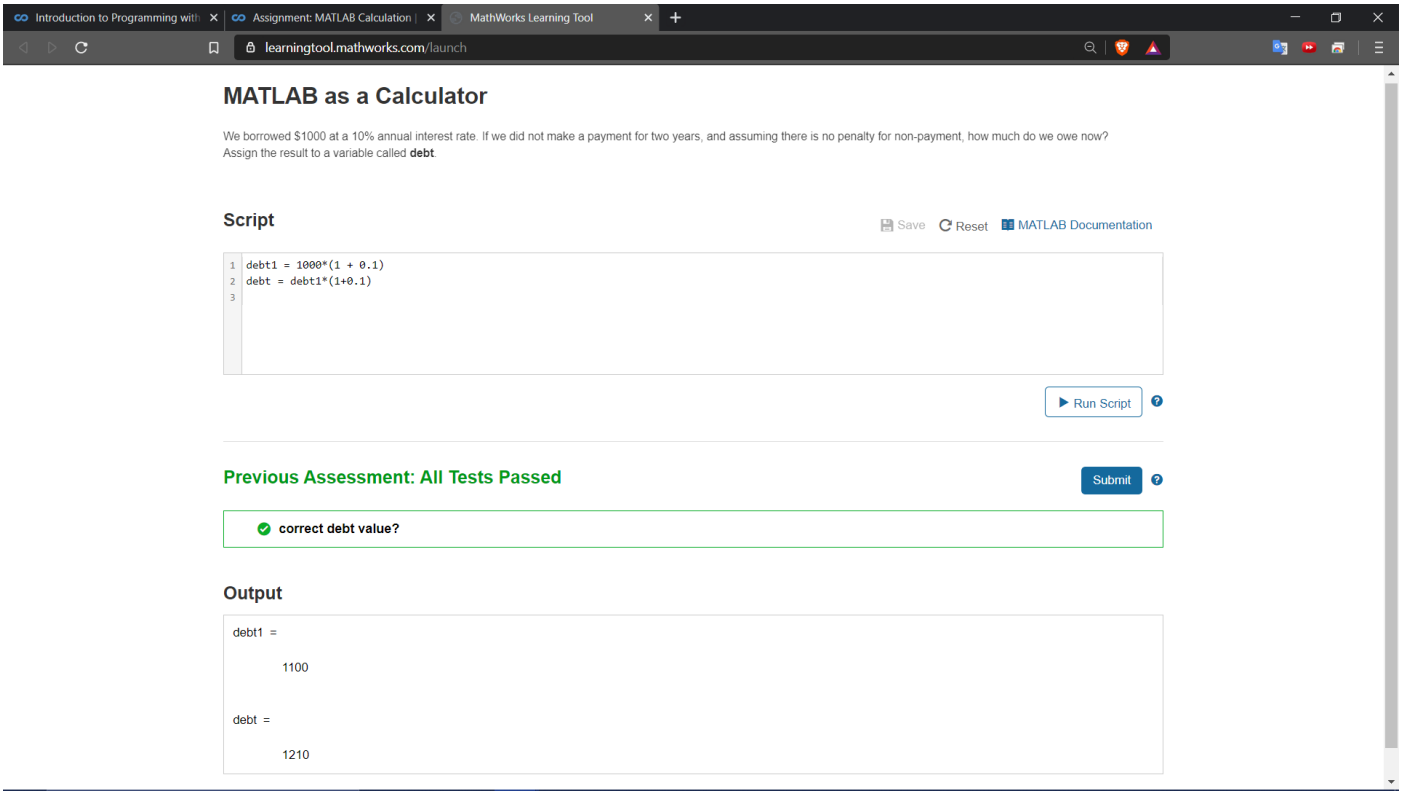
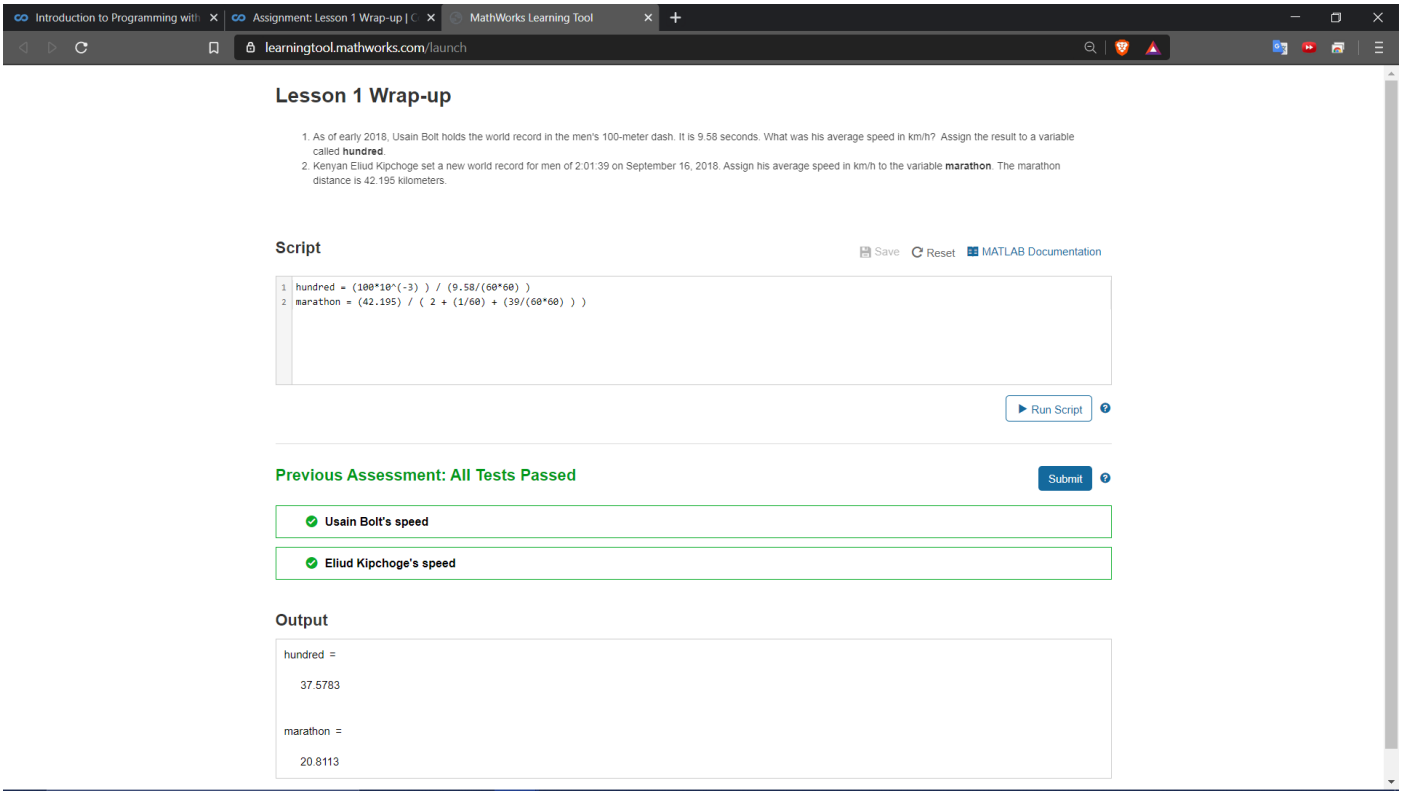
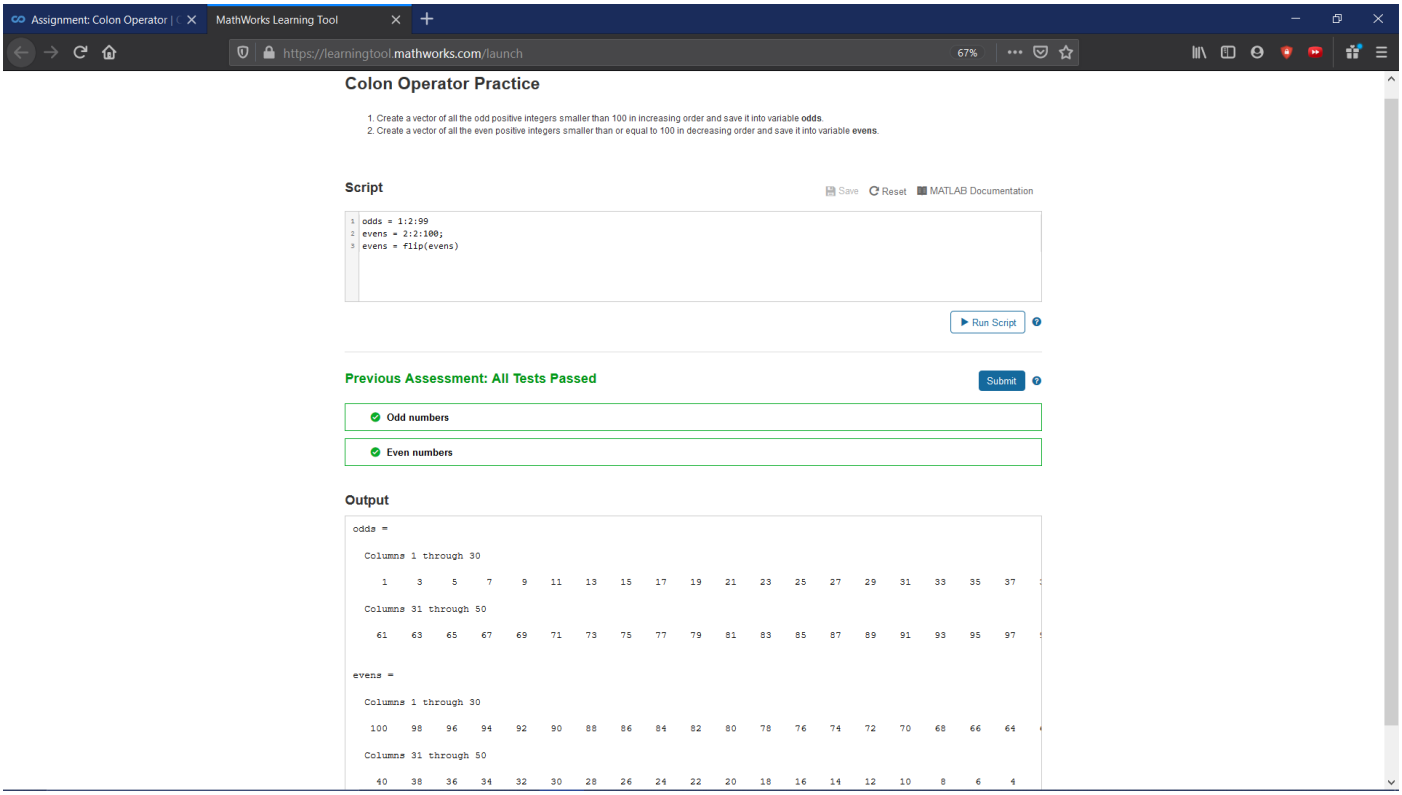
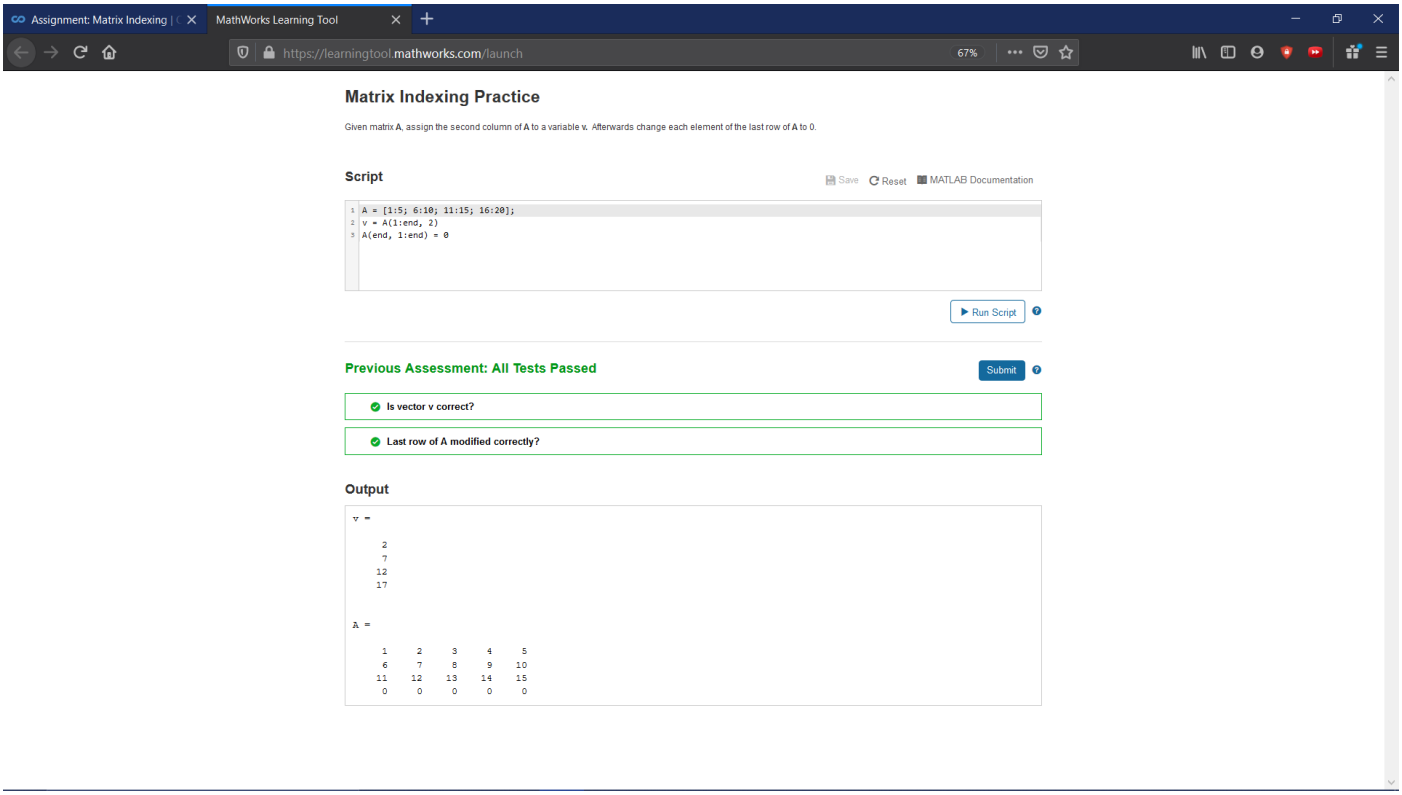
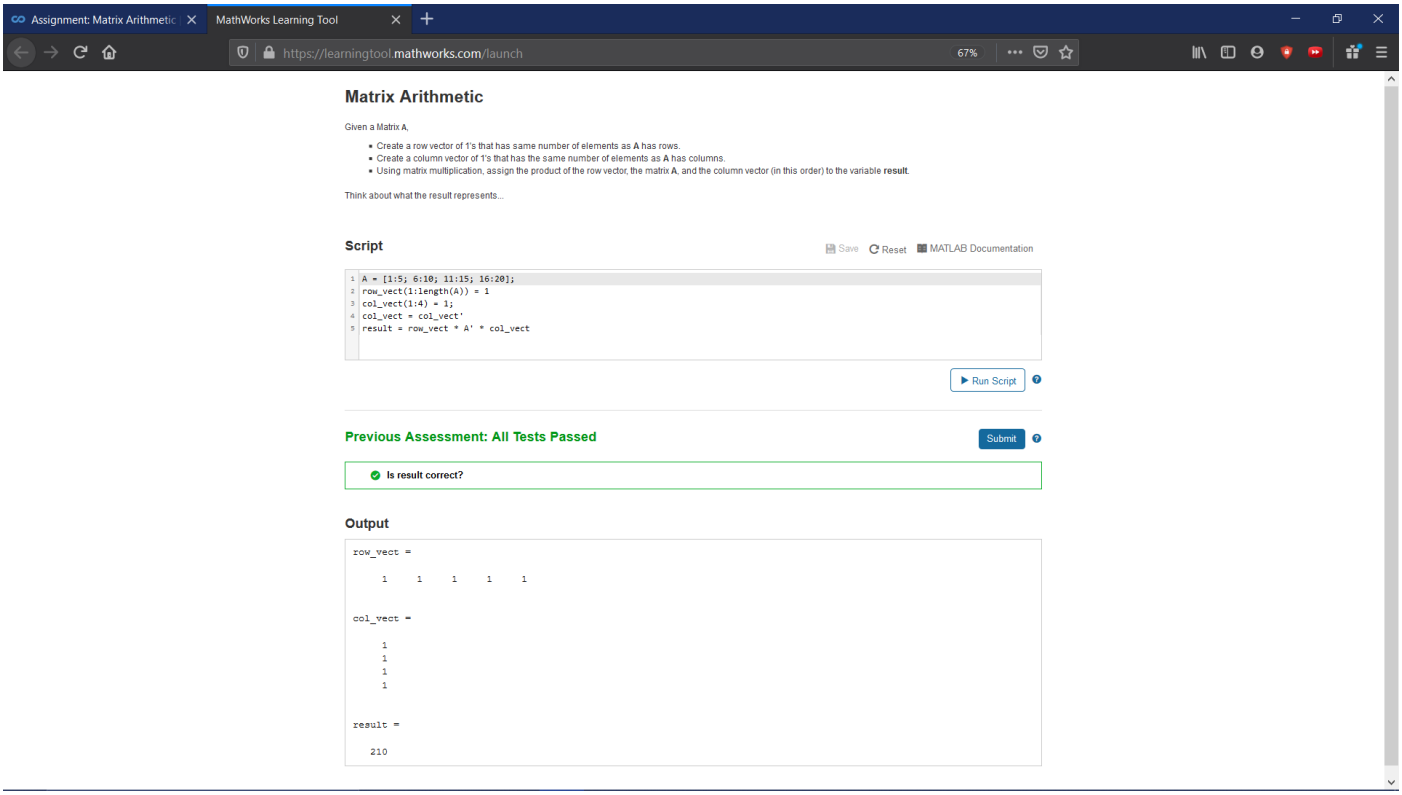
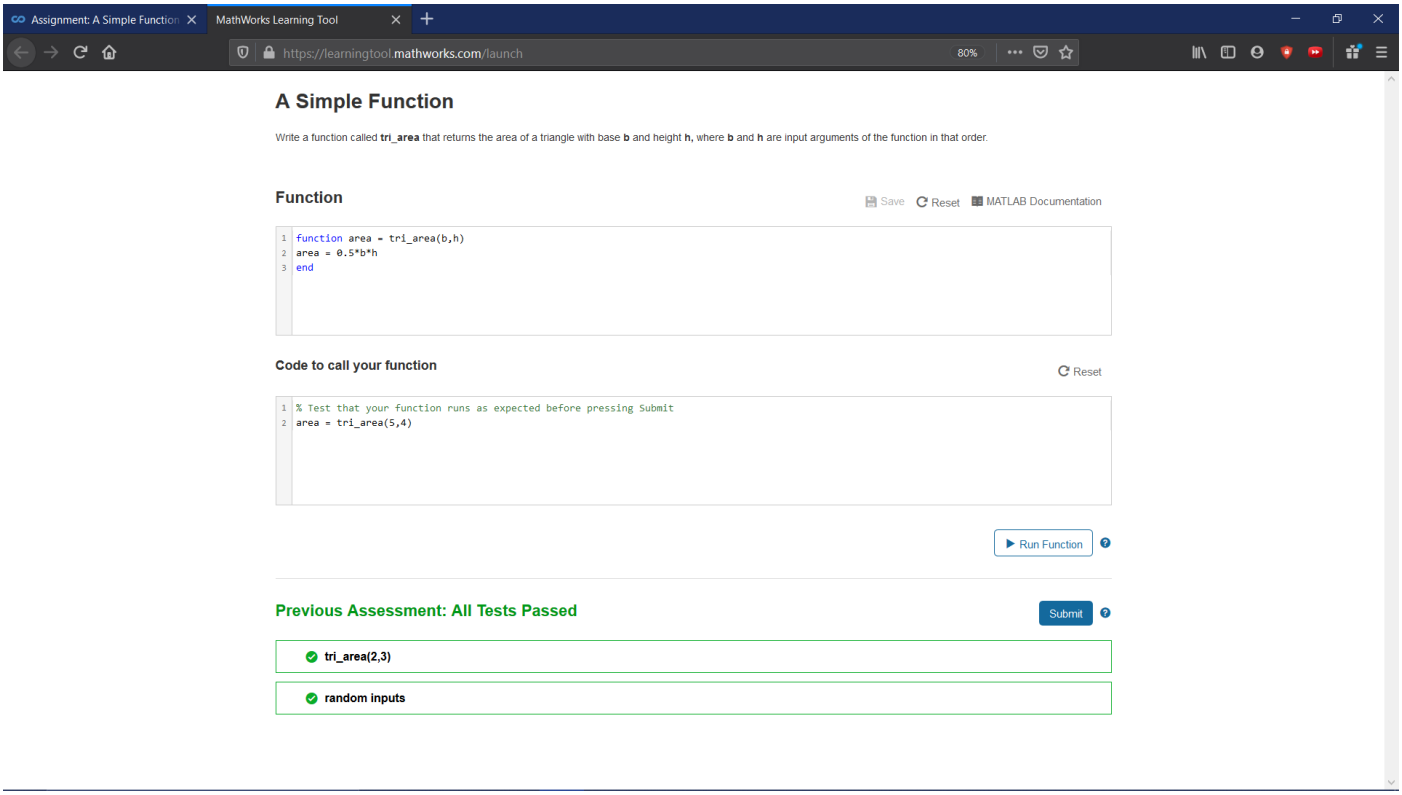

Introduction to Programming with
Assignment: MATLAB Calculation
MathWorks Learning Tool
learningtool.mathworks.com/launch
MATLAB as a Calculator
We borrowed $1000 at a 10% annual interest rate. If we did not make a payment for two years, and assuming there is no penalty for non-payment, how much do we owe now?
Assign the result to a variable called debt.
Script
Save
Reset
MATLAB Documentation
1 debt1 = 1000*(1 + 0.1)
2 debt = debt1*(1+0.1)
3
Run Script
Previous Assessment: All Tests Passed
Submit
correct debt value?
Output
debt1 =
1100
debt =
1210

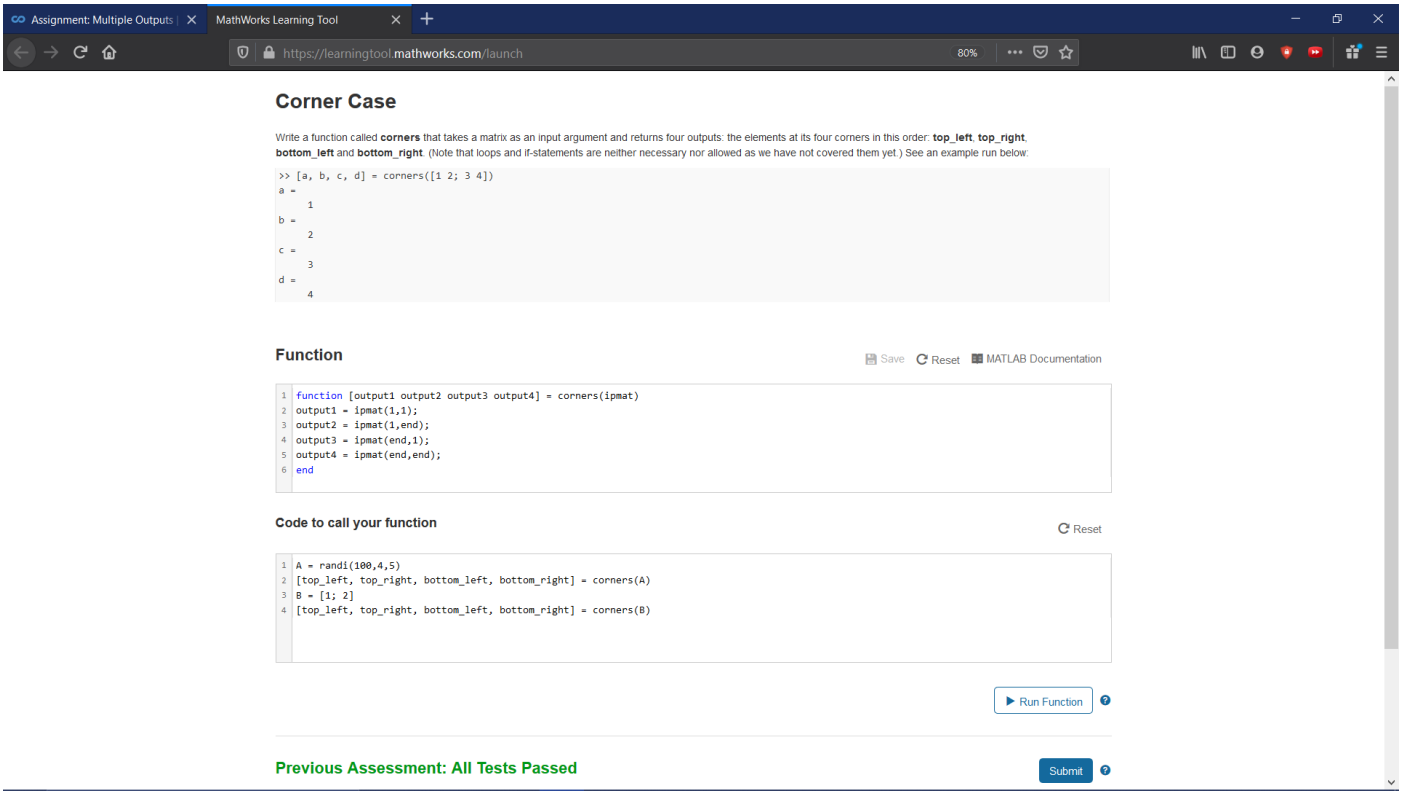

Introduction to Programming with
Assignment: Lesson 1 Wrap-up |
MathWorks Learning Tool
learningtool.mathworks.com/launch
Lesson 1 Wrap-up
1. As of early 2018, Usain Bolt holds the world record in the men's 100-meter dash. It is 9.58 seconds. What was his average speed in km/h? Assign the result to a variable called hundred.
2. Kenyan Eliud Kipchoge set a new world record for men of 2:01:39 on September 16, 2018. Assign his average speed in km/h to the variable marathon. The marathon distance is 42.195 kilometers.
Script
Save
Reset
MATLAB Documentation
1 hundred = (100*10^(-3)) / (9.58/(60*60))
2 marathon = (42.195) / (2 + (1/60) + (39/(60*60)))
Run Script
Previous Assessment: All Tests Passed
Submit
Usain Bolt's speed
Eliud Kipchoge's speed
Output
hundred =
37.5783
marathon =
20.8113

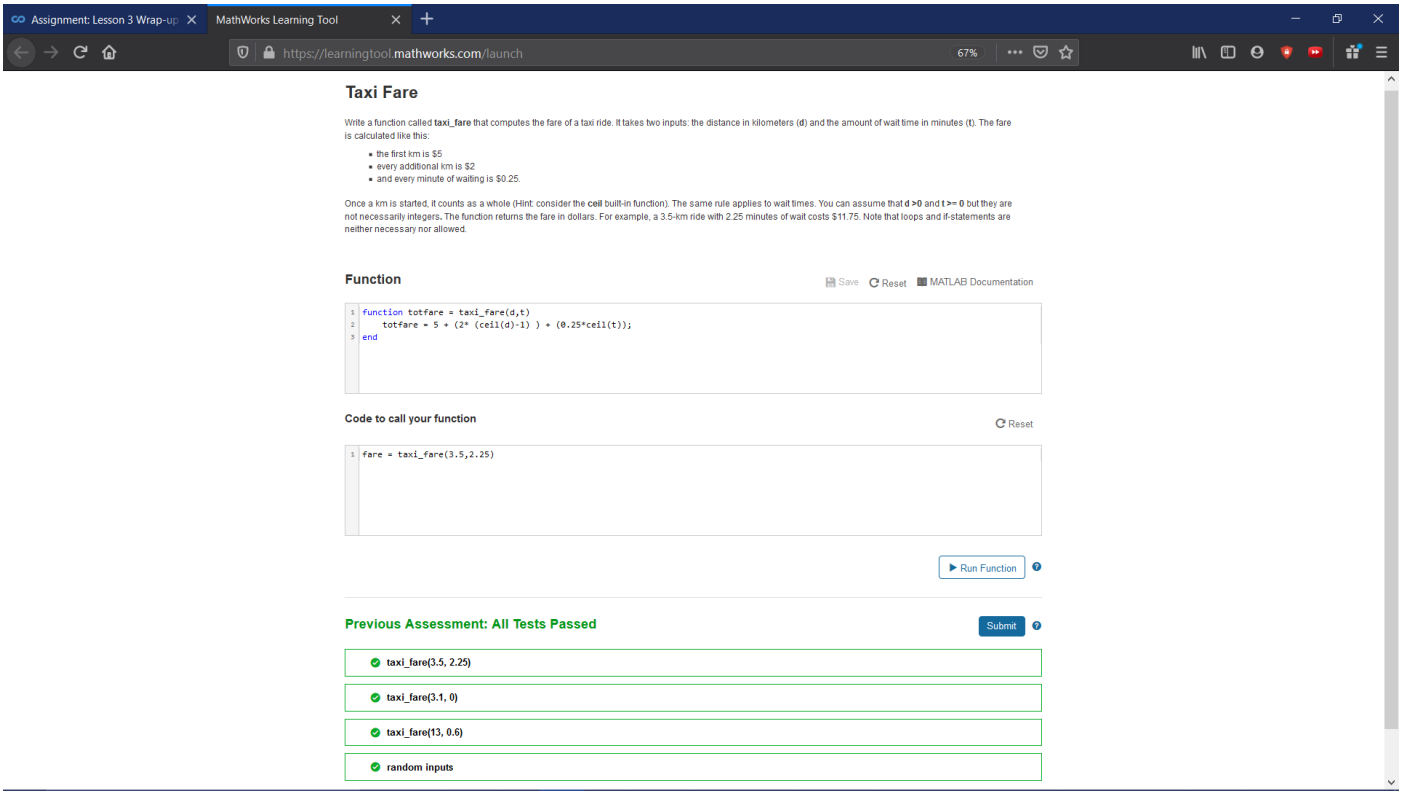

Assignment: Colon Operator
MathWorks Learning Tool
https://learningtool.mathworks.com/launch
67%
Colon Operator Practice
1. Create a vector of all the odd positive integers smaller than 100 in increasing order and save it into variable odds.
2. Create a vector of all the even positive integers smaller than or equal to 100 in decreasing order and save it into variable evens.
Script
Save
Reset
MATLAB Documentation
1 odds = 1:2:99
2 evens = 2:2:100;
3 evens = flip(evens)
Run Script
Previous Assessment: All Tests Passed
Submit
Odd numbers
Even numbers
Output
odds =
Columns 1 through 30
1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 33 35 37
Columns 31 through 50
61 63 65 67 69 71 73 75 77 79 81 83 85 87 89 91 93 95 97
evens =
Columns 1 through 30
100 98 96 94 92 90 88 86 84 82 80 78 76 74 72 70 68 66 64
Columns 31 through 50
40 38 36 34 32 30 28 26 24 22 20 18 16 14 12 10 8 6 4

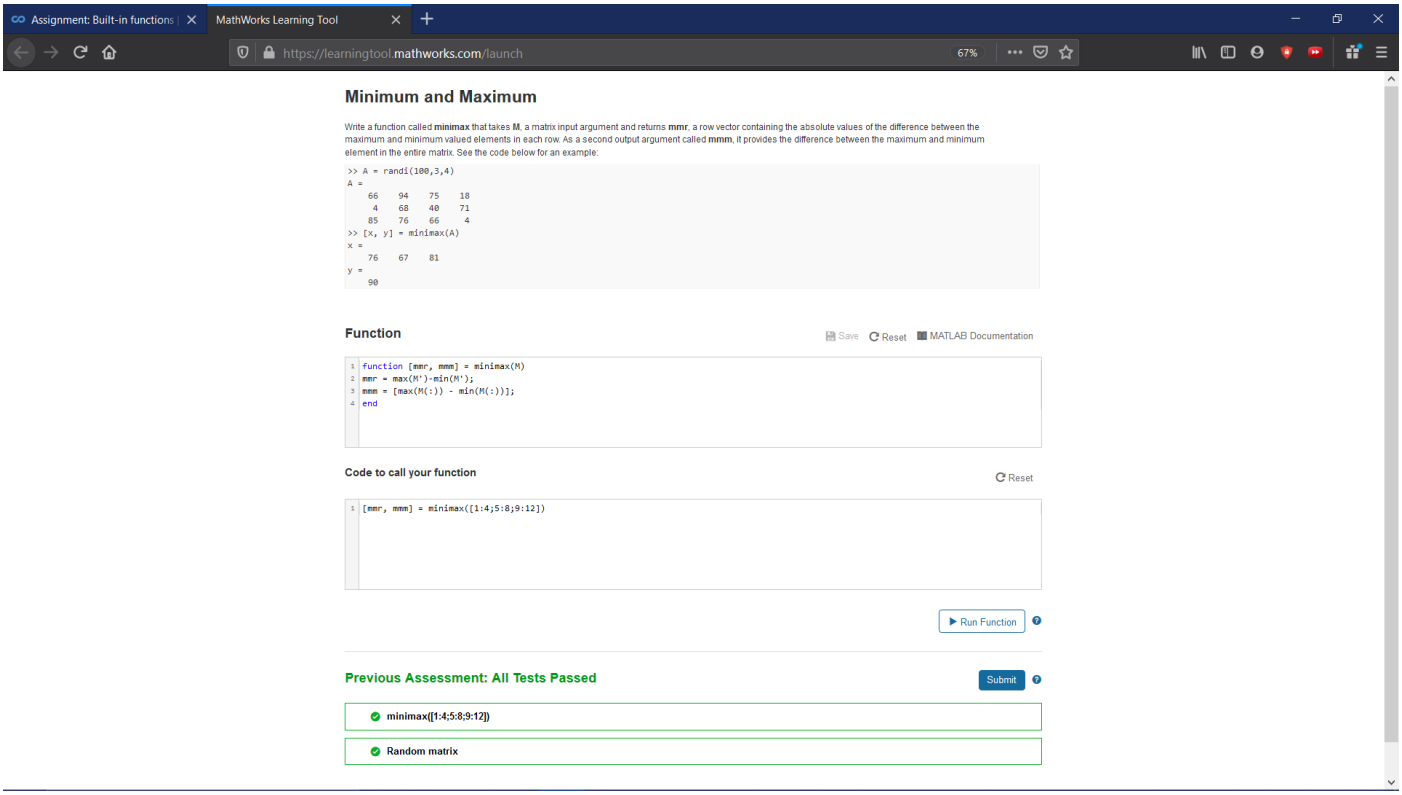

Assignment: Matrix Indexing
MathWorks Learning Tool
https://learningtool.mathworks.com/launch
67%
Matrix Indexing Practice
Given matrix A, assign the second column of A to a variable v. Afterwards change each element of the last row of A to 0.
Script
Save
Reset
MATLAB Documentation
1 A = [1:5; 6:10; 11:15; 16:20];
2 v = A(1:end, 2)
3 A(end, 1:end) = 0
Run Script
Previous Assessment: All Tests Passed
Submit
Is vector v correct?
Last row of A modified correctly?
Output
v =
2
7
12
17
A =
1 2 3 4 5
6 7 8 9 10
11 12 13 14 15
0 0 0 0 0

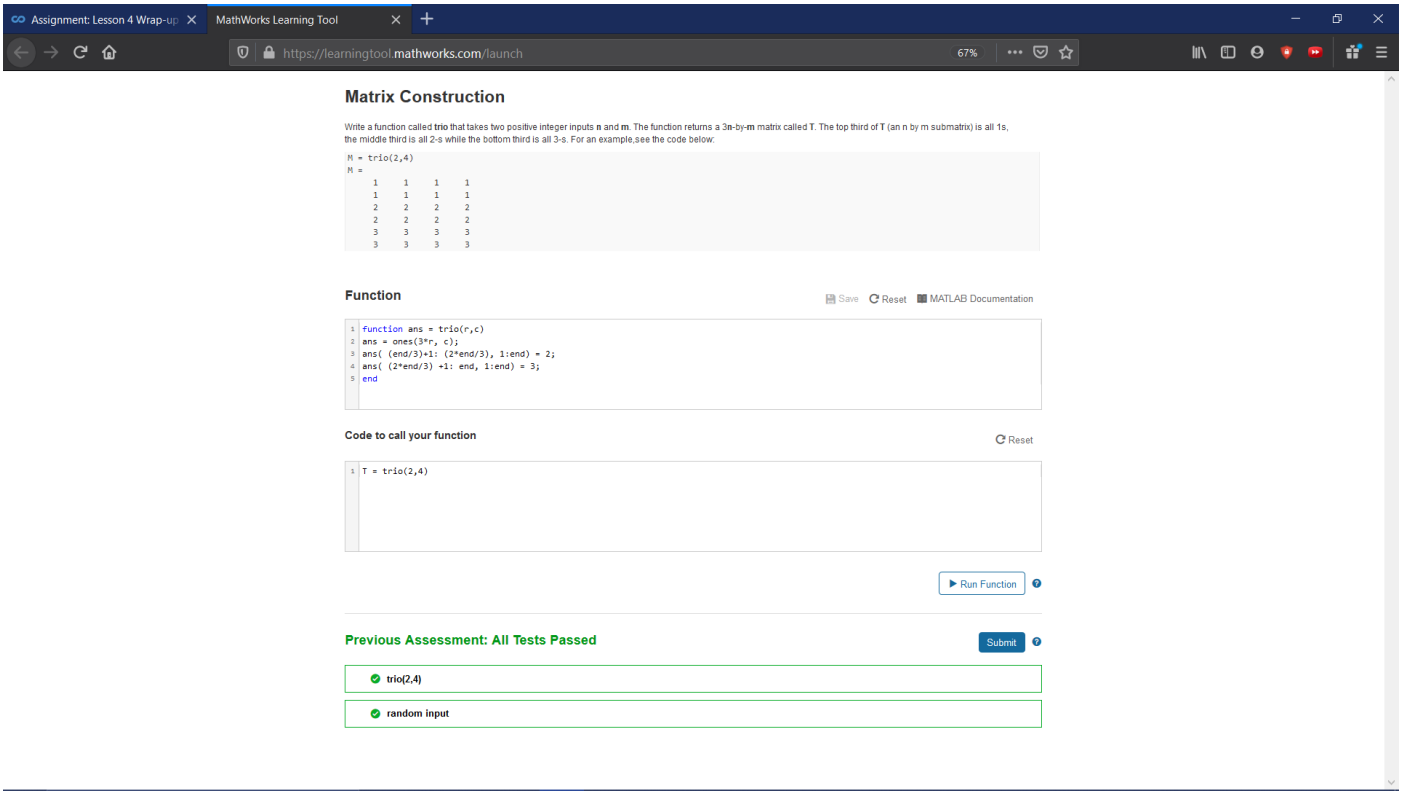

Assignment: Matrix Arithmetic
MathWorks Learning Tool
https://learningtool.mathworks.com/launch
67%
Matrix Arithmetic
Given a Matrix A,
Create a row vector of 1's that has same number of elements as A has rows.
Create a column vector of 1's that has the same number of elements as A has columns.
Using matrix multiplication, assign the product of the row vector, the matrix A, and the column vector (in this order) to the variable result.
Think about what the result represents...
Script
Save
Reset
MATLAB Documentation
A = [1:5; 6:10; 11:15; 16:20];
row_vect(1:length(A)) = 1
col_vect(1:4) = 1;
col_vect = col_vect'
result = row_vect * A' * col_vect
Run Script
Previous Assessment: All Tests Passed
Submit
Is result correct?
Output
row_vect =
1 1 1 1 1
col_vect =
1
1
1
1
result =
210

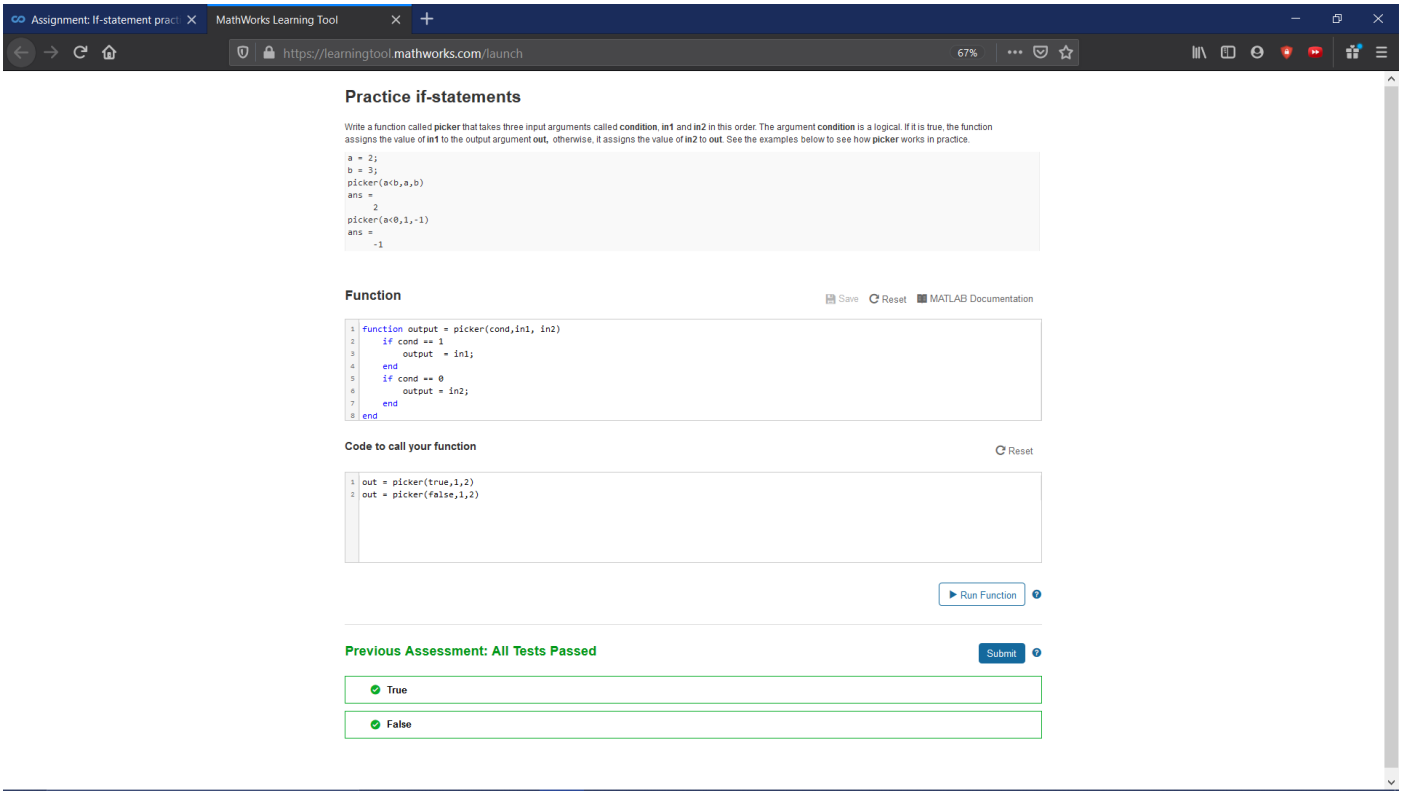

Assignment: A Simple Function
MathWorks Learning Tool
https://learningtool.mathworks.com/launch
80%
A Simple Function
Write a function called tri_area that returns the area of a triangle with base b and height h, where b and h are input arguments of the function in that order.
Function
Save
Reset
MATLAB Documentation
function area = tri_area(b,h)
area = 0.5*b*h
end
Code to call your function
Reset
% Test that your function runs as expected before pressing Submit
area = tri_area(5,4)
Run Function
Previous Assessment: All Tests Passed
Submit
tri_area(2,3)
random inputs

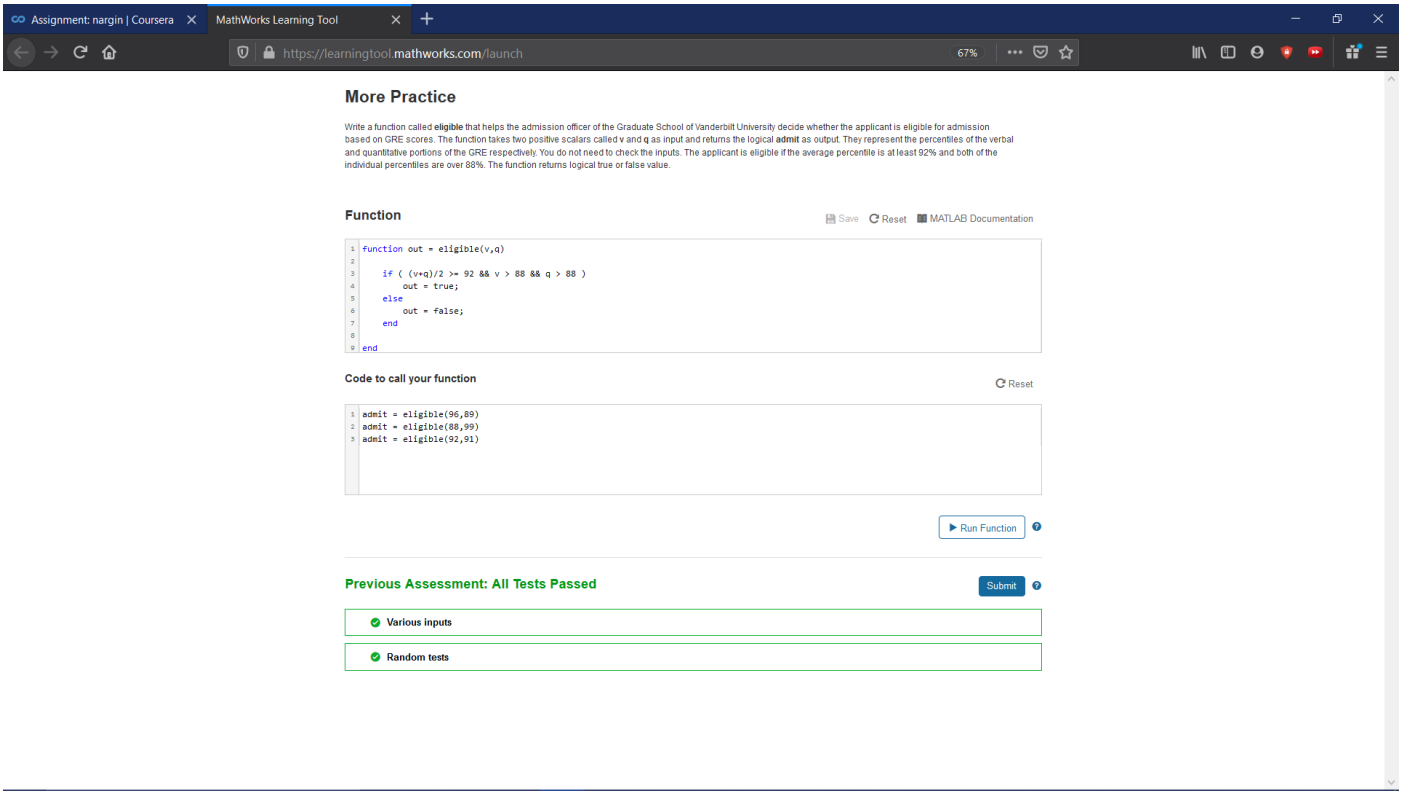

Assignment: Multiple Outputs
MathWorks Learning Tool
https://learningtool.mathworks.com/launch
80%
Corner Case
Write a function called corners that takes a matrix as an input argument and returns four outputs: the elements at its four corners in this order: top_left, top_right, bottom_left and bottom_right. (Note that loops and if-statements are neither necessary nor allowed as we have not covered them yet.) See an example run below:
>> [a, b, c, d] = corners([1 2; 3 4])
a =
1
b =
2
c =
3
d =
4
Function
Save
Reset
MATLAB Documentation
function [output1 output2 output3 output4] = corners(ipmat)
output1 = ipmat(1,1);
output2 = ipmat(1,end);
output3 = ipmat(end,1);
output4 = ipmat(end,end);
end
Code to call your function
Reset
A = randi(100,4,5)
[top_left, top_right, bottom_left, bottom_right] = corners(A)
B = [1; 2]
[top_left, top_right, bottom_left, bottom_right] = corners(B)
Run Function
Previous Assessment: All Tests Passed
Submit

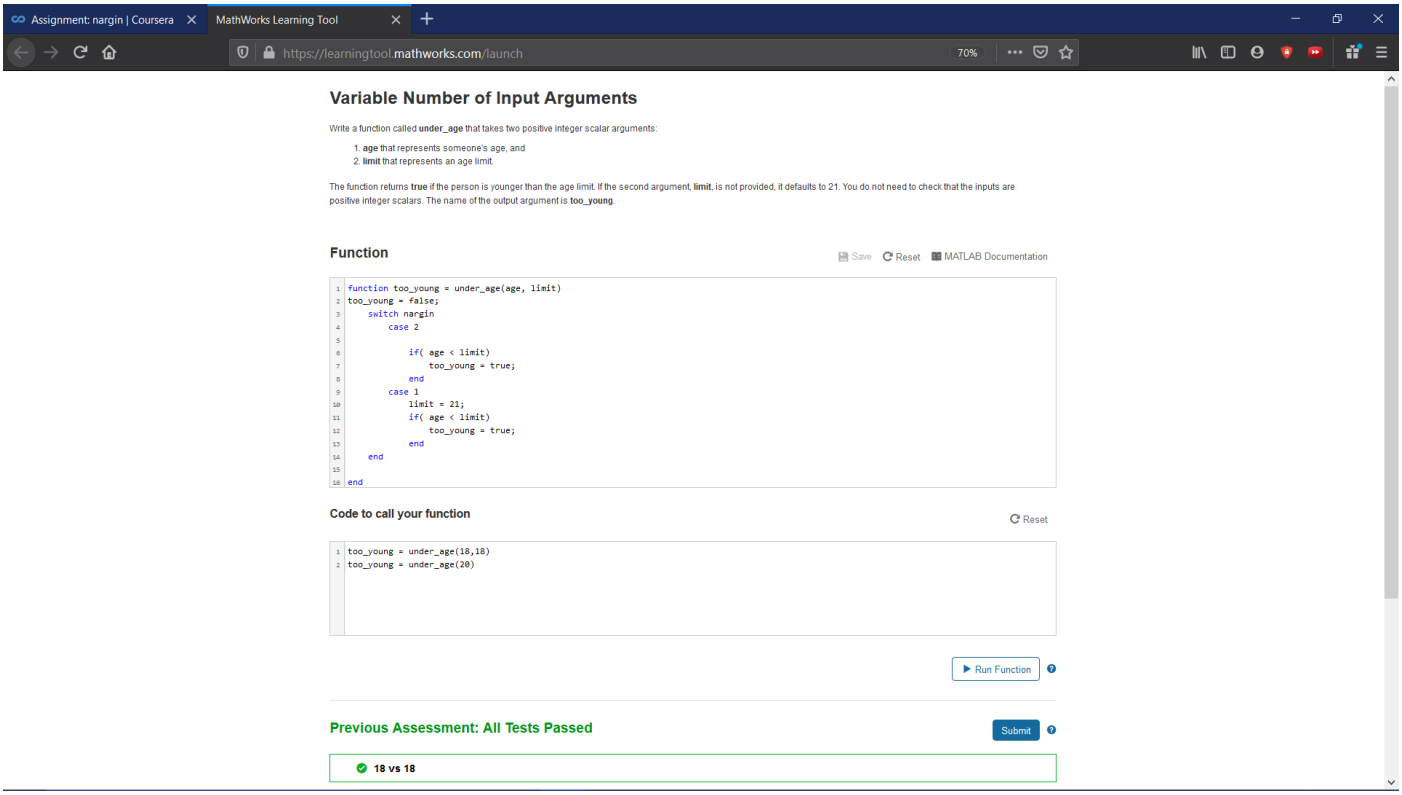

Assignment: Lesson 3 Wrap-up
MathWorks Learning Tool
https://learningtool.mathworks.com/launch
67%
Taxi Fare
Write a function called taxi_fare that computes the fare of a taxi ride. It takes two inputs: the distance in kilometers (d) and the amount of wait time in minutes (t). The fare is calculated like this:
the first km is $5
every additional km is $2
and every minute of waiting is $0.25.
Once a km is started, it counts as a whole (Hint: consider the ceil built-in function). The same rule applies to wait times. You can assume that d >0 and t >= 0 but they are not necessarily integers. The function returns the fare in dollars. For example, a 3.5-km ride with 2.25 minutes of wait costs $11.75. Note that loops and if-statements are neither necessary nor allowed.
Function
Save
Reset
MATLAB Documentation
function totfare = taxi_fare(d,t)
totfare = 5 + (2* (ceil(d)-1)) + (0.25*ceil(t));
end
Code to call your function
Reset
fare = taxi_fare(3.5,2.25)
Run Function
Previous Assessment: All Tests Passed
Submit
taxi_fare(3.5, 2.25)
taxi_fare(3.1, 0)
taxi_fare(13, 0.6)
random inputs

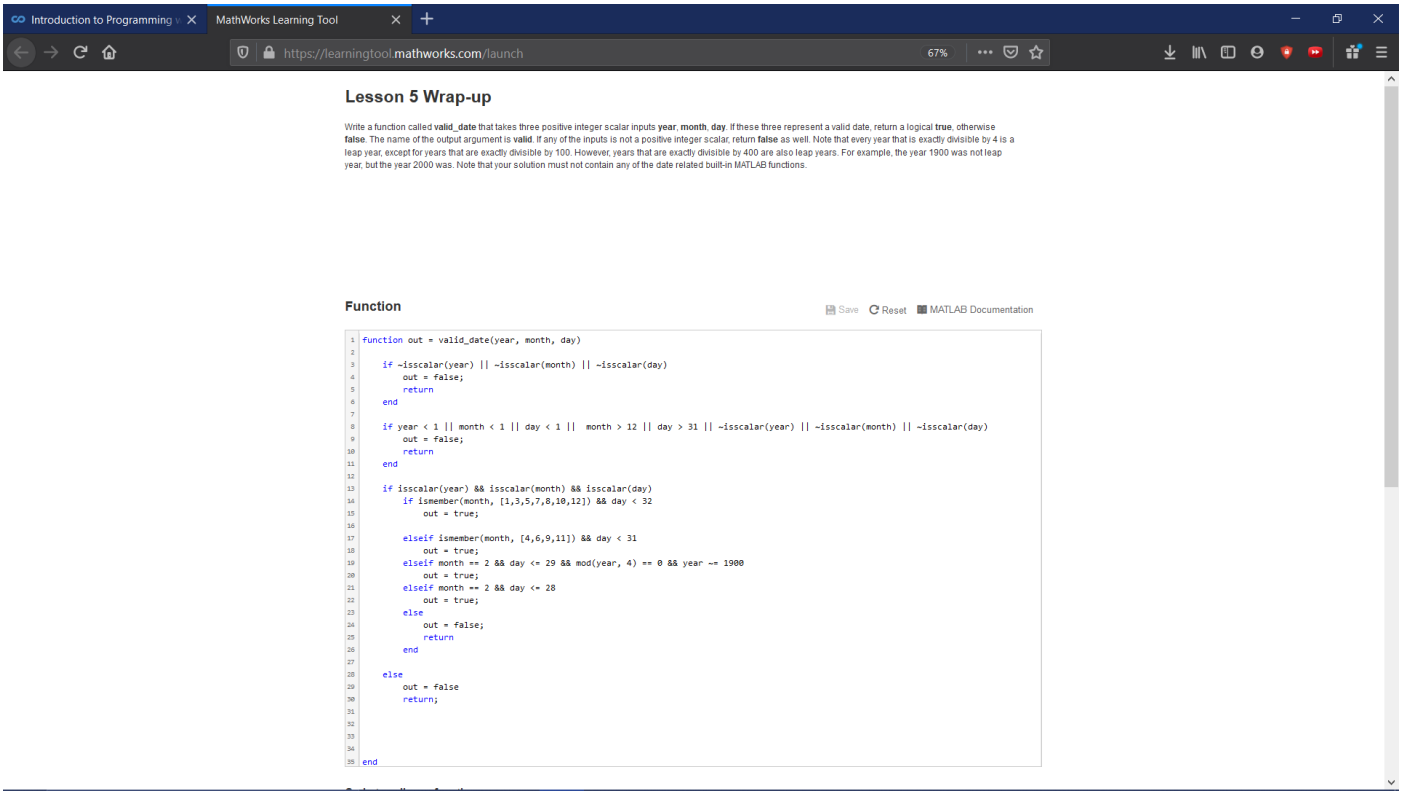

Assignment: Built-in functions
MathWorks Learning Tool
https://learningtool.mathworks.com/launch
67%
Minimum and Maximum
Write a function called minimax that takes M, a matrix input argument and returns mmr, a row vector containing the absolute values of the difference between the maximum and minimum valued elements in each row. As a second output argument called mmm, it provides the difference between the maximum and minimum element in the entire matrix. See the code below for an example:
>> A = randi(100,3,4)
A =
66 94 75 18
4 68 40 71
85 76 66 4
>> [x, y] = minimax(A)
x =
76 67 81
y =
90
Function
Save
Reset
MATLAB Documentation
function [mmr, mmm] = minimax(M)
mmr = max(M')-min(M');
mmm = [max(M(:)) - min(M(:))];
end
Code to call your function
Reset
[mmr, mmm] = minimax([1:4;5:8;9:12])
Run Function
Previous Assessment: All Tests Passed
Submit
minimax([1:4;5:8;9:12])
Random matrix


Assignment: Lesson 4 Wrap-up
MathWorks Learning Tool
https://learningtool.mathworks.com/launch
67%
Matrix Construction
Write a function called trio that takes two positive integer inputs n and m. The function returns a 3n-by-m matrix called T. The top third of T (an n by m submatrix) is all 1s, the middle third is all 2-s while the bottom third is all 3-s. For an example,see the code below:
M = trio(2,4)
M =
1 1 1 1
1 1 1 1
2 2 2 2
2 2 2 2
3 3 3 3
3 3 3 3
Function
Save
Reset
MATLAB Documentation
function ans = trio(r,c)
ans = ones(3*r, c);
ans((end/3)+1: (2*end/3), 1:end) = 2;
ans((2*end/3) +1: end, 1:end) = 3;
end
Code to call your function
Reset
T = trio(2,4)
Run Function
Previous Assessment: All Tests Passed
Submit
trio(2,4)
random input


Assignment: If-statement practi
MathWorks Learning Tool
https://learningtool.mathworks.com/launch
67%
Practice if-statements
Write a function called picker that takes three input arguments called condition, in1 and in2 in this order. The argument condition is a logical. If it is true, the function assigns the value of in1 to the output argument out, otherwise, it assigns the value of in2 to out. See the examples below to see how picker works in practice.
a = 2;
b = 3;
picker(a<b,a,b)
ans =
2
picker(a<0,1,-1)
ans =
-1
Function
Save
Reset
MATLAB Documentation
function output = picker(cond,in1, in2)
if cond == 1
output = in1;
end
if cond == 0
output = in2;
end
end
Code to call your function
Reset
out = picker(true,1,2)
out = picker(false,1,2)
Run Function
Previous Assessment: All Tests Passed
Submit
True
False


Assignment: nargin | Coursera
MathWorks Learning Tool
https://learningtool.mathworks.com/launch
67%
More Practice
Write a function called eligible that helps the admission officer of the Graduate School of Vanderbilt University decide whether the applicant is eligible for admission based on GRE scores. The function takes two positive scalars called v and q as input and returns the logical admit as output. They represent the percentiles of the verbal and quantitative portions of the GRE respectively. You do not need to check the inputs. The applicant is eligible if the average percentile is at least 92% and both of the individual percentiles are over 88%. The function returns logical true or false value.
Function
Save
Reset
MATLAB Documentation
function out = eligible(v,q)
if ((v+q)/2 >= 92 && v > 88 && q > 88)
out = true;
else
out = false;
end
end
Code to call your function
Reset
admit = eligible(96,89)
admit = eligible(88,99)
admit = eligible(92,91)
Run Function
Previous Assessment: All Tests Passed
Submit
Various inputs
Random tests


Assignment: nargin | Coursera
MathWorks Learning Tool
https://learningtool.mathworks.com/launch
70%
Variable Number of Input Arguments
Write a function called under_age that takes two positive integer scalar arguments:
1. age that represents someone's age, and
2. limit that represents an age limit.
The function returns true if the person is younger than the age limit. If the second argument, limit, is not provided, it defaults to 21. You do not need to check that the inputs are positive integer scalars. The name of the output argument is too_young.
Function
Save
Reset
MATLAB Documentation
function too_young = under_age(age, limit)
too_young = false;
switch nargin
case 2
if(age < limit)
too_young = true;
end
case 1
limit = 21;
if(age < limit)
too_young = true;
end
end
end
Code to call your function
Reset
too_young = under_age(18,18)
too_young = under_age(20)
Run Function
Previous Assessment: All Tests Passed
Submit
18 vs 18



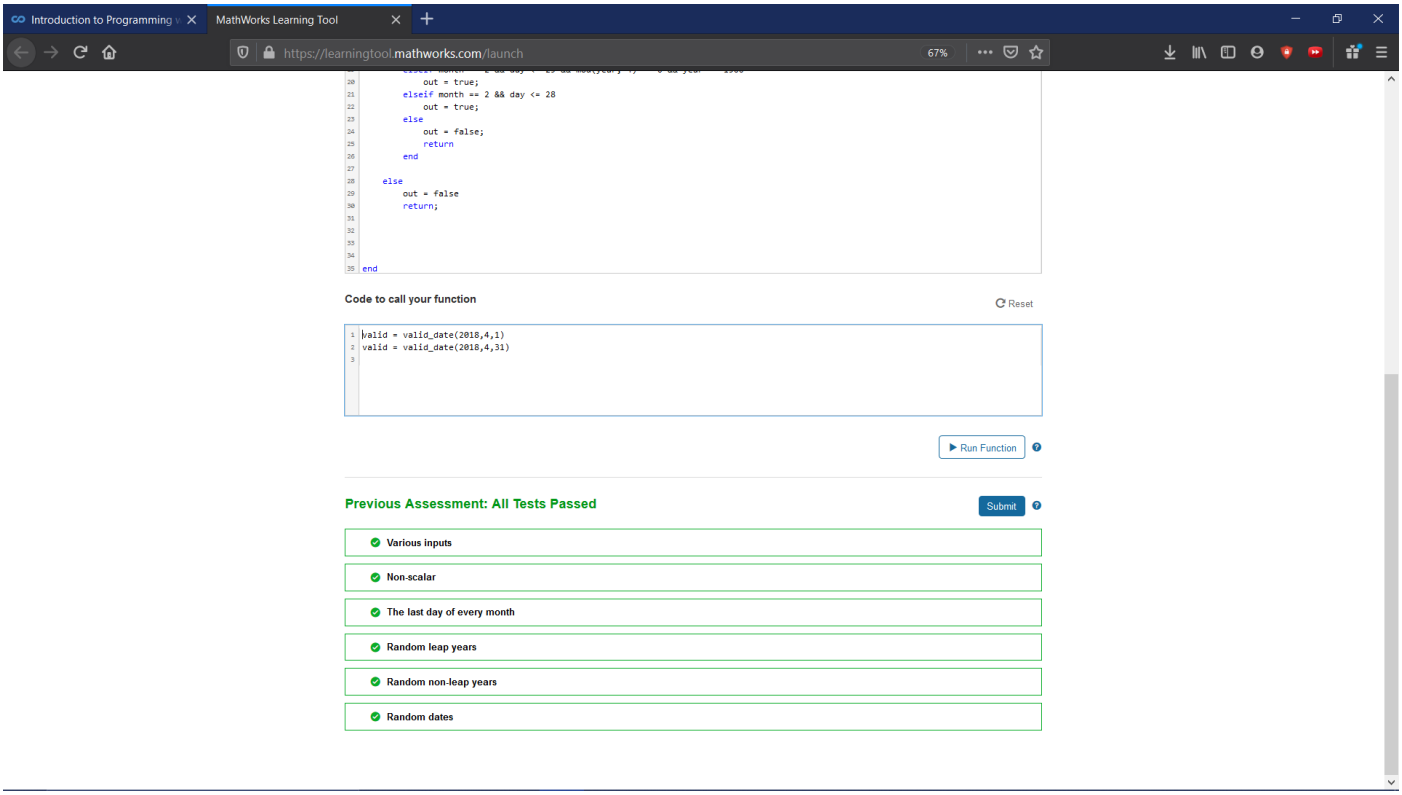
Lesson 5 Wrap-up

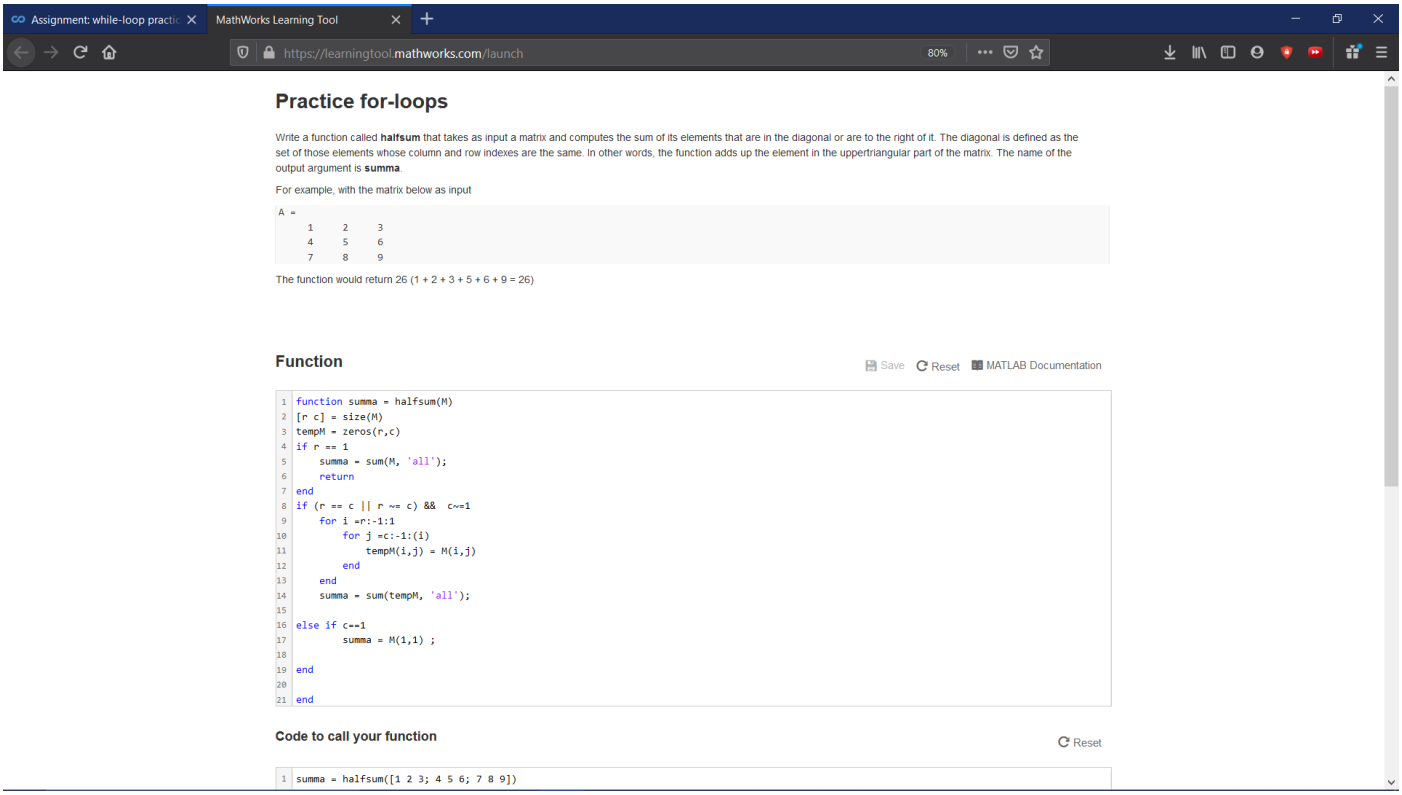
Write a function called **valid_date** that takes three positive integer scalar inputs **year**, **month**, **day**. If these three represent a valid date, return a logical **true**, otherwise **false**. The name of the output argument is **valid**. If any of the inputs is not a positive integer scalar, return **false** as well. Note that every year that is exactly divisible by 4 is a leap year, except for years that are exactly divisible by 100. However, years that are exactly divisible by 400 are also leap years. For example, the year 1900 was not leap year, but the year 2000 was. Note that your solution must not contain any of the date related built-in MATLAB functions.

Function

Save Reset MATLAB Documentation

```
function out = valid_date(year, month, day)

    if ~isscalar(year) || ~isscalar(month) || ~isscalar(day)
        out = false;
        return
    end

    if year < 1 || month < 1 || day < 1 ||  month > 12 || day > 31 || ~isscalar(year) || ~isscalar(month) || ~isscalar(day)
        out = false;
        return
    end

    if isscalar(year) && isscalar(month) && isscalar(day)
        if ismember(month, [1,3,5,7,8,10,12]) && day < 32
            out = true;

        elseif ismember(month, [4,6,9,11]) && day < 31
            out = true;
        elseif month == 2 && day <= 29 && mod(year, 4) == 0 && year ~= 1900
            out = true;
        elseif month == 2 && day <= 28
            out = true;
        else
            out = false;
            return
        end

    else
        out = false
        return;




end
```


Introduction to Programming
MathWorks Learning Tool
https://learningtool.mathworks.com/launch
67%
out = true;
elseif month == 2 && day <= 28
out = true;
else
out = false;
return
end
else
out = false
return;
end
Code to call your function
Reset
valid = valid_date(2018,4,1)
valid = valid_date(2018,4,31)
Run Function
Previous Assessment: All Tests Passed
Submit
Various inputs
Non-scalar
The last day of every month
Random leap years
Random non-leap years
Random dates



Practice for-loops

Write a function called **halfsum** that takes as input a matrix and computes the sum of its elements that are in the diagonal or are to the right of it. The diagonal is defined as the set of those elements whose column and row indexes are the same. In other words, the function adds up the element in the uppertriangular part of the matrix. The name of the output argument is **summa**.

For example, with the matrix below as input

```
A =
     1     2     3
     4     5     6
     7     8     9
```

The function would return 26 (1 + 2 + 3 + 5 + 6 + 9 = 26)

Function

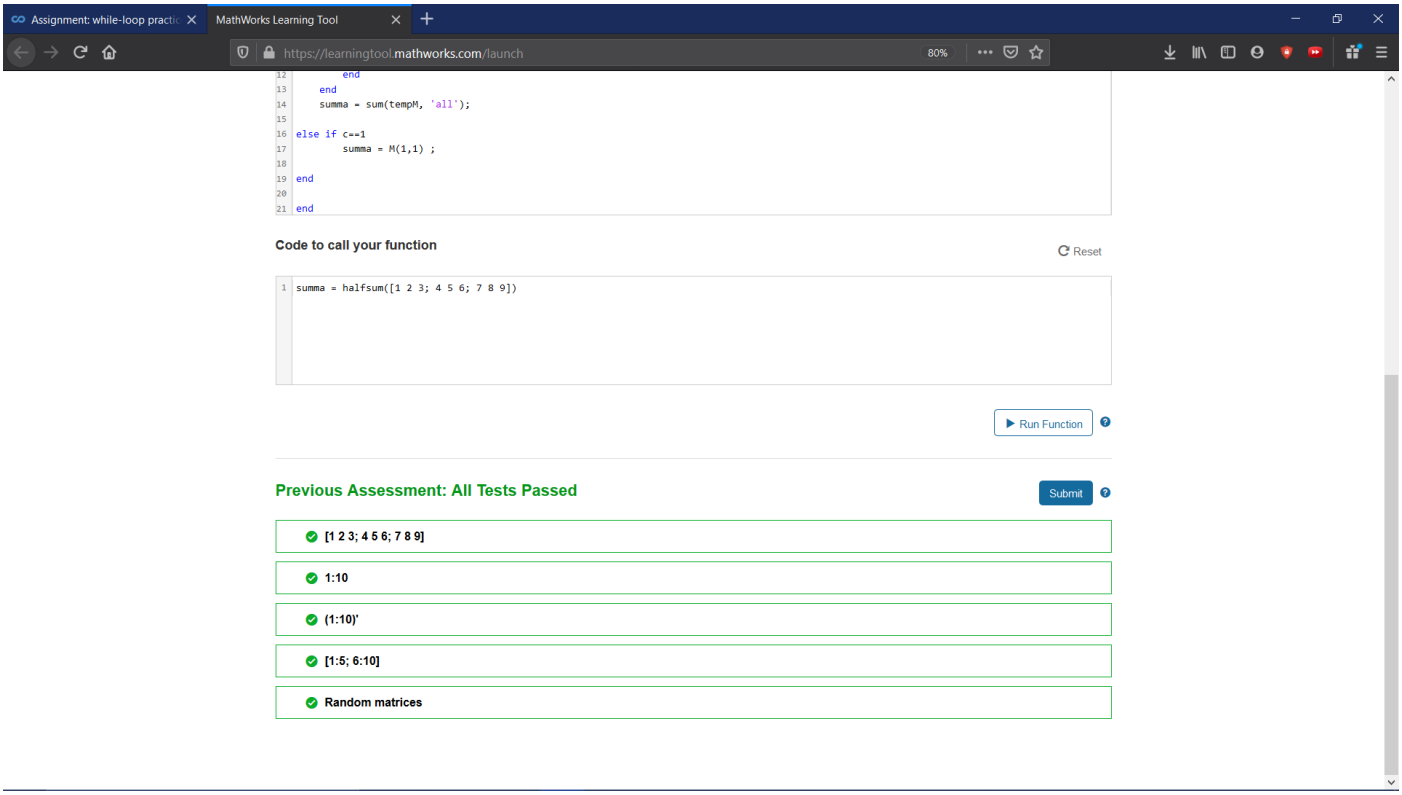
Save Reset MATLAB Documentation

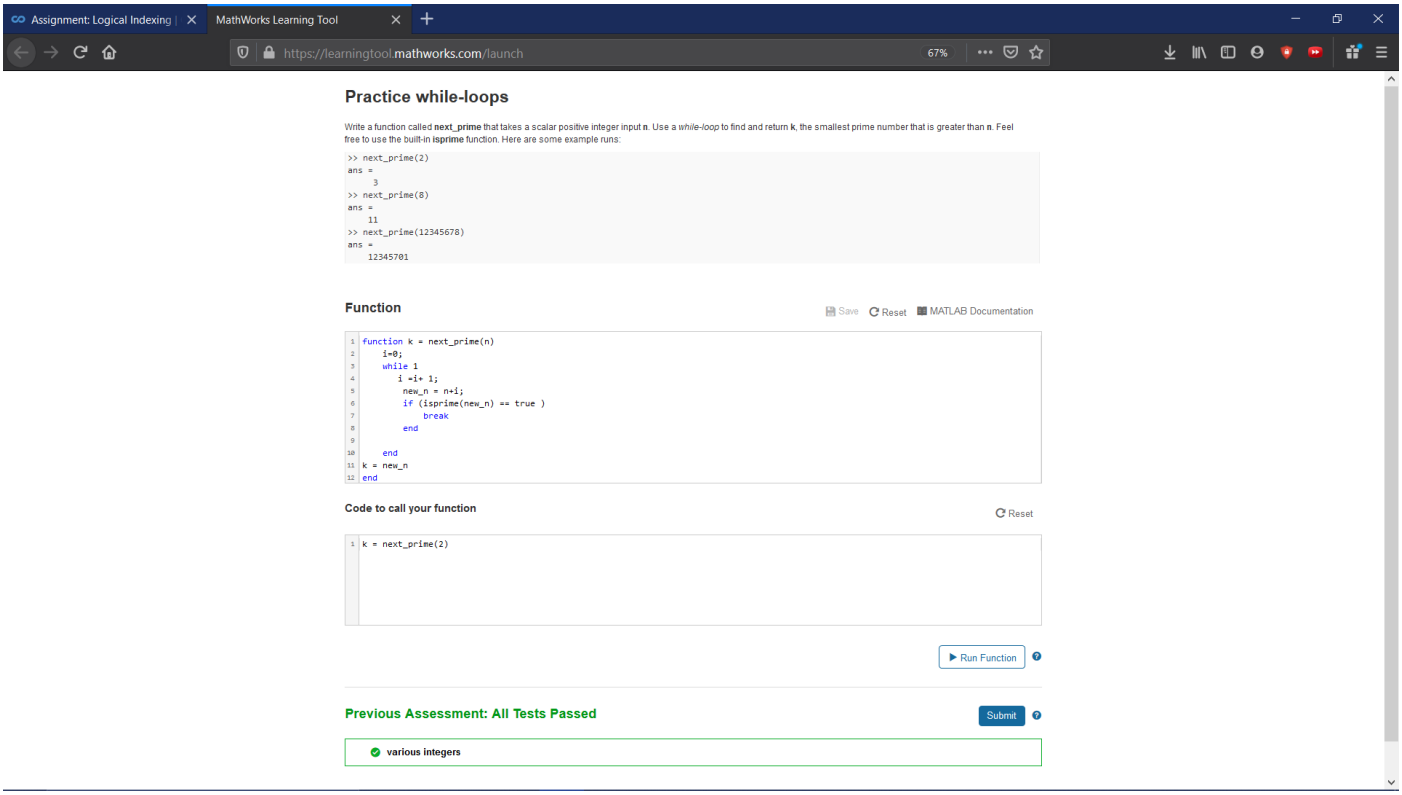
```
function summa = halfsum(M)
[r c] = size(M)
tempM = zeros(r,c)
if r == 1
    summa = sum(M, 'all');
    return
end
if (r == c || r ~= c) &&  c~=1
    for i =r:-1:1
        for j =c:-1:(i)
            tempM(i,j) = M(i,j)
        end
    end
    summa = sum(tempM, 'all');

else if c==1
        summa = M(1,1) ;

end

end
```

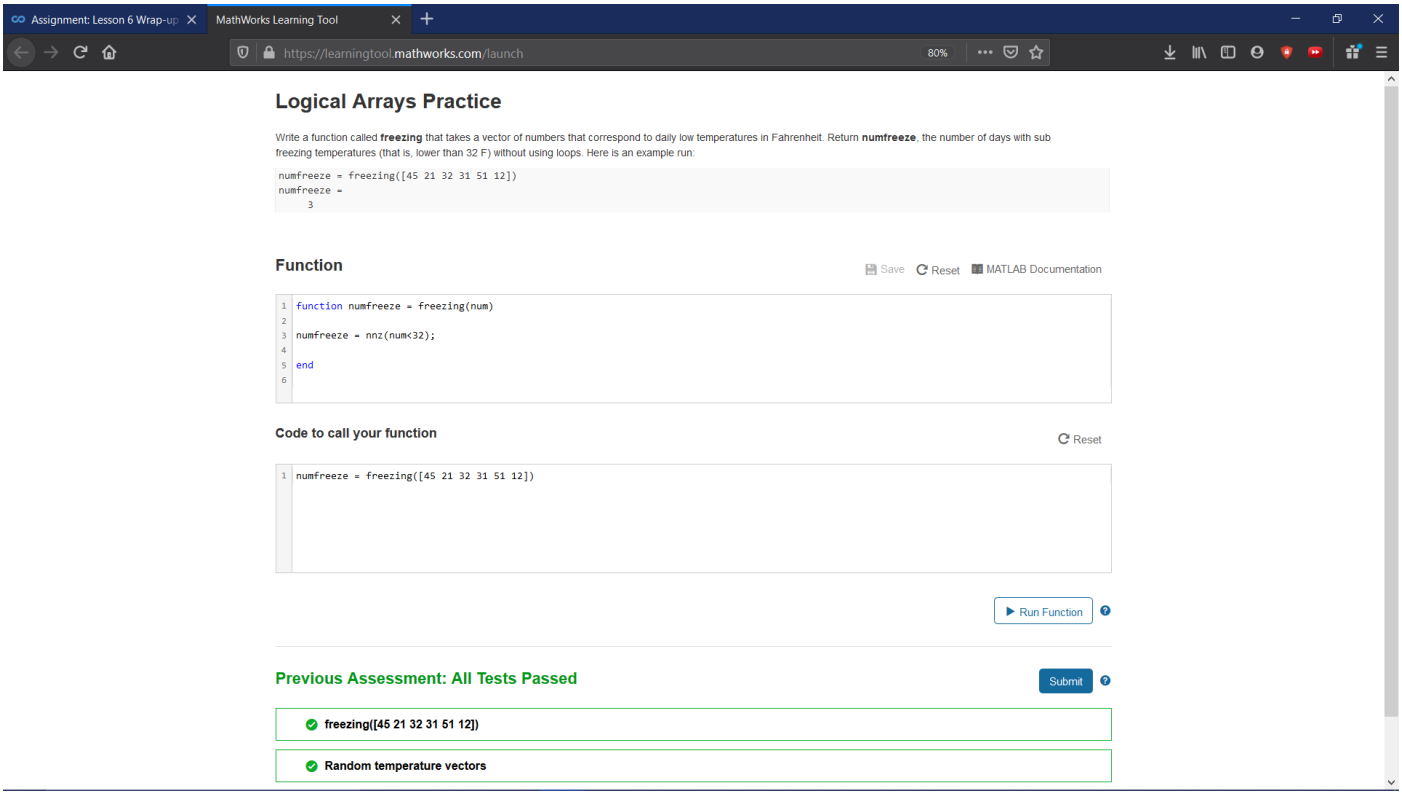
Code to call your function

Reset

```
summa = halfsum([1 2 3; 4 5 6; 7 8 9])
```


Assignment: while-loop practic
MathWorks Learning Tool
https://learningtool.mathworks.com/launch
80%
end
end
summa = sum(tempM, 'all');
else if c==1
summa = M(1,1) ;
end
end
Code to call your function
Reset
summa = halfsum([1 2 3; 4 5 6; 7 8 9])
Run Function
Previous Assessment: All Tests Passed
Submit
[1 2 3; 4 5 6; 7 8 9]
1:10
(1:10)'
[1:5; 6:10]
Random matrices


Assignment: Logical Indexing
MathWorks Learning Tool
https://learningtool.mathworks.com/launch
67%
Practice while-loops
Write a function called next_prime that takes a scalar positive integer input n. Use a while-loop to find and return k, the smallest prime number that is greater than n. Feel free to use the built-in isprime function. Here are some example runs:
>> next_prime(2)
ans =
3
>> next_prime(8)
ans =
11
>> next_prime(12345678)
ans =
12345701
Function
Save
Reset
MATLAB Documentation
function k = next_prime(n)
i=0;
while 1
i =i+ 1;
new_n = n+i;
if (isprime(new_n) == true)
break
end
end
k = new_n
end
Code to call your function
Reset
k = next_prime(2)
Run Function
Previous Assessment: All Tests Passed
Submit
various integers


Assignment: Lesson 6 Wrap-up
MathWorks Learning Tool
https://learningtool.mathworks.com/launch
80%
Logical Arrays Practice
Write a function called freezing that takes a vector of numbers that correspond to daily low temperatures in Fahrenheit. Return numfreeze, the number of days with sub freezing temperatures (that is, lower than 32 F) without using loops. Here is an example run:
numfreeze = freezing([45 21 32 31 51 12])
numfreeze =
3
Function
Save
Reset
MATLAB Documentation
function numfreeze = freezing(num)
numfreeze = nnz(num<32);
end
Code to call your function
Reset
numfreeze = freezing([45 21 32 31 51 12])
Run Function
Previous Assessment: All Tests Passed
Submit
freezing([45 21 32 31 51 12])
Random temperature vectors

Lesson 6 Wrap-up

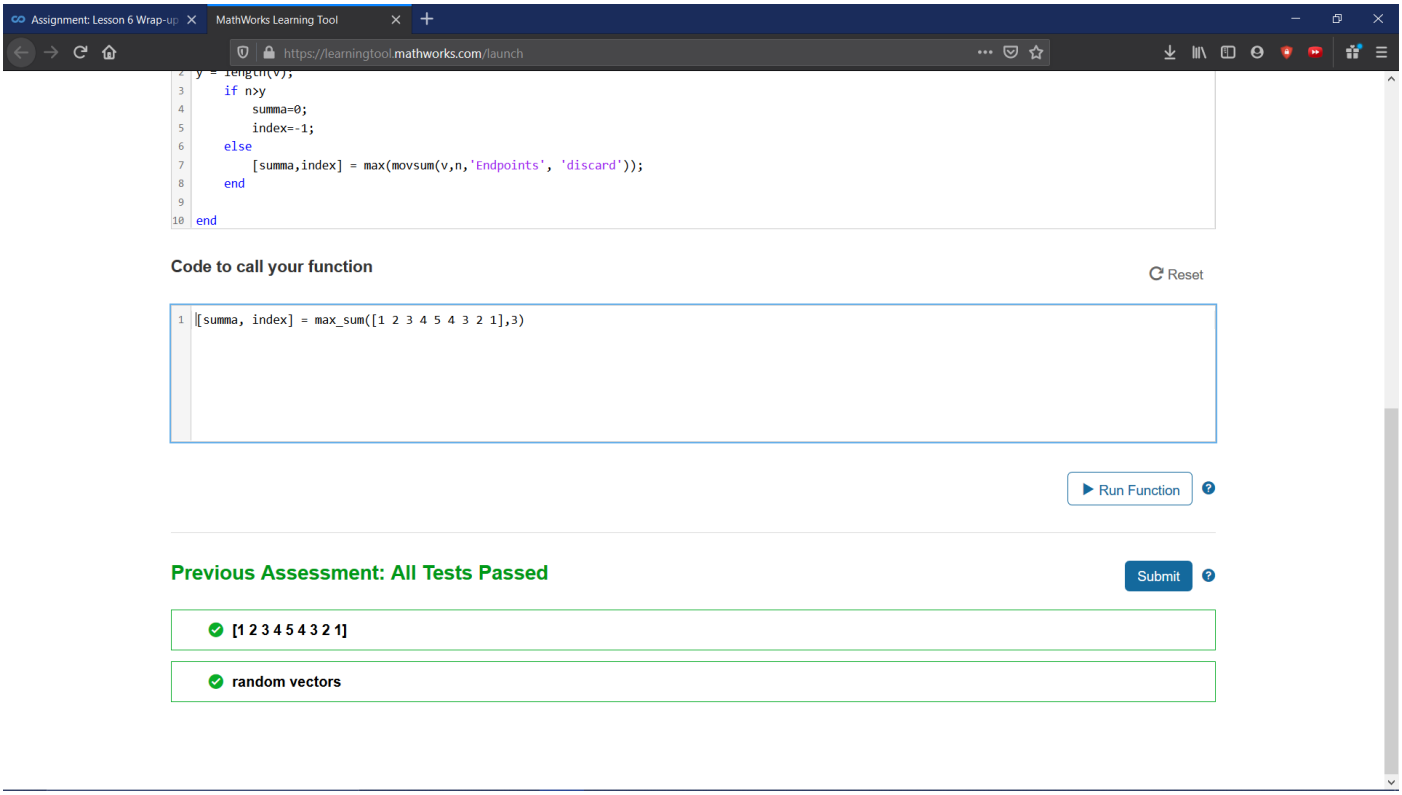
Write a function called **max_sum** that takes **v**, a row vector of numbers, and **n**, a positive integer as inputs. The function needs to find the **n** consecutive elements of **v** whose sum is the largest possible. In other words, if **v** is **[1 2 3 4 5 4 3 2 1]** and **n** is 3, it will find 4 5 and 4 because their sum of 13 is the largest of any 3 consecutive elements of **v**. If multiple such sequences exist in **v**, **max_sum** returns the first one. The function returns **summa**, the sum as the first output argument and **index**, the index of the first element of the **n** consecutive ones as the second output. If the input **n** is larger than the number of elements of **v**, the function returns 0 as the sum and -1 as the index. Here are a few example runs:

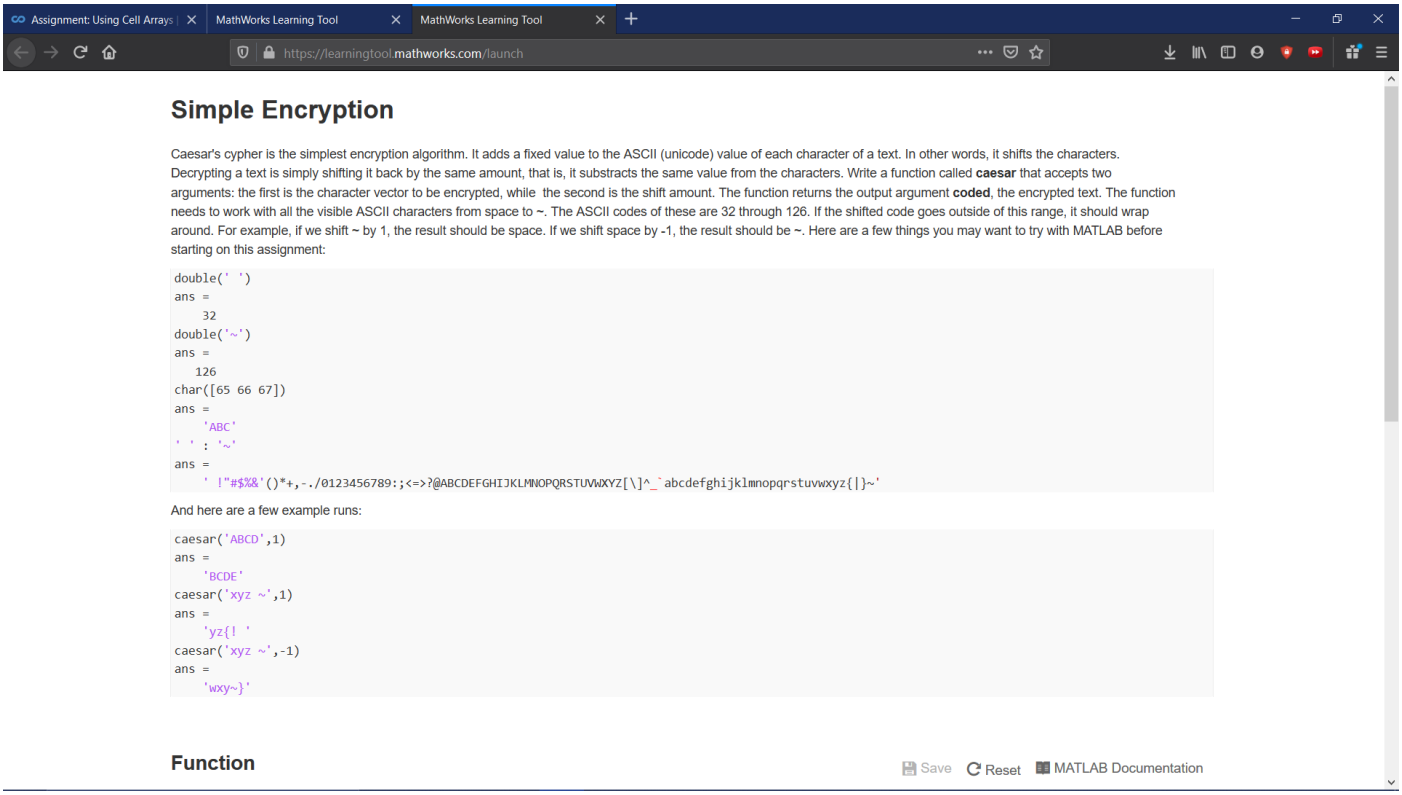
```
[summa, index] = max_sum([1 2 3 4 5 4 3 2 1],3)
    summa = 13
    index = 4
[summa, index] = max_sum([1 2 3 4 5 4 3 2 1],2)
    summa = 9
    index = 4
[summa, index] = max_sum([1 2 3 4 5 4 3 2 1],1)
    summa = 5
    index = 5
[summa, index] = max_sum([1 2 3 4 5 4 3 2 1],9)
    summa = 25
    index = 1
[summa, index] = max_sum([1 2 3 4 5 4 3 2 1],10)
    summa = 0
    index = -1
```

Function

Save Reset MATLAB Documentation

```
function [summa,index] = max_sum(v,n)
y = length(v);
    if n>y
        summa=0;
        index=-1;
    else
        [summa,index] = max(movsum(v,n,'Endpoints', 'discard'));
    end

end
```


Assignment: Lesson 6 Wrap-up
MathWorks Learning Tool
https://learningtool.mathworks.com/launch
y = length(v);
if n>y
summa=0;
index=-1;
else
[summa,index] = max(movsum(v,n,'Endpoints', 'discard'));
end
end
Code to call your function
Reset
[summa, index] = max_sum([1 2 3 4 5 4 3 2 1],3)
Run Function
Previous Assessment: All Tests Passed
Submit
[1 2 3 4 5 4 3 2 1]
random vectors



Simple Encryption

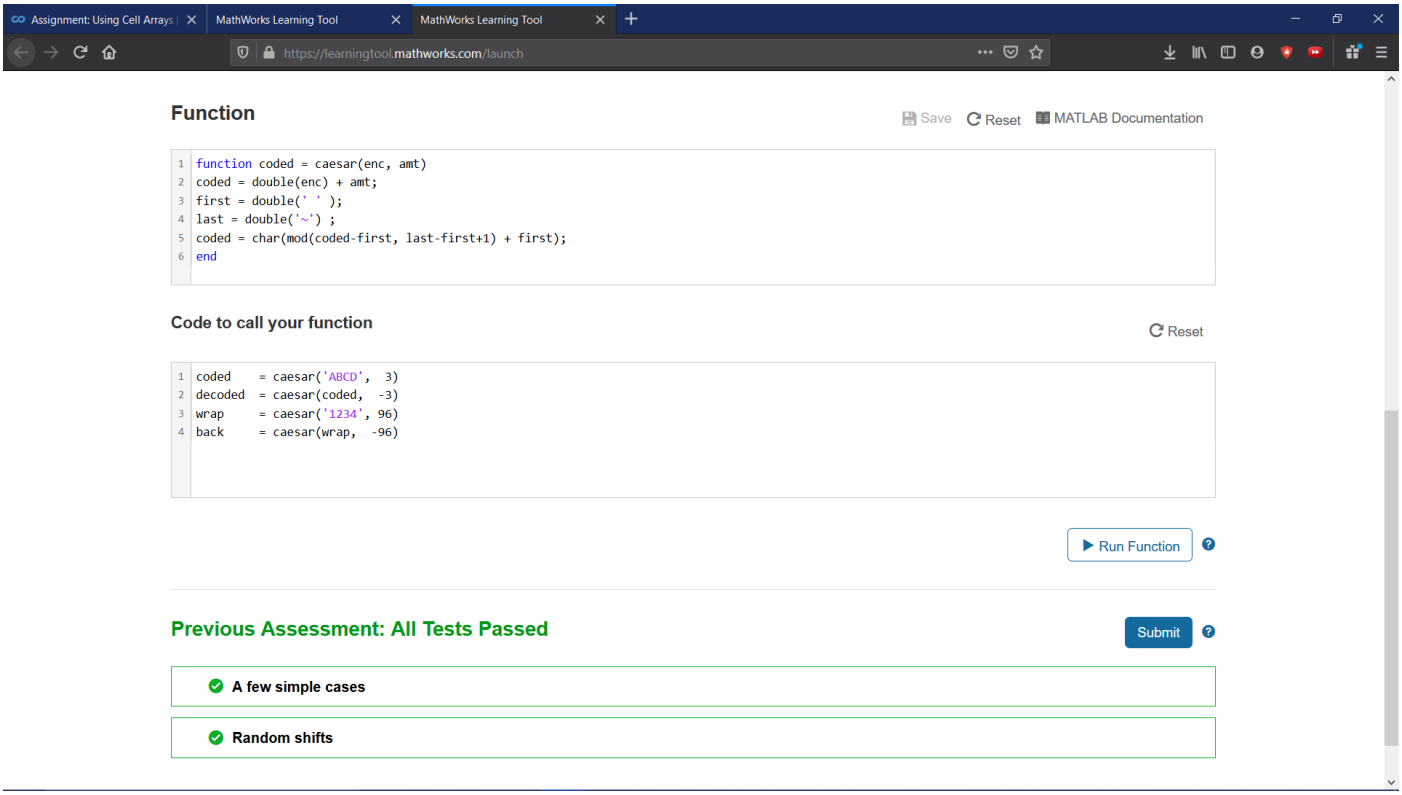
Caesar's cypher is the simplest encryption algorithm. It adds a fixed value to the ASCII (unicode) value of each character of a text. In other words, it shifts the characters. Decrypting a text is simply shifting it back by the same amount, that is, it substracts the same value from the characters. Write a function called **caesar** that accepts two arguments: the first is the character vector to be encrypted, while the second is the shift amount. The function returns the output argument **coded**, the encrypted text. The function needs to work with all the visible ASCII characters from space to ~. The ASCII codes of these are 32 through 126. If the shifted code goes outside of this range, it should wrap around. For example, if we shift ~ by 1, the result should be space. If we shift space by -1, the result should be ~. Here are a few things you may want to try with MATLAB before starting on this assignment:

```
double(' ')
ans =
    32
double('~')
ans =
   126
char([65 66 67])
ans =
    'ABC'
' ' : '~'
ans =
    ' !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxyz{|}~'
```

And here are a few example runs:

```
caesar('ABCD',1)
ans =
    'BCDE'
caesar('xyz ~',1)
ans =
    'yz{! '
caesar('xyz ~',-1)
ans =
    'wxy~}'
```

Function


Assignment: Using Cell Arrays
MathWorks Learning Tool
MathWorks Learning Tool
https://learningtool.mathworks.com/launch
Function
Save
Reset
MATLAB Documentation
function coded = caesar(enc, amt)
coded = double(enc) + amt;
first = double(' ');
last = double('~') ;
coded = char(mod(coded-first, last-first+1) + first);
end
Code to call your function
Reset
coded = caesar('ABCD', 3)
decoded = caesar(coded, -3)
wrap = caesar('1234', 96)
back = caesar(wrap, -96)
Run Function
Previous Assessment: All Tests Passed
Submit
A few simple cases
Random shifts

Sparse Matrix

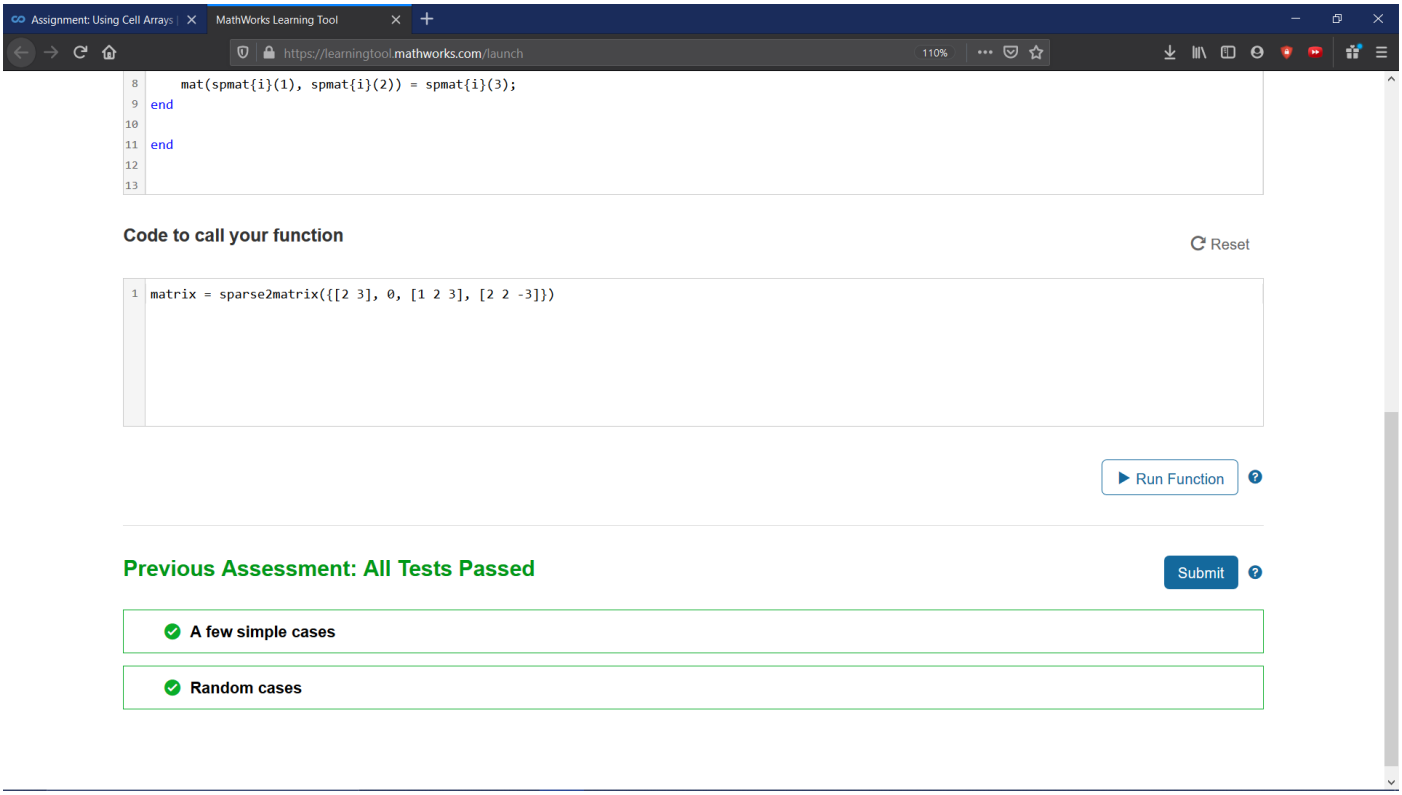
A sparse matrix is a large matrix with almost all elements of the same value (typically zero). The normal representation of a sparse matrix takes up lots of memory when the useful information can be captured with much less. A possible way to represent a sparse matrix is with a cell vector whose first element is a 2-element vector representing the size of the sparse matrix. The second element is a scalar specifying the default value of the sparse matrix. Each successive element of the cell vector is a 3-element vector representing one element of the sparse matrix that has a value other than the default. The three elements are the row index, the column index and the actual value. Write a function called **sparse2matrix** that takes a single input of a cell vector as defined above and returns the output argument called **matrix**, the matrix in its traditional form. Consider the following run:

```
cellvec = {[2 3], 0, [1 2 3], [2 2 -3]};
matrix = sparse2matrix(cellvec)

matrix =

     0     3     0
     0    -3     0
```

Function

Save Reset MATLAB Documentation

```
function mat = sparse2matrix(spmat)
matA = spmat{1};
matB = spmat{2};
mat(matA(1),matA(2))= matB;
mat(:,:) = matB;
s = size(spmat);
for i = 3:s(2)
    mat(spmat{i}(1), spmat{i}(2)) = spmat{i}(3);
end

end

```


Assignment: Using Cell Arrays
MathWorks Learning Tool
https://learningtool.mathworks.com/launch
110%
mat(spmat{i}(1), spmat{i}(2)) = spmat{i}(3);
end
end
Code to call your function
Reset
matrix = sparse2matrix({[2 3], 0, [1 2 3], [2 2 -3]})
Run Function
Previous Assessment: All Tests Passed
Submit
A few simple cases
Random cases

Excel File I/O

The attached Distances.xlsx file contains a spreadsheet with the pairwise distances in miles of the top 100 US cities by population. A preview of the spreadsheet is shown below. The first row and first column contain the city names using the following format: city name comma space capitalized state abbreviation, e.g., Nashville, TN. Note that the very first cell of the spresheet, A1, is blank.

Write a function called **get_distance** that accepts two character vector inputs representing the names of two cities. The function returns the distance between them as an output argument called **distance**. For example, the call **get_distance('Seattle, WA','Miami, FL')** should return 3723. If one or both of the specified cities are not in the file, the function returns -1.

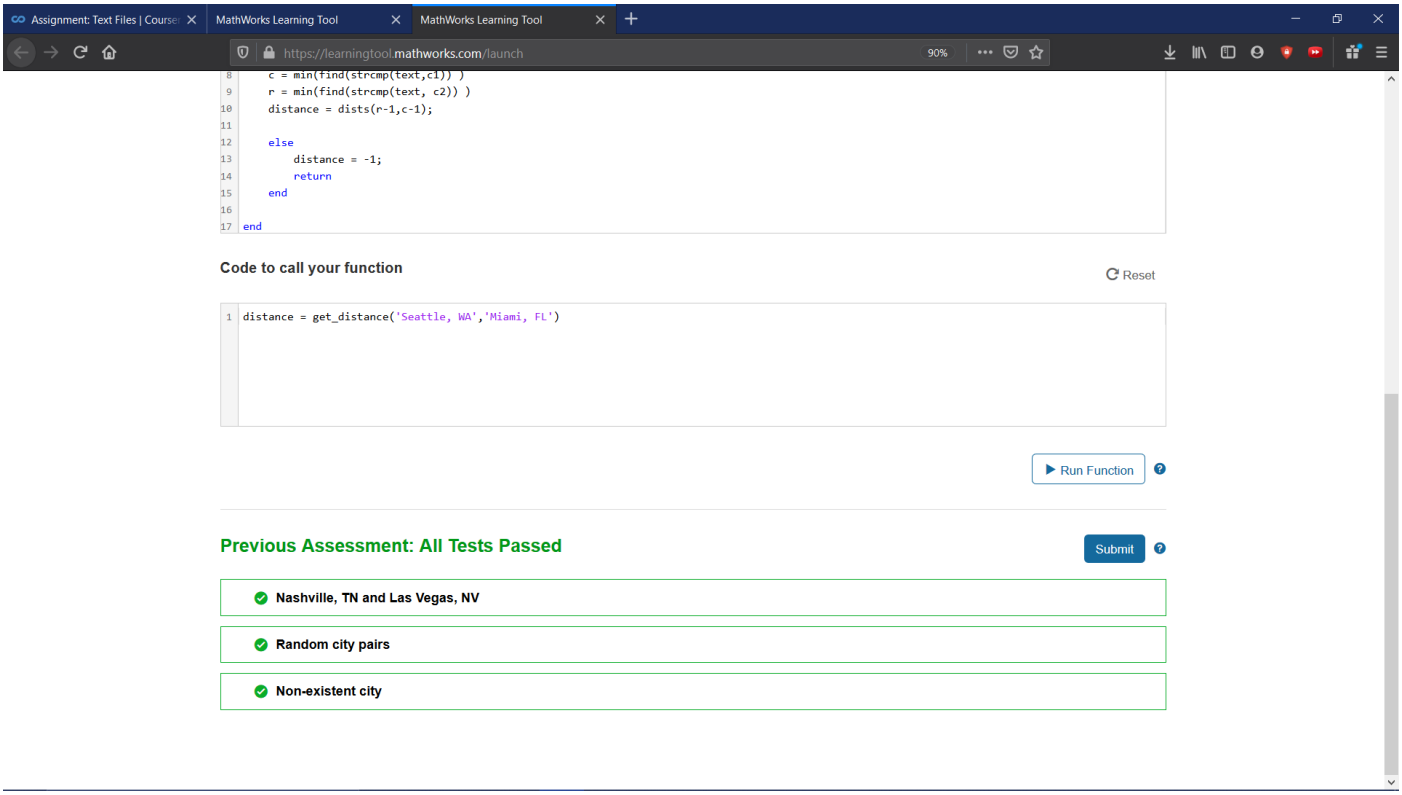
Preview of the first five cities of *Distances.xlsx*

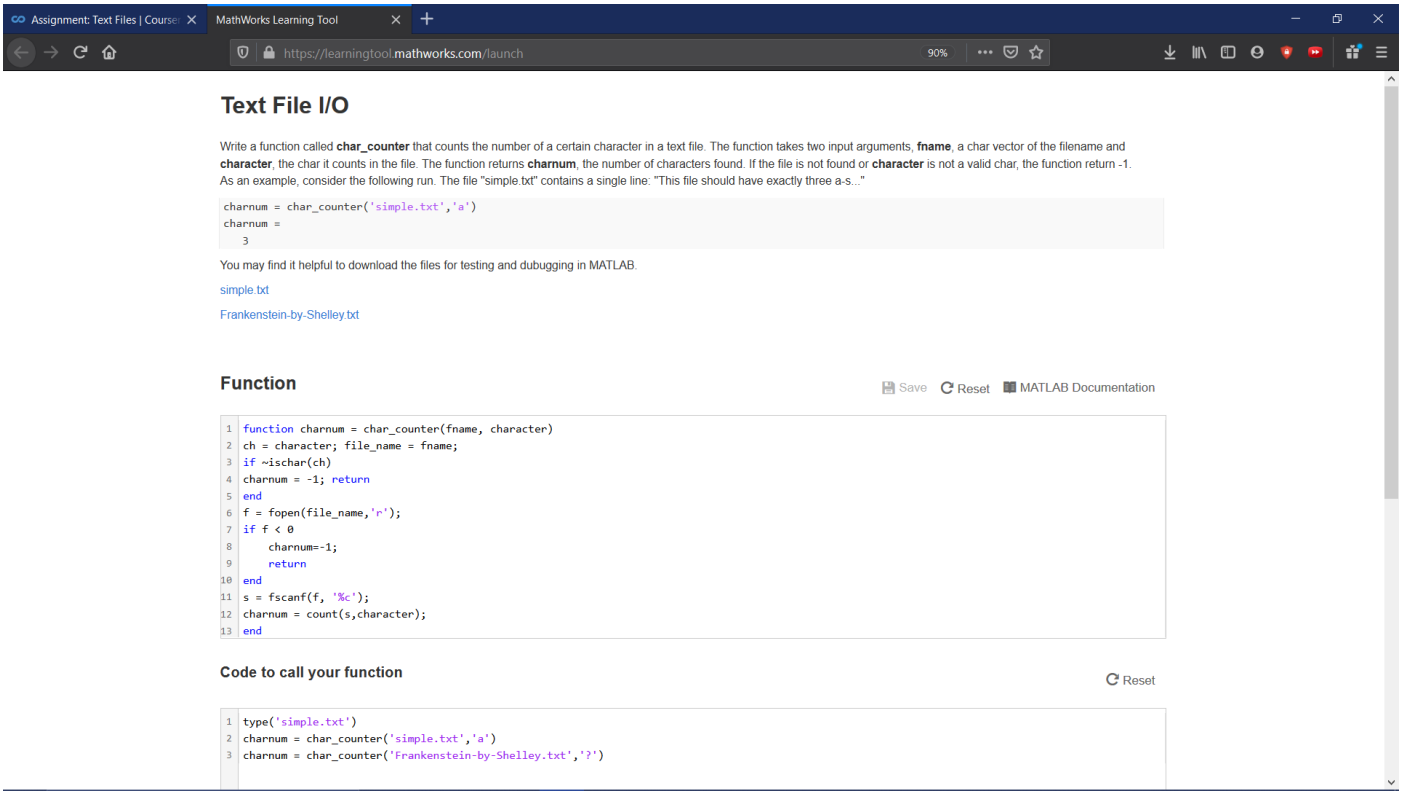
	A	B	C	D	E	F
1		Abilene, TX	Akron, OH	Albuquerque, NM	Alexandria, VA	Allentown, PA
2	Abilene, TX	0	1481	724	1615	1730
3	Akron, OH	1481	0	2185	382	552
4	Albuquerque, NM	724	2185	0	2331	2447
5	Alexandria, VA	1615	382	2331	0	195
6	Allentown, PA	1730	552	2447	195	0

Function

Save Reset MATLAB Documentation

```
function distance = get_distance(c1, c2)

[dists, text] = xlsread('Distances.xlsx');
SUM1 = sum(ismember(c1, text))
SUM2 = sum(ismember(c2, text))
    if SUM1 == length(ismember(c1, text)) && SUM2 == length(ismember(c2, text))

    c = min(find(strcmp(text,c1)) )
    r = min(find(strcmp(text, c2)) )
    distance = dists(r-1,c-1);

    else
        distance = -1;
        return
    end

end
```


Assignment: Text Files | Course
MathWorks Learning Tool
MathWorks Learning Tool
https://learningtool.mathworks.com/launch
90%
c = min(find(strcmp(text,c1)))
r = min(find(strcmp(text, c2)))
distance = dists(r-1,c-1);
else
distance = -1;
return
end
end
Code to call your function
Reset
distance = get_distance('Seattle, WA','Miami, FL')
Run Function
Previous Assessment: All Tests Passed
Submit
Nashville, TN and Las Vegas, NV
Random city pairs
Non-existent city



Text File I/O

Write a function called **char_counter** that counts the number of a certain character in a text file. The function takes two input arguments, **fname**, a char vector of the filename and **character**, the char it counts in the file. The function returns **charnum**, the number of characters found. If the file is not found or **character** is not a valid char, the function return -1. As an example, consider the following run. The file "simple.txt" contains a single line: "This file should have exactly three a-s..."

```
charnum = char_counter('simple.txt','a')
charnum =
   3
```

You may find it helpful to download the files for testing and dubugging in MATLAB.

simple.txt

Frankenstein-by-Shelley.txt

Function

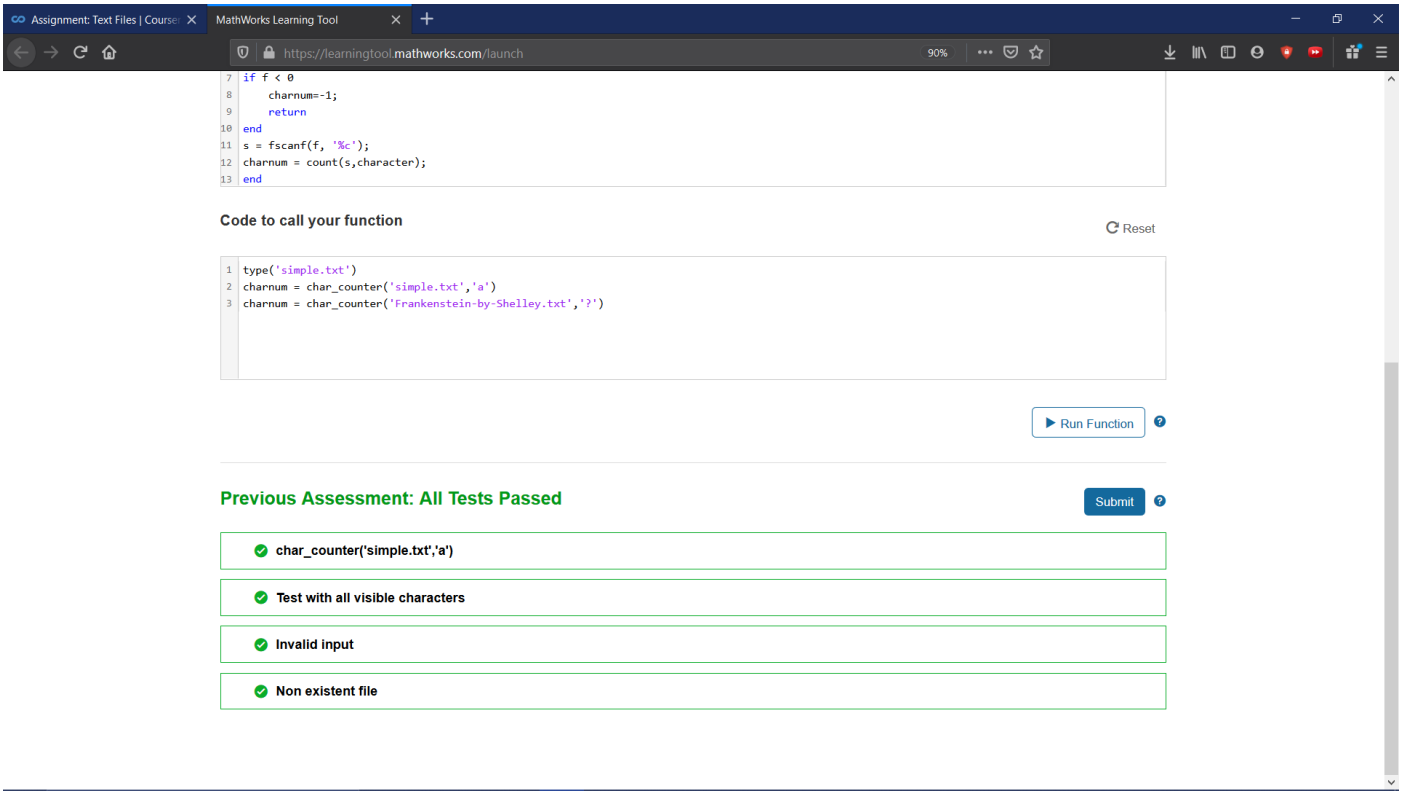
Save Reset MATLAB Documentation

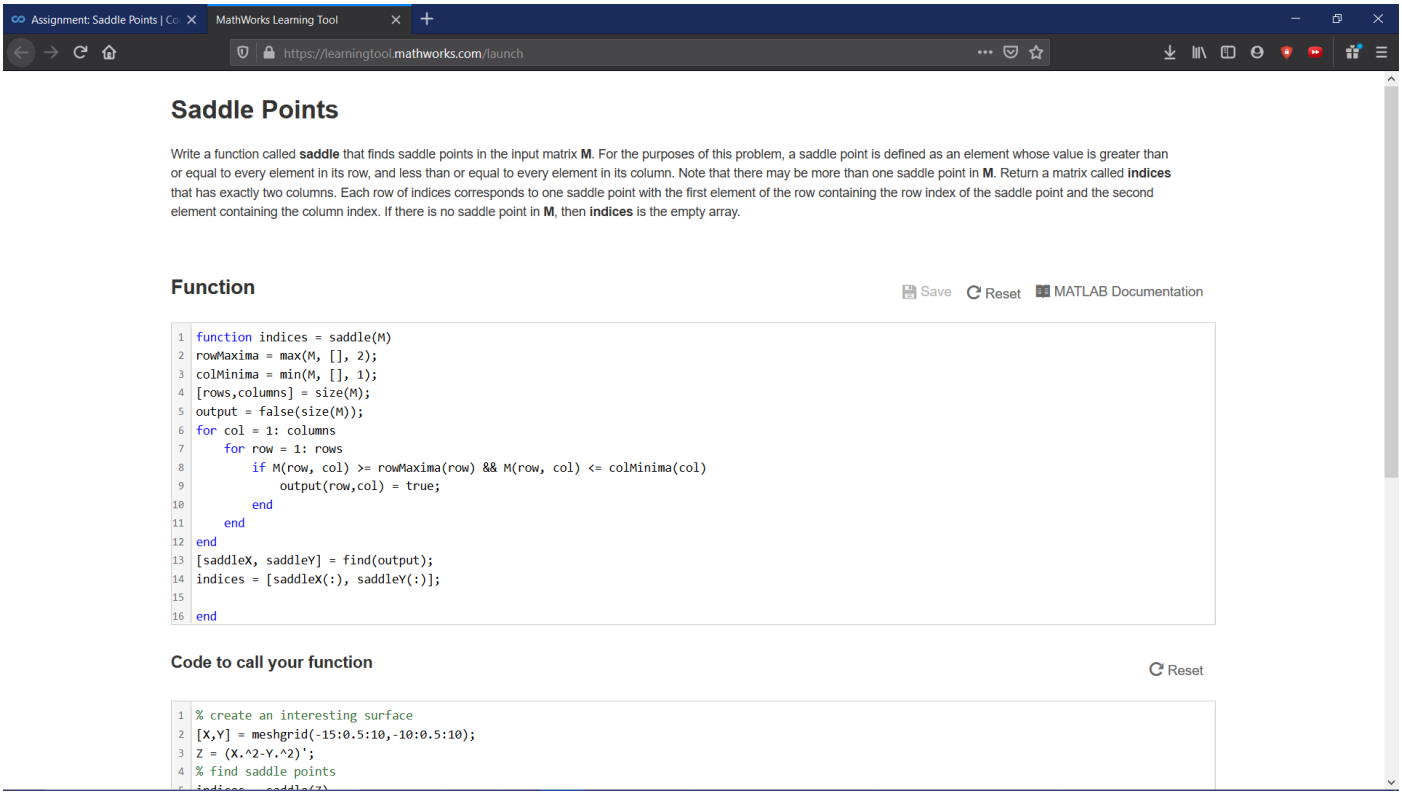
```
function charnum = char_counter(fname, character)
ch = character; file_name = fname;
if ~ischar(ch)
charnum = -1; return
end
f = fopen(file_name,'r');
if f < 0
    charnum=-1;
    return
end
s = fscanf(f, '%c');
charnum = count(s,character);
end
```

Code to call your function

Reset

```
type('simple.txt')
charnum = char_counter('simple.txt','a')
charnum = char_counter('Frankenstein-by-Shelley.txt','?')
```


Assignment: Text Files | Course
MathWorks Learning Tool
https://learningtool.mathworks.com/launch
90%
if f < 0
charnum=-1;
return
end
s = fscanf(f, '%c');
charnum = count(s,character);
end
Code to call your function
Reset
type('simple.txt')
charnum = char_counter('simple.txt','a')
charnum = char_counter('Frankenstein-by-Shelley.txt','?')
Run Function
Previous Assessment: All Tests Passed
Submit
char_counter('simple.txt','a')
Test with all visible characters
Invalid input
Non existent file



Saddle Points

Write a function called **saddle** that finds saddle points in the input matrix **M**. For the purposes of this problem, a saddle point is defined as an element whose value is greater than or equal to every element in its row, and less than or equal to every element in its column. Note that there may be more than one saddle point in **M**. Return a matrix called **indices** that has exactly two columns. Each row of indices corresponds to one saddle point with the first element of the row containing the row index of the saddle point and the second element containing the column index. If there is no saddle point in **M**, then **indices** is the empty array.

Function

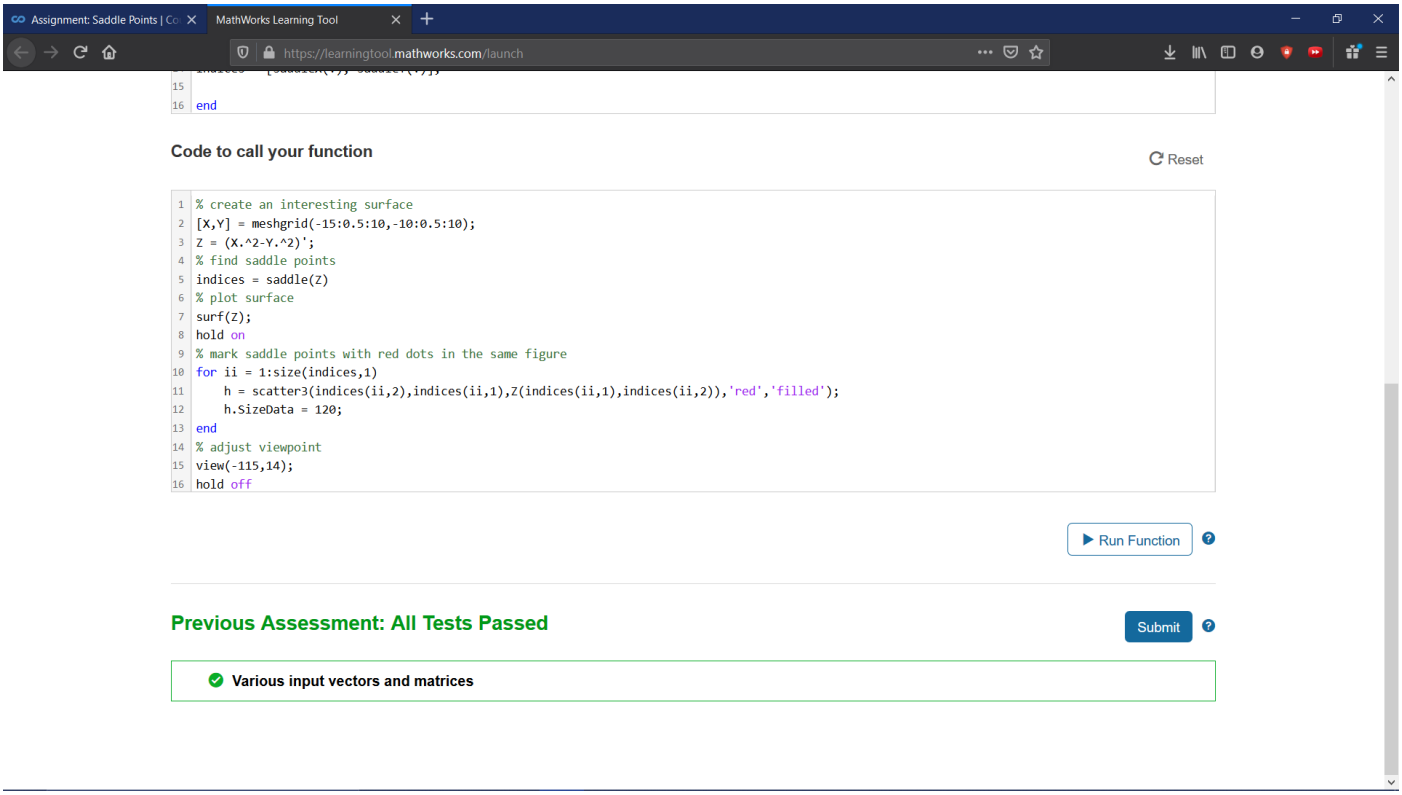
Save Reset MATLAB Documentation

```
function indices = saddle(M)
rowMaxima = max(M, [], 2);
colMinima = min(M, [], 1);
[rows,columns] = size(M);
output = false(size(M));
for col = 1: columns
    for row = 1: rows
        if M(row, col) >= rowMaxima(row) && M(row, col) <= colMinima(col)
            output(row,col) = true;
        end
    end
end
[saddleX, saddleY] = find(output);
indices = [saddleX(:), saddleY(:)];

end
```

Code to call your function

Reset

```
% create an interesting surface
[X,Y] = meshgrid(-15:0.5:10,-10:0.5:10);
Z = (X.^2-Y.^2)';
% find saddle points
```


Assignment: Saddle Points | Co
MathWorks Learning Tool
https://learningtool.mathworks.com/launch
end
Code to call your function
Reset
% create an interesting surface
[X,Y] = meshgrid(-15:0.5:10,-10:0.5:10);
Z = (X.^2-Y.^2)';
% find saddle points
indices = saddle(Z)
% plot surface
surf(Z);
hold on
% mark saddle points with red dots in the same figure
for ii = 1:size(indices,1)
h = scatter3(indices(ii,2),indices(ii,1),Z(indices(ii,1),indices(ii,2)),'red','filled');
h.SizeData = 120;
end
% adjust viewpoint
view(-115,14);
hold off
Run Function
Previous Assessment: All Tests Passed
Submit
Various input vectors and matrices

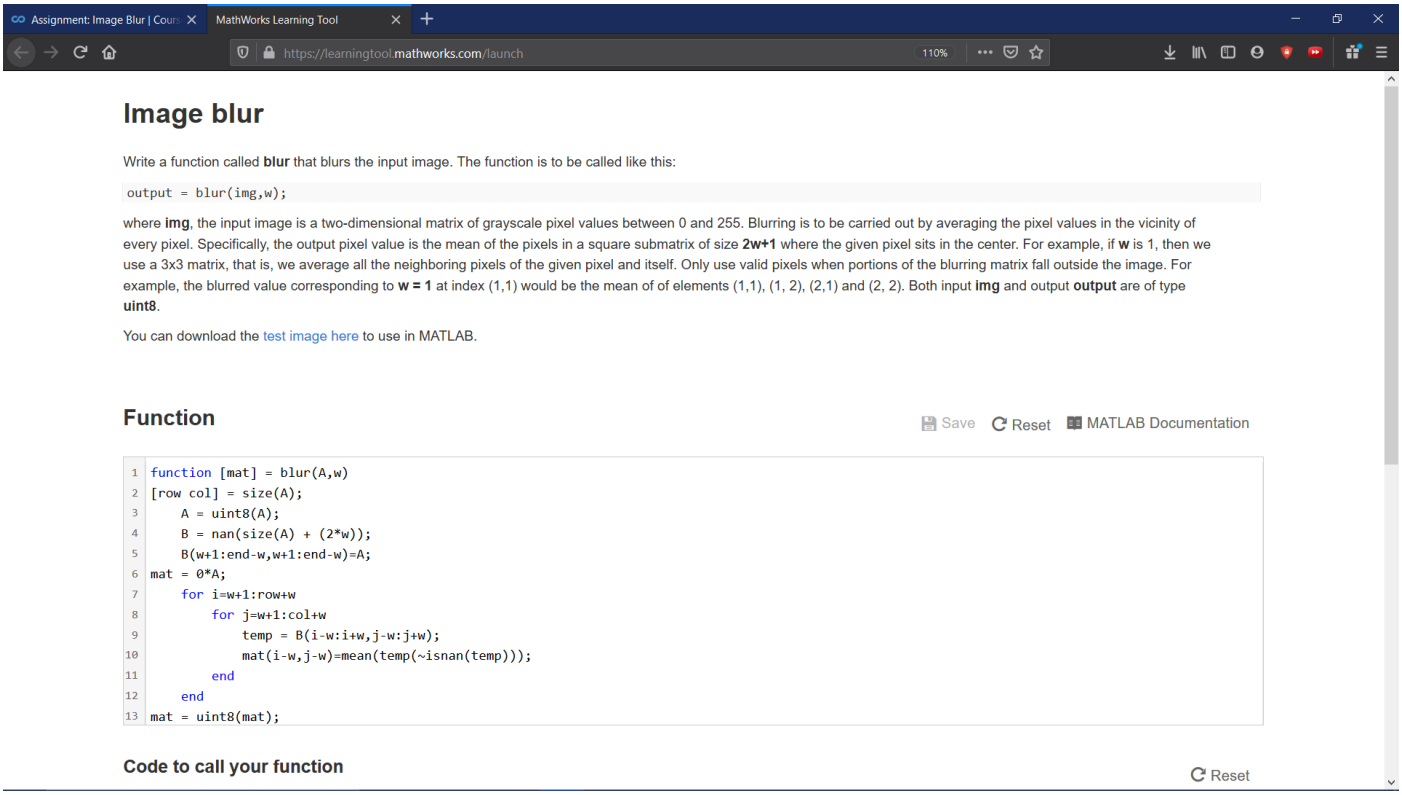


Image blur

Write a function called **blur** that blurs the input image. The function is to be called like this:

```
output = blur(img,w);
```

where **img**, the input image is a two-dimensional matrix of grayscale pixel values between 0 and 255. Blurring is to be carried out by averaging the pixel values in the vicinity of every pixel. Specifically, the output pixel value is the mean of the pixels in a square submatrix of size **2w+1** where the given pixel sits in the center. For example, if **w** is 1, then we use a 3x3 matrix, that is, we average all the neighboring pixels of the given pixel and itself. Only use valid pixels when portions of the blurring matrix fall outside the image. For example, the blurred value corresponding to **w = 1** at index (1,1) would be the mean of of elements (1,1), (1, 2), (2,1) and (2, 2). Both input **img** and output **output** are of type **uint8**.

You can download the test image here to use in MATLAB.

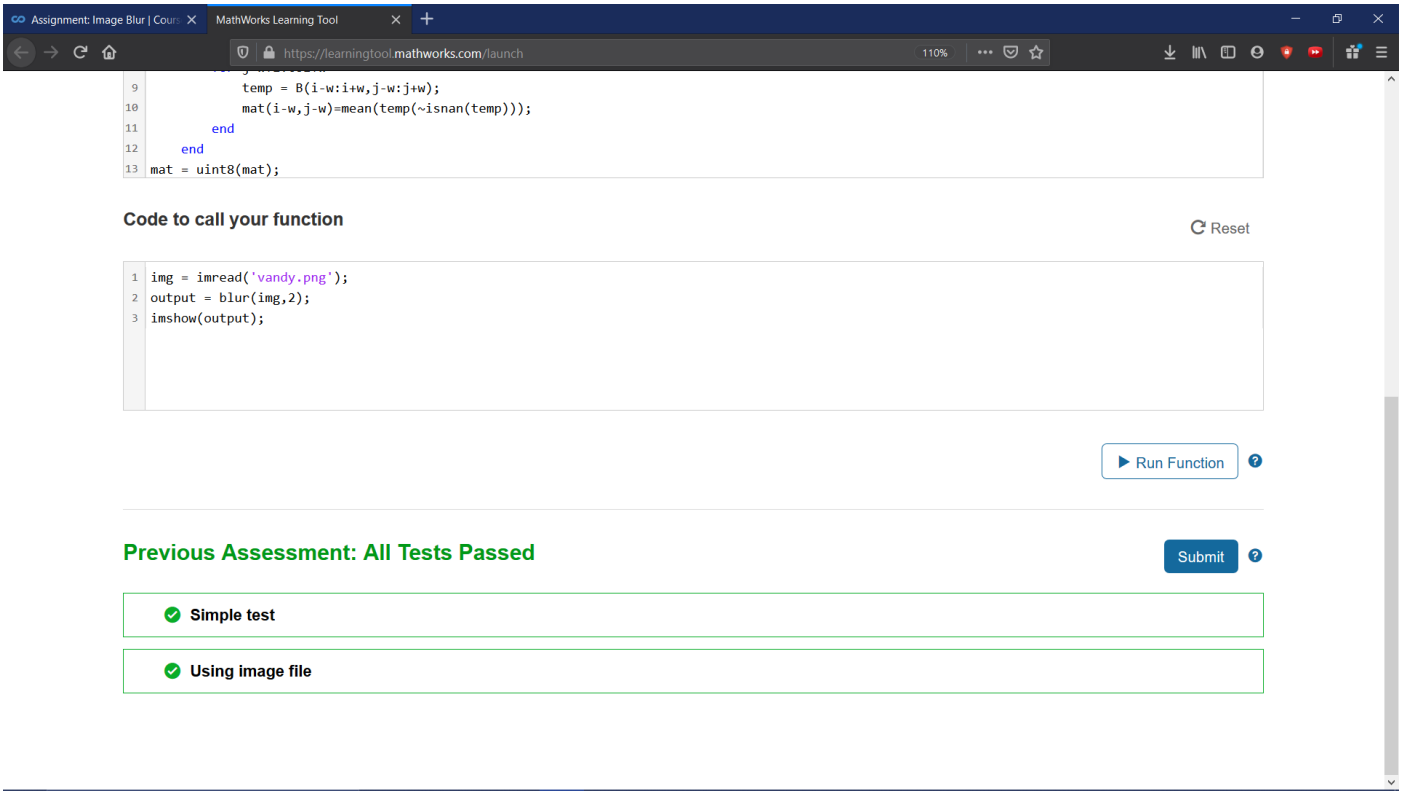
Function

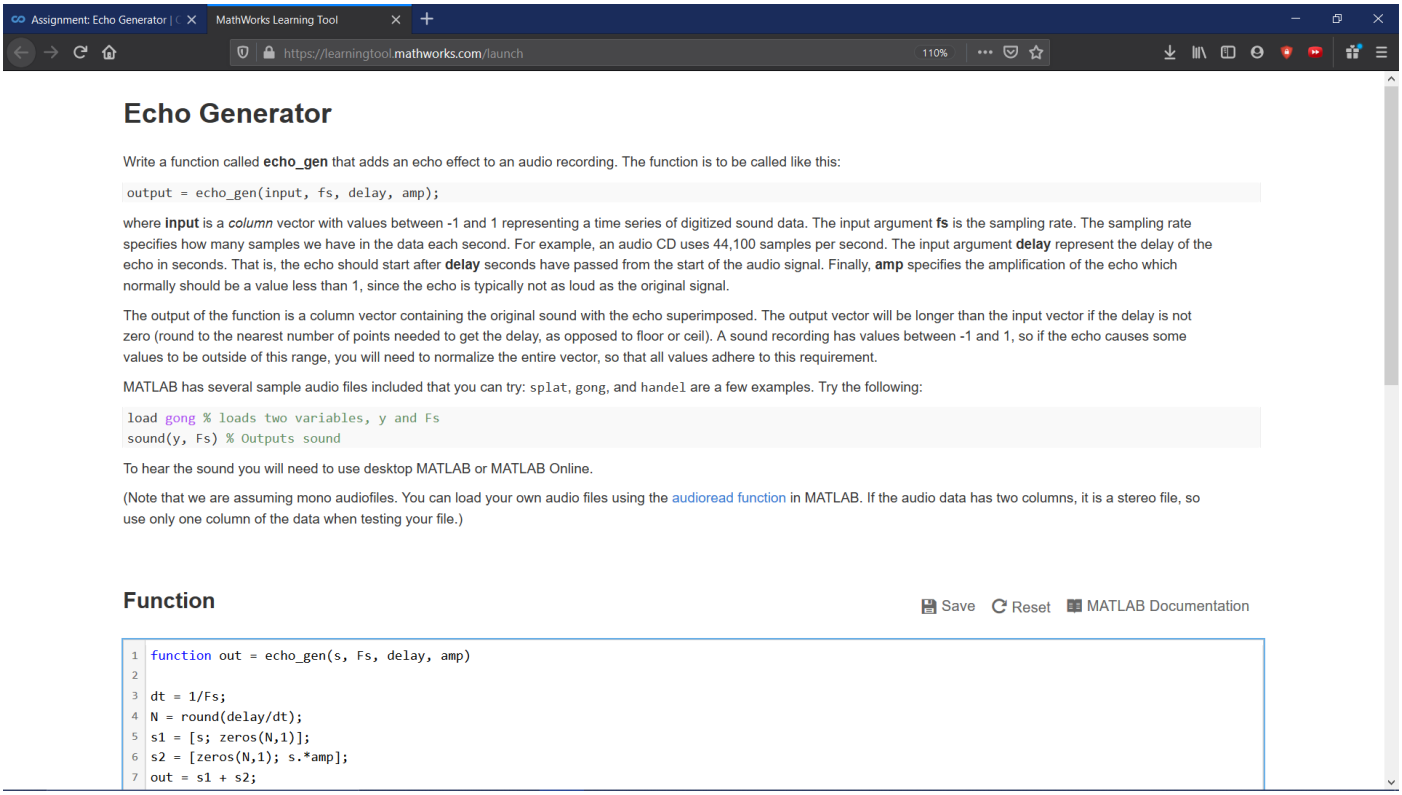
Save Reset MATLAB Documentation

```
function [mat] = blur(A,w)
[row col] = size(A);
    A = uint8(A);
    B = nan(size(A) + (2*w));
    B(w+1:end-w,w+1:end-w)=A;
mat = 0*A;
    for i=w+1:row+w
        for j=w+1:col+w
            temp = B(i-w:i+w,j-w:j+w);
            mat(i-w,j-w)=mean(temp(~isnan(temp)));
        end
    end
mat = uint8(mat);
```

Code to call your function

Reset


Assignment: Image Blur | Cours
MathWorks Learning Tool
https://learningtool.mathworks.com/launch
110%
temp = B(i-w:i+w,j-w:j+w);
mat(i-w,j-w)=mean(temp(~isnan(temp)));
end
end
mat = uint8(mat);
Code to call your function
Reset
img = imread('vandy.png');
output = blur(img,2);
imshow(output);
Run Function
Previous Assessment: All Tests Passed
Submit
Simple test
Using image file



Echo Generator

Write a function called **echo_gen** that adds an echo effect to an audio recording. The function is to be called like this:

```
output = echo_gen(input, fs, delay, amp);
```

where **input** is a *column* vector with values between -1 and 1 representing a time series of digitized sound data. The input argument **fs** is the sampling rate. The sampling rate specifies how many samples we have in the data each second. For example, an audio CD uses 44,100 samples per second. The input argument **delay** represent the delay of the echo in seconds. That is, the echo should start after **delay** seconds have passed from the start of the audio signal. Finally, **amp** specifies the amplification of the echo which normally should be a value less than 1, since the echo is typically not as loud as the original signal.

The output of the function is a column vector containing the original sound with the echo superimposed. The output vector will be longer than the input vector if the delay is not zero (round to the nearest number of points needed to get the delay, as opposed to floor or ceil). A sound recording has values between -1 and 1, so if the echo causes some values to be outside of this range, you will need to normalize the entire vector, so that all values adhere to this requirement.

MATLAB has several sample audio files included that you can try: `splat`, `gong`, and `handel` are a few examples. Try the following:

```
load gong % loads two variables, y and Fs
sound(y, Fs) % Outputs sound
```

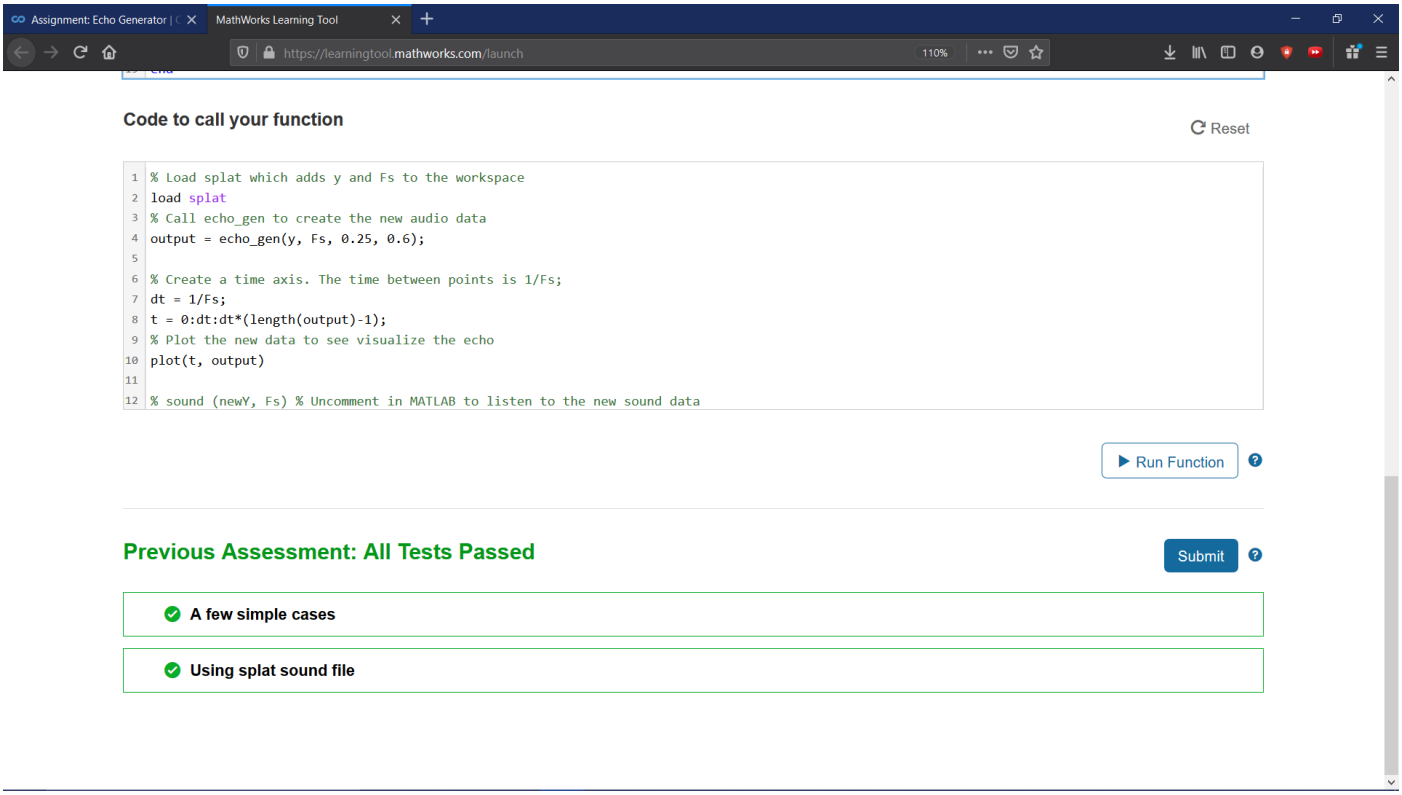
To hear the sound you will need to use desktop MATLAB or MATLAB Online.

(Note that we are assuming mono audiofiles. You can load your own audio files using the audioread function in MATLAB. If the audio data has two columns, it is a stereo file, so use only one column of the data when testing your file.)

Function

Save Reset MATLAB Documentation

```
function out = echo_gen(s, Fs, delay, amp)

dt = 1/Fs;
N = round(delay/dt);
s1 = [s; zeros(N,1)];
s2 = [zeros(N,1); s.*amp];
out = s1 + s2;
```


Assignment: Echo Generator
MathWorks Learning Tool
https://learningtool.mathworks.com/launch
110%
end
Code to call your function
Reset
1 % Load splat which adds y and Fs to the workspace
2 load splat
3 % Call echo_gen to create the new audio data
4 output = echo_gen(y, Fs, 0.25, 0.6);
5
6 % Create a time axis. The time between points is 1/Fs;
7 dt = 1/Fs;
8 t = 0:dt:dt*(length(output)-1);
9 % Plot the new data to see visualize the echo
10 plot(t, output)
11
12 % sound (newY, Fs) % Uncomment in MATLAB to listen to the new sound data
Run Function
Previous Assessment: All Tests Passed
Submit
A few simple cases
Using splat sound file