```
#list: multiple elements in one variable , can be any type
mylist = [1, 2, 3, 4.5, "ABC", True]
print(len(mylist))
print(mylist[0])
print(mylist[1:3])
print(len(mylist[4])) #LENGTH OF [4]-"ABC"
print(mylist[4][1]) # [4} - "ABC", [1] - B
mylist.append("XYZ") # add "XYZ" into the list
print(mylist)
mylist.append([1,2,3])
mylist = [1,2,3,4.5,"ABC",True,"XYZ", [1,2,3]]
mylist[0]=10
print(mylist)
```

```
6
1
[2, 3]
3
B
[1, 2, 3, 4.5, 'ABC', True, 'XYZ']
[10, 2, 3, 4.5, 'ABC', True, 'XYZ', [1, 2, 3]]
```

mylist.clear()

```
mylist2 = []
for num in range(10):
  mylist2.append(num)
mylist3 = [10, 20 ,30 ,"ABC", [1,2,3]]
for idx in range(len(mylist3)):
__print(mylist3[idx])
for name in mylist3:
__print(name)
for num in mylist3:
    print(num)
```

```
10
20
30
ABC
[1, 2, 3]
10
20
30
ABC
[1, 2, 3]
10
20
30
ABC
[1, 2, 3]
```

```
infile = open("myData.txt","r")
for eachline in infile:
  print(eachline)
infile.close()
outfile = open("output.txt", "w")
  for n in range(10):
    print(n, file=outfile)
    print(n)
outfile.close()
```

```
#create a file in the folder where the python file is created, and key input
in randomly. Open the file “mySample” , basically open the file to do changes
in python.
```

```
infile = open("mySample.txt", "r")
for eachline in infile:
    mylist = eachline.split()
    print(mylist)
    print(len(mylist))

infile.close()
```

mySample.txt

```
['fyu', 'jn']
2
['jhny', 'fu', 'gfhu']
3
['n', 'fy', 'un', 'dx']
4
['hy']
1
['k']
1
['l']
1
['y']
1
['qw']
1
['l,kugf']
1
['8']
1
```

```
'''
Save user's input into list, compare with the actual answer, and print the result
'''
answerKey = ("a", "b", "b", "a", "d", "c", "b", "a", "b", "c")
```

```
inputAnswer = []
for idx in range(len(answerKey)):
    ans = input("Question {} answer? ".format(idx + 1)).lower()
    while ans < "a" or ans > "d":
        ans = input("Invalid!! Question {} answer?".format(idx + 1)).lower()
    inputAnswer.append(ans)

#Compare inputAnswer with answerKey
count = 0
for idx in range(len(answerKey)):
    if answerKey[idx] == inputAnswer[idx]:
        count = count + 1
        print("Q{} {} correct".format(idx + 1, answerKey[idx]))
    else:
        print("Q{} {} incorrect, answer is {}".format(idx + 1, inputAnswer[idx],
answerKey[idx]))

print("Total {} out of {}".format(count, len(answerKey)))
```

```
Question 1 answer? A
Question 2 answer? b
Question 3 answer? c
Question 4 answer? a
Question 5 answer? D
Question 6 answer? c
Question 7 answer? B
Question 8 answer? a
Question 9 answer? 1
Invalid!! Question 9 answer? b
Invalid!! Question 9 answer?b
Question 10 answer? c
Q1 a correct
Q2 b correct
Q3 c incorrect, answer is b
Q4 a correct
Q5 d correct
Q6 c correct
Q7 b correct
Q8 a correct
Q9 b correct
Q10 c correct
Total 9 out of 10
```

```
'''

Summary:
Get lower limit and upper limit
Generate a number in between
Run a loop as long as user does not guess correct
    if too high or too low inform the user accordingly
Print the number of attempts
'''
import random

lower = int(input("Enter lower limit: "))
```

```
upper = int(input("Enter upper limit: "))
number = random.randint(lower, upper)
# Guess the first guess, and start counting the number of attempts
count = 0
guess = int(input("Enter your guess between {} and {}: ".format(lower, upper)))
# Start counting
count = count + 1
while guess != number:
    if guess < lower or guess > upper:
        print("Invalid guess")
    elif guess < number:
        print("Too low")
    else:
        print("Too high")

    # Remember to prompt for the next guess, otherwise infinite loop
    guess = int(input("Enter next guess: "))
    count = count + 1

#After the loop, print the value of count
print("You got it in {} tries".format(count))

# The question says to ensure the user's guess is valid. It's assumed that the
guess
# must be between lower and upper. So inside the loop, we check for 3 things
'''
Points to note:
Line 17: Have to perform one input before while loop (line 20)
Line 29: Must perform input (for subsequent guess) inside the loop
'''

Enter lower limit:  22
Enter upper limit:  55
Enter your guess between 22 and 55: 33
Too low
Enter next guess: 22
Too low
Enter next guess:   66
Invalid guess
Enter next guess:


'''
Summary:
Initialise weight (for calculation)
Initialise letters A to Z (for look up and comparison)
Read NRIC as string
Multiple each digit with corresponding weight in "weights"
Sum all
Perform remainder and then 11 - remainder
Use the number as index to locate a letter
Compare letter with last letter in NRIC
Same --> Valid otherwise --> Invalid

The question does not specify what (list, tuple, string) to use, so i will go with
something simple. It is OK if you something else.
'''
# Set up a string for all letters
reference = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
```

```
weights   = (2, 7, 6, 5, 4, 3, 2)   # Use tuple because the values won't be
changes

# Ask user to enter NRIC, and convert to upper case
nric = input("Enter NRIC: ").upper()    # The digits will not be affected by
calling upper()

# Go through the first 7 characters in nric, convert each to int.
# multiple with corresponding number in weights.
# Sum them along the way

prodSum = 0
for idx in range(len(weights)):
    value       = int(nric[idx])        # convert to integer
    digitWeight = value * weights[idx]  # Multiply a number in the list: weights
    prodSum     = prodSum + digitWeight

# % 11 and 11 - remaining
letterIndex = 11 - (prodSum % 11)       # Follow the instruction in the question
# Use this number to look up a letter
letter = reference[letterIndex - 1]     # Remember string index starts at 0
# Compare 2 letters
if letter == nric[-1]:                  # nric[-1] gives up the last character
    print("Valid NRIC")
else:
    print("Invalid NRIC")
```

```
Enter NRIC: 9227051c
Valid NRIC
```

```
'''
Ask the user to enter branch code and quantity (stock)
Save this as a list. Example: [ "B1", 200 ]
There will be multiple brances. Therefore will have a list of lists:
Example
 [ [ "B1", 200], [ "B2", 400 ], [ "B3", 100 ],  .... ]
 After all the branches are saved into a list (shown above)
 Use another Loop to ask user to enter:
     - Branch code
     - Number (of items to be added to quantity)
Find the branch in the list, and update the quantity accordingly
'''

branches = []   # Start with an empty list
brCode = input("Enter branch code: ")
while brCode != "":     # "" means no more input
    qty       = int(input("Enter qty: "))
    oneBranch = [brCode, qty]                # Put them inside a list
    branches.append(oneBranch)               # Put the list inside another 1
    brCode = input("Enter next branch code")

# Use a loop to allow user to enter branch code and quantity to be
# added to that branch. Update the list accordingly.
# To save time, i read 2 into 1 string and split them

stockData = input("Branch code and qty?")
while stockData != "":
    code, qty = stockData.split()
    # Now search the list to look for a branch code that matches code
    for br in branches:
        if br[0] == code:                # br is a list. br[0] is the branch code
            br[1] = br[1] + int(qty)     # br[1] is the quantity
            break
    stockData = input("Enter next branch code and qty: ")

print(branches)

# This exercise is meant to give you a feel of list inside a list
# Because of that, we have loop inside loop (line 31 and line 28)
```