txt:Nor how it may concern my modesty,\n" "midsummer-night.txt:In such a presence here to plead my thoughts;\n" "midsummer-night.txt:But I beseech your grace that I may know\n" "midsummer-night.txt:The worst that may befall me in this case,\n" "midsummer-night.txt:If I refuse to wed Demetrius.\n" "paradise-lost.txt:Of Mans First Disobedience, and the Fruit\n" "paradise-lost.txt:Of that Forbidden Tree, whose mortal tast\n" "paradise-lost.txt:Brought Death into the World, and all our woe,\n" "paradise-lost.txt:With loss of Eden, till one greater Man\n"

		"paradise-lost.txt:Restore us, and regain the blissful Seat,\n" "paradise-lost.txt:Sing Heav'nly Muse, that on the secret top\n" "paradise-lost.txt:Of Oreb, or of Sinai, didst inspire\n" "paradise-lost.txt:That Shepherd, who first taught the chosen Seed\n"

****

CS101. A29 - Robot Simulator

Write a robot simulator.

A robot factory's test facility needs a program to verify robot movements.

The robots have three possible movements:

- turn right
- turn left
- advance

Robots are placed on a hypothetical infinite grid, facing a particular direction (north, east, south, or west) at a set of {x,y} coordinates, e.g., {3,8}, with coordinates increasing to the north and east.

The robot then receives a number of instructions, at which point the testing facility verifies the robot's new position, and in which direction it is pointing.

- The letter-string "RAALAL" means:
  - Turn right
  - Advance twice
  - Turn left
  - Advance once
  - Turn left yet again
- Say a robot starts at {7, 3} facing north. Then running this stream of instructions should leave it at {9, 4} facing west.

Tasks to complete

In the Robot Class, complete the __init__ and the move methods.

Initialize the position of the Robot in __init__ . Default is to place the Robot at the origin (0,0) facing North. The position of the Robot is present in the attribute coordinates (which is a tuple) containing (x_pos, y_pos).

Implement the move method. The movement is given by the route input parameter (which is a string). The move method updates the coordinates and the direction attributes.

You are free to use the global constants: EAST, NORTH, WEST, SOUTH as desired.

Code Template (robot_simulator.py)

```
# Globals for the directions
# Change the values as you see fit
EAST = 1
NORTH = 2
WEST = 3
SOUTH = 4

class Robot:

     coordinates=()
     direction=NORTH

     def __init__(self, direction=NORTH, x_pos=0, y_pos=0):
        pass
     def move(self, route):
       pass
```

Sample output

```
Shown in description
```

Test Cases

	Inputs	method/attributes	

	Robot(NORTH, 0, 0)	coordinates=(0,0)
		direction=NORTH	

	Robot(SOUTH, -1, -1)	coordinates=(-1, -1)
		direction=SOUTH	

	robot = Robot(NORTH, 0, 0)
	move(“R”)
		coordinates=(0,0)
		direction=EAST	

	robot = Robot(EAST, 0, 0)
	robot.move("R")
		coordinates=(0,0)
		direction=SOUTH	

	robot = Robot(SOUTH, 0, 0)
	robot.move("R")
		coordinates=(0,0)
		direction=WEST	

	robot = Robot(WEST, 0, 0)
	robot.move("R")
		coordinates=(0,0)
		direction=NORTH	

	robot = Robot(NORTH, 0, 0)
	robot.move("L")
		coordinates=(0,0)
		direction=WEST	

	robot = Robot(WEST, 0, 0)
	robot.move("L")
		coordinates=(0,0)
		direction=SOUTH	

	robot = Robot(SOUTH, 0, 0)
	robot.move("L")
		coordinates=(0,0)
		direction=EAST	

	robot = Robot(EAST, 0, 0)
	robot.move("L")
		coordinates=(0,0)
		direction=NORTH	

	robot = Robot(NORTH, 0, 0)
	robot.move("A")
		coordinates=(0, 1)
		direction=NORTH	

	robot = Robot(SOUTH, 0, 0)
	robot.move("A")
		coordinates=(0, -1)
		direction=SOUTH	

	robot = Robot(EAST, 0, 0)
	robot.move("A")
		coordinates=(1, 0)
		direction=EAST	

	robot = Robot(WEST, 0, 0)
	robot.move("A")
		coordinates=(-1, 0)
		direction=WEST	

	robot = Robot(NORTH, 7, 3)
	robot.move("RAALAL")
		coordinates=(9, 4)
		direction=WEST	

	robot = Robot(NORTH, 0, 0)
	robot.move("LAAARALA")
		coordinates=(-4, 1)
		direction=WEST	

	robot = Robot(EAST, 2, -7)	coordinates=(-3, -8)	

	robot.move("RRAAAAALA")	direction=SOUTH	

	robot = Robot(SOUTH, 8, 4)
	robot.move("LAAARRRALLLL")
		coordinates=(11, 5)
		direction=NORTH	

****

CS101. A30 - Minesweeper

Add the mine counts to a completed Minesweeper board.

Minesweeper is a popular game where the user has to find the mines using numeric hints that indicate how many mines are directly adjacent (horizontally, vertically, diagonally) to a square.

In this exercise you have to create some code that counts the number of mines adjacent to a given empty square and replaces that square with the count.

The board is a rectangle composed of blank space (' ') characters. A mine is represented by an asterisk ('*') character.

If a given space has no adjacent mines at all, leave that square blank.

For example you may receive a 5 x 4 board like this (empty spaces are represented here with the '·' character):

```
.*.*.
..*..
..*..
.....
```

The above mine board will be input to the function as a list of strings: [“ * * ”,” * ”,” * ”,”   ”]

Your code will transform the input into this:

```
1*3*1
13*31
·2*2·
·111·
```

The output of the function (return value) is a list of strings: [“1*3*1”, “13*31”, “ 2*2 ”, “ 111 ”]

Tasks to complete

Implement the `annotate(minefield)` function.

Input to the function is a list of Strings. Each string corresponds to a row of mines. ‘ ‘ - Space is no mine, ‘*’ - asterisk is a Mine. Eg. [" * ", " * ", "*****", " * ", " * "] is 5 rows.

Return value is also a list of strings. Eg. [" 2*2 ", "25*52", "*****", "25*52", " 2*2 "]

Raise a ValueError in case malformed board is input. Eg. [" ", "* ", " "] is a malformed board - unequal columns in rows.

Code Template (minefield.py)

```
def annotate(minefield):
     pass
```

Test Cases

	Inputs	Return value	
	[ ]	[ ]	
	[""]	[""]	
	[" ", " ", " "] ([“space space space”, “space space space”, “space space space”])	[" ", " ", " "] ([“space space space”, “space space space”, “space space space”])	
	["***", "***", "***"] ([“star star star”, “star star star”, “star star star”])	["***", "***", "***"] ([“star star star”, “star star star”, “star star star”])	
	[" ", " * ", " "]	["111", "1*1", "111"]	
	["***", "* *", "***"]	["***", "*8*", "***"]	
	[" * * "] ([“space star space star space”])	["1*2*1"]	
	["* *"] ([“star space space space star”])	["*1 1*"] ([“star 1 space 1 star”])	
	[" ", "*", " ", "*", " "] ([“space”, “star”, “space”, “star”, “space”])	["1", "*", "2", "*", "1"]	
	["*", " ", " ", " ", "*"] ([“star”, “space”, “space”, “space”, “star”])	["*", "1", " ", "1", "*"] ([“star”, “1”, “space”, “1”, “star”])	
	[" * ", " * ", "*****", " * ", " * "] ([“space space star space space”, “space space star space space”, “star star star star star”, “space space star space space”, “space space star space space”])	[" 2*2 ", "25*52", "*****", "25*52", " 2*2 "]	
	[" * * ", " * ", " * ", " * *", " * * ", " "] ([“space star space space star space”, “space space star space space space”, “space space space space star space”, “space space space star space star”, “space star space space star space”, “space space space space space”])	['1*11*1', '111222', ' 113*2', '12*4*3', '1*23*2', '111111']	
	[" ", " * ", " ", " ", " * "] ([“space space space space space”, “space space space star space”, “space space space space space”, “space space space space space”, “space star space space space”])	[" 111", " 1*1", " 111", "111 ", "1*1 "] ([“space space 1 1 1”, “space space 1 star 1”, “space space 1 1 1”, “1 1 1 space space”, “1 star 1 space space”])	
	[" ", "* ", " "] ([“space”, “star space space”, “space space space space space”, “space space”])	raise ValueError("The board is invalid with current input.")	
	["X * "] ([“X space space star space”])	raise ValueError("The board is invalid with current input.")	

****

CS101. A31 - Say

Ref: http://www.javaranch.com/say.jsp , exercism.org.

Given a number from 0 to 999,999,999,999, spell out that number in English.

Step 1

Handle the basic case of 0 through 99. If the input to the program is 22, then the output should be 'twenty-two'. Your program should complain if given a number outside the range.

Some good test cases for this program are:

- 0
- 14
- 50
- 98
- -1
- 100

Step 2

Implement breaking a number up into chunks of thousands. So 1234567890 should yield a list like 1, 234, 567, and 890, while the far simpler 1000 should yield just 1 and 0. The program must also report any values that are out of range.

Step 3

Now handle inserting the appropriate scale word between those chunks. So 1234567890 should yield '1 billion 234 million 567 thousand 890'

The program must also report any values that are out of range. Stop at "trillion".

Step 4

Put it all together to get nothing but plain English. 12345 should give twelve thousand three hundred forty-five. The program must also report any values that are out of range.

Use *and* (correctly) when spelling out the number in English:

- 14 becomes "fourteen".
- 100 becomes "one hundred".
- 120 becomes "one hundred and twenty".
- 1002 becomes "one thousand and two".
- 1323 becomes "one thousand three hundred and twenty-three".

Error messages

# if the number is negative raise ValueError("input out of range")	# if the number is larger than 999,999,999,99 raise ValueError("input out of range")

Tasks to complete

Implement the say(number) function. Input is an integer. Return value is a string.

Code Template (say.py)

```
'''
convert digits to numbers
'''

def say(number):
    pass
```

Sample output

```
Shown in description
```

Test Cases

	Inputs		
	0	“zero”	
	1	“one”	
	14	“fourteen”	
	20	“twenty”	
	22	“twenty-two”	
	100	“one hundred”	
	123	“one hundred twenty-three”	
	170	“one hundred seventy”	
	1000	“one thousand”	
	1234	“one thousand two hundred thirty-four”	
	1000000	“one million”	
	1002345	“one million two thousand three hundred forty-five”	
	1000000000	"one billion"	
	987654321123	"nine hundred eighty-seven billion six hundred fifty-four million three hundred twenty-one thousand one hundred twenty-three"	
	-1	ValueError, "input out of range"	
	1000000000000	ValueError, "input out of range"	

****

CS101. A32 - Scrabble Score

Given a word, compute the Scrabble score for that word.

Letter Values

Letter	Value
A,E,I,O,U,L,N,R,S,T	1
D,G	2
B,C,M,P	3
F,H,V,W,Y	4
K	5
J,X	8
Q,Z	10

Example

"cabbage" should be scored as worth 14 points:

- 3 points for C
- 1 point for A, twice
- 3 points for B, twice
- 2 points for G
- 1 point for E

And to total:

- 3 + 2*1 + 2*3 + 2 + 1
- = 3 + 2 + 6 + 3
- = 5 + 9
- = 14

Tasks to complete

Implement the score(word) function. Input is a string. Return value is an integer.

Code Template (scrabble_score.py)

```
def score(word):
    pass
```

Sample output

```
Shown in description
```

Test Cases

	Inputs	Return value	
	“a”	1	

	“A”	1	
	“f”	4	
	“at”	2	
	“zoo”	12	
	“street”	6	
	“quirky”	22	
	“OxyphenButazone”	41	
	“pinata”	8	
	“”	0	
	“abcdefghijklmnopqrstuvwxyz”	87	

****

CS101. A33 - Kindergarten Garden

Ref: http://jumpstartlab.com/, exercism.org

Given a diagram, determine which plants each child in the kindergarten class is responsible for.

The kindergarten class is learning about growing plants. The teacher thought it would be a good idea to give them actual seeds, plant them in actual dirt, and grow actual plants.

They've chosen to grow grass, clover, radishes, and violets.

To this end, the children have put little cups along the window sills, and planted one type of plant in each cup, choosing randomly from the available types of seeds.

```
[window][window][window]

........................ # each dot represents a cup

........................
```

There are 12 children in the class:

- Alice, Bob, Charlie, David,
- Eve, Fred, Ginny, Harriet,
- Ileana, Joseph, Kincaid, and Larry.

Each child gets 4 cups, two on each row. Their teacher assigns cups to the children alphabetically by their names.

The following diagram represents Alice's plants:

```
[window][window][window]

VR......................

RG......................
```

In the first row, nearest the windows, she has a violet and a radish. In the second row she has a radish and some grass.

Your program will be given the plants from left-to-right starting with the row nearest the windows. From this, it should be able to determine which plants belong to each student.

For example, if it's told that the garden looks like so:

```
[window][window][window]

VRCGVVRVCGGCCGVRGCVCGCGV

VRCCCGCRRGVCGCRVVCVGCGCV
```

Then if asked for Alice's plants, it should provide:

- Violets, radishes, violets, radishes

While asking for Bob's plants would yield:

- Clover, grass, clover, clover

Tasks to complete

In the Garden Class, complete the `__init__` and the `plants` methods.

Inputs to the constructor: diagram, and list of students in the class. Eg. ['Alice','Bob', 'Charlie', 'David','Eve', 'Fred', 'Ginny', 'Harriet','Ileana', 'Joseph', 'Kincaid','Larry']

The `plants` method takes a student's name as input and returns his/her plant names.

Code Template (kindergarten_garden.py)

```
class Garden:

     def __init__(self, diagram, students=student_list):
    pass

     def plants(self, name):
    pass
```

Sample output

```
Shown in description
```

Test Cases

	Inputs	method/attributes	

	Garden("RC\nGG")
	garden.plants("Alice")
		["Radishes", "Clover", "Grass", "Grass"]	

	Garden("VC\nRC")
	garden.plants("Alice")
		["Violets", "Clover", "Radishes", "Clover"]	

	Garden("VVCG\nVVRC")
	garden.plants("Bob")
		["Clover", "Grass", "Radishes", "Clover"]	

	Garden("VVCCGG\nVVCCGG")
	garden.plants("Bob")
		["Clover", "Clover", "Clover", "Clover"]	

	Garden("VVCCGG\nVVCCGG")
	garden.plants("Charlie")
		["Grass", "Grass", "Grass", "Grass"]	

	Garden("VRCGVVRVCGGCCGVRGCVCGCGV\nVRCC CGCRRGVCGCRVVCVGCGCV")
	garden.plants("Alice")
		["Violets", "Radishes", "Violets", "Radishes"]	

	Garden("VRCGVVRVCGGCCGVRGCVCGCGV\nVRCC CGCRRGVCGCRVVCVGCGCV") (garden.plants("Bob")	["Clover", "Grass", "Clover", "Clover"]	
	Garden("VRCGVVRVCGGCCGVRGCVCGCGV\nVRCC	["Violets", "Violets", "Clover", "Grass"]	

	CGCRRGVCGCRVVCVGCGCV") garden.plants("Charlie")		

	Garden("VRCGVVRVCGGCCGVRGCVCGCGV\nVRCC CGCRRGVCGCRVVCVGCGCV")
	garden.plants("David")
		["Radishes", "Violets", "Clover", "Radishes"]	

	Garden("VRCGVVRVCGGCCGVRGCVCGCGV\nVRCC CGCRRGVCGCRVVCVGCGCV")
	garden.plants("Eve")
		["Clover", "Grass", "Radishes", "Grass"]	

	Garden("VRCGVVRVCGGCCGVRGCVCGCGV\nVRCC CGCRRGVCGCRVVCVGCGCV")
	garden.plants("Fred")
		["Grass", "Clover", "Violets", "Clover"]	

	Garden("VRCGVVRVCGGCCGVRGCVCGCGV\nVRCC CGCRRGVCGCRVVCVGCGCV")
	garden.plants("Ginny")
		["Clover", "Grass", "Grass", "Clover"]	

	Garden("VRCGVVRVCGGCCGVRGCVCGCGV\nVRCC CGCRRGVCGCRVVCVGCGCV")
	garden.plants("Harriet")
		["Violets", "Radishes", "Radishes", "Violets"]	

	Garden("VRCGVVRVCGGCCGVRGCVCGCGV\nVRCC CGCRRGVCGCRVVCVGCGCV")
	garden.plants("Ileana")
		["Grass", "Clover", "Violets", "Clover"]	

	Garden("VRCGVVRVCGGCCGVRGCVCGCGV\nVRCC CGCRRGVCGCRVVCVGCGCV")
	garden.plants("Joseph")
		["Violets", "Clover", "Violets", "Grass"]	

	Garden("VRCGVVRVCGGCCGVRGCVCGCGV\nVRCC CGCRRGVCGCRVVCVGCGCV")
	garden.plants("Kincaid")
		["Grass", "Clover", "Clover", "Grass"]	

	Garden("VRCGVVRVCGGCCGVRGCVCGCGV\nVRCC CGCRRGVCGCRVVCVGCGCV")
	garden.plants("Larry")
		["Grass", "Violets", "Clover", "Violets"]	

	Garden("VCRRGVRG\nRVGCCGCV",
	students=["Samantha", "Patricia", "Xander", "Roger"])
	garden.plants("Patricia")
		["Violets", "Clover", "Radishes", "Violets"]	

	Garden("VCRRGVRG\nRVGCCGCV",
	students=["Samantha", "Pgatricia", "Xander", "Roger"])
	garden.plants("Xander")
		["Radishes", "Grass", "Clover", "Violets"]	

****

CS101. A34 - Alien Game

Instructions

You are creating a game where the player has to fight aliens. You have decided to use Object Oriented Programming (OOP) to take advantage of using classes for this game.

1. Create the Alien Class

Define the Alien class with a constructor that accepts two parameters <x_coordinate> and <y_coordinate>, putting them into x_coordinate and y_coordinate instance variables. Every alien will also start off with a health level of 3, so the health variable should be initialized as well.

```
>>> alien = Alien(2, 0)

>>> alien.x_coordinate

2

>>> alien.y_coordinate

0

>>> alien.health

3
```

Now, each alien should be able to internally track its own position and health.

2. The hit Method

The Alien class has a `hit` method that decrements the health of an alien object by 1 when called. Calling hit on a zero health Alien does not reduce the health to *below* zero.

```
>>> alien = Alien(0, 0)

>>> alien.health

# Initialized health value.

3

# Decrements health by 1 point.

>>> alien.hit()

>>> alien.health

2
```

3. The is_alive Method

You realize that if the health keeps decreasing, at some point it will probably hit 0. It would be a good idea to add an is_alive method that can be called to check if the alien is alive. <alien>.is_alive() should return a boolean.

```
>>> alien.health

1

>>> alien.is_alive()

True

>>> alien.hit()

>>> alien.health

0

>>> alien.is_alive()

False
```

4. The teleport Method

In the game, the aliens have the ability to teleport! You will need to write a teleport method that takes new x_coordinate and y_coordinate values, and changes the alien's coordinates accordingly.

```
>>> alien.teleport(5, -4)

>>> alien.x_coordinate

5

>>> alien.y_coordinate

-4
```

5. The collision_detection Method

Obviously, if the aliens can be hit by something, then they need to be able to detect when such a collision has occurred. However, collision detection algorithms can be tricky, and you do not yet know how to implement one. This will be implemented later, but the collision_detection method has to appear in the class as a reminder to build out the functionality. It will need to take an input variable of some kind (probably another object), but that's really all you know. You will need to make sure that putting the method definition into the class doesn't cause any errors when called:

>>> alien.collision_detection(other_object)

>>>

6. Alien Counter

Keep track of how many aliens have been created over the game's lifetime. Create a counter to count number of Alien objects created. Ideally, this counter would increment whenever someone *made an new Alien*. Class attributes can be changed from an instance method by using the *class name*: `Alien.<class attribute name>`

For example:

```
>>> alien_one = Alien(5, 1)

>>> alien_one.total_aliens_created

1

>>> alien_two = Alien(3, 0)

>>> alien_two.total_aliens_created

2

>>> alien_one.total_aliens_created

2

>>> Alien.total_aliens_created

# Accessing the variable from the class directly

2
```

7. Object Creation

Create a standalone function to create a list of alien objects, given a list of positions (as tuples).

For example:

```
>>> alien_start_positions = [(4, 7), (-1, 0)]

>>> aliens = new_aliens_collection(alien_start_positions)

>>> for alien in aliens:

      print(alien.x_coordinate, alien.y_coordinate)

(4, 7)

(-1, 0)
```

Tasks to complete

In the Alien Class, complete the __init__ and the above mentioned methods.

Code Template (aliens.py)

```
"""Solution to Ellen's Alien Game exercise."""

class Alien:
      """Create an Alien object with location x_coordinate and y_coordinate.

      Attributes
      ----------
      (class)total_aliens_created: int
      x_coordinate: int - Position on the x-axis.
      y_coordinate: int - Position on the y-axis.
      health: int - Amount of health points.
```

```
    Methods
    -------
    hit(): Decrement Alien health by one point.
    is_alive(): Return a boolean for if Alien is alive (if health is > 0).
    teleport(new_x_coordinate, new_y_coordinate): Move Alien object to new
coordinates.
    collision_detection(other): Implementation TBD.
    """
```

Sample output

```
Shown in description
```

Test Cases

	Inputs	method/attributes	

	alien = Alien(2, -1)	alien.x_coordinate = 2
		alien.y_coordinate = -1
		alien.health = 3	

	alien_one = Alien(-8, -1)
	alien_two = Alien(2, 5)
		alien_one.x_coordinate = -8
		alien_one.y_coordinate = -1
		alien_one.health = 3
		alien_two.x_coordinate = 2
		alien_two.y_coordinate = 5
		alien_two.health = 3	

	alien = Alien(0, 0)
	alien.hit() called (1, 2, 3, 4) times
		When called once:
		alien.health = 2
		alien.is_alive = True
		
		When called 2 times:
		alien.health = 1
		alien.is_alive = True
		
		When called 3 times:
		alien.health = 0
		alien.is_alive = False
		
		When called 4 times:
		alien.health = 0
		alien.is_alive = False	

	alien = Alien(0, 0)
	alien.teleport(-1, -4)
		alien.x_coordinate = -1
		alien.y_coordinate = -4
		alien.health = 3	

	alien = Alien(7, 3)
	alien.collision_detection(Alien())
		return value is None	

	alien_one = Alien(0, 2)
	alien_two = Alien(-6, -1)
	Alien.total_aliens_created = -2
		alien_one.total_aliens_created ==
		alien_two.total_aliens_created	

	Alien.total_aliens_created = 0
	aliens = [Alien(-2, 6)]
	aliens.append(Alien(3, 5))
	aliens.append(Alien(-5, -5))
		Alien.total_aliens_created = 3	

	position_data = [(-2, 6), (1, 5), (-4, -3)]
	obj_list = new_aliens_collection(position_data)
		obj_list[0].x_coordinate = -2
		obj_list[0].y_coordinate = 6
		obj_list[1].x_coordinate = 1
		obj_list[1].y_coordinate = 5
		obj_list[2].x_coordinate = -4
		obj_list[2].y_coordinate = -3	

****

CS101. A35 - Binary Search

Implement a binary search algorithm.

Searching a sorted collection is a common task. A dictionary is a sorted list of word definitions. Given a word, one can find its definition. A telephone book is a sorted list of people's names, addresses, and telephone numbers. Knowing someone's name allows one to quickly find their telephone number and address.

If the list to be searched contains more than a few items (a dozen, say) a binary search will require far fewer comparisons than a linear search, but it imposes the requirement that the list be sorted.

In computer science, a binary search or half-interval search algorithm finds the position of a specified input value (the search "key") within an array sorted by key value.

In each step, the algorithm compares the search key value with the key value of the middle element of the array.

If the keys match, then a matching element has been found and its index, or position, is returned.

Otherwise, if the search key is less than the middle element's key, then the algorithm repeats its action on the sub-array to the left of the middle element or, if the search key is greater, on the sub-array to the right.

If the remaining array to be searched is empty, then the key cannot be found in the array and a special "not found" indication is returned.

A binary search halves the number of items to check with each iteration, so locating an item (or determining its absence) takes logarithmic time.. A binary search is a dichotomic divide and conquer search algorithm.

Example 1.

Value to search: 6

1	3	4	6	8	9	11

Length of the list = 7. Mid index = 3.

1	3	4	6	8	9	11

Value found at index = 3.

Example 2.

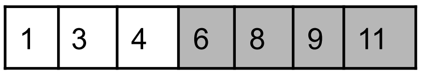
Value to search: 1

1	3	4	6	8	9	11

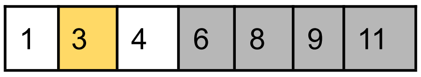
Length of the list = 7. Mid index = 3.

1	3	4	6	8	9	11

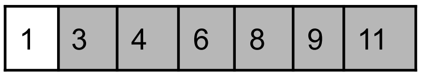
search_list[3] = 6. 1 is lesser than 6. Search in the upper half of the list.



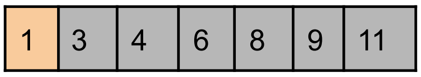
Mid index of the upper half = 1.



search_list[1] = 3. 1 is lesser than 3. Search in the upper half.



Mid index = 0.



search_list[0] == 1. 1 == 1. Value found. **Return 0**.

If the value to be searched was 0. Raise “value not found” ValueError exception

Tasks to complete

Implement BinarySearch algorithm in the find() function. Inputs to the find function: a sorted list of numbers (search_list), and a value to search.

Return: index if found, raise ValueError(“value not in array”) if not found

Code Template (binary_search.py)

```
def find(search_list, value):
  pass
```

Sample output

```
find([1, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 634], 144)
9
find([1, 3, 4, 6, 8, 9, 11], 13)
ValueError: "value not in array"
```

Test Cases

	Inputs	**Return Value**	
	find([6], 6)	0	
	find([1, 3, 4, 6, 8, 9, 11], 6)	3	
	find([1, 3, 4, 6, 8, 9, 11], 1)	0	
	find([1, 3, 4, 6, 8, 9, 11], 11)	6	
	find([1, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 634], 144)	9	
	find([1, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377], 21)	5	

	find([1, 3, 4, 6, 8, 9, 11], 7)	raise ValueError("value not in array")	
	find([1, 3, 4, 6, 8, 9, 11], 0)	raise ValueError("value not in array")	
	find([ ], 1)	raise ValueError("value not in array")	
	find([1, 2], 0)	raise ValueError("value not in array")	

****