

# ECE 449/590

## Project 01

### Numpy and EasyNN

Paolo Orguim

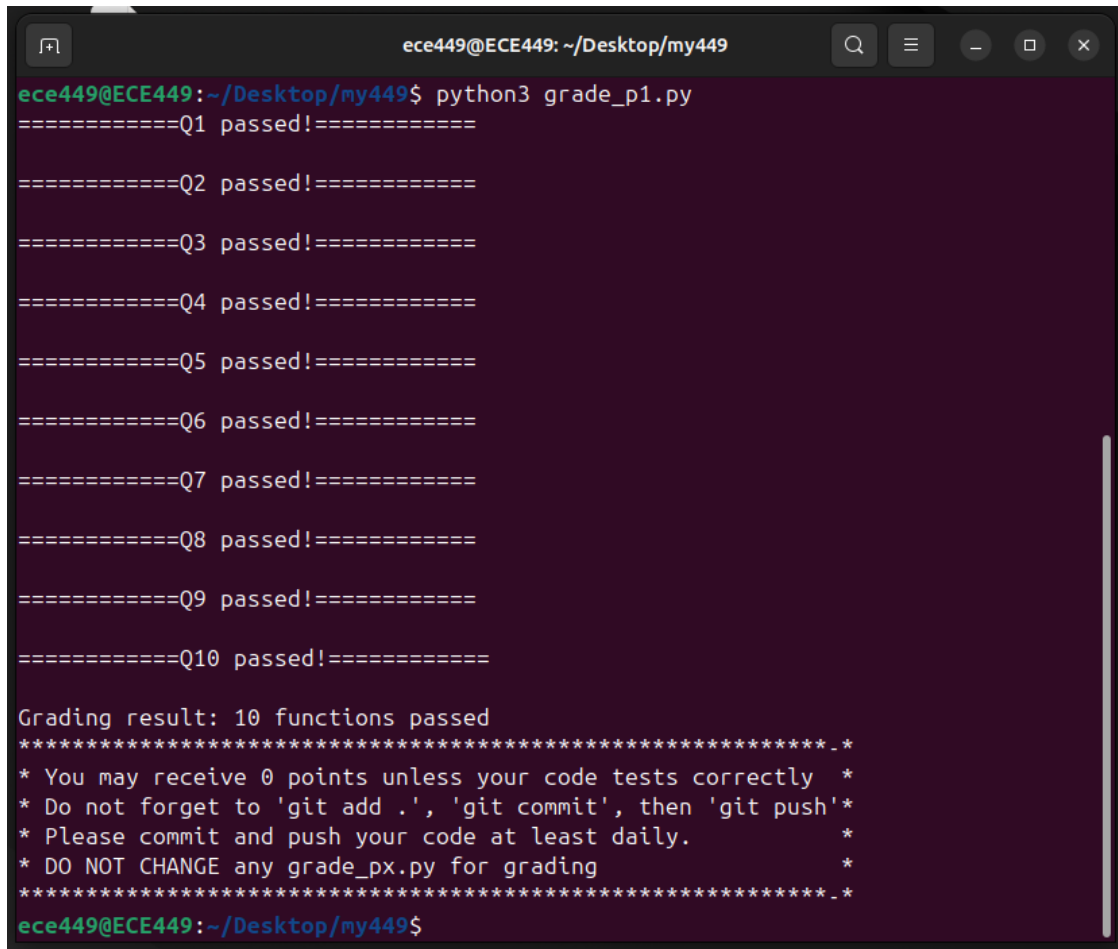
Due: Sunday, September 7<sup>th</sup>, 2025, 11:59 PM

#### Abstract

This report documents the implementation of Project 01. It includes screenshots of the grading outputs and explanations of each implemented function.

## 1 Screenshots

This section contains the required screenshots of the VM output: the grading test case results and the git push results.



```
ece449@ECE449: ~/Desktop/my449
ece449@ECE449:~/Desktop/my449$ python3 grade_p1.py
=====Q1 passed!=====
=====Q2 passed!=====
=====Q3 passed!=====
=====Q4 passed!=====
=====Q5 passed!=====
=====Q6 passed!=====
=====Q7 passed!=====
=====Q8 passed!=====
=====Q9 passed!=====
=====Q10 passed!=====

Grading result: 10 functions passed
*****
* You may receive 0 points unless your code tests correctly *
* Do not forget to 'git add .', 'git commit', then 'git push'*
* Please commit and push your code at least daily.          *
* DO NOT CHANGE any grade_px.py for grading                 *
*****
ece449@ECE449:~/Desktop/my449$
```

Figure 1: Result from running `grade_p1.py` locally.

```
ece449@ECE449: ~/Desktop/my449
ece449@ECE449:~/Desktop/my449$ git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 315 bytes | 315.00 KiB/s, done.
Total 3 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Cloning into 'psimioni_449fa25'...
remote: Your Latest Submission for
remote: =====
remote: Project p1
remote: Submitted Time: 20250907200926
remote: Q1: pass
remote: Q2: pass
remote: Q3: pass
remote: Q4: pass
remote: Q5: pass
remote: Q6: pass
remote: Q7: pass
remote: Q8: pass
remote: Q9: pass
remote: Q10: pass
remote: Your submitted code passes 10 functions on Dasan
remote: =====
remote: Dasan Git Repo has successfully received your code.
remote: Make sure to 'git add .' to include all files of your project.
remote: Otherwise, your project submission will not be graded properly.
remote: =====
remote: please wait..system processing...
remote: done!
To dasan.ece.iit.edu:psimioni_449fa25
   0c03175..711bb7f  master -> master
ece449@ECE449:~/Desktop/my449$
```

Figure 2: Result after pushing the changes to the repository.

## 2 Function Explanations

This section briefly describes the functions implemented in the project. Each function's code is shown below.

### 2.1 Function 1

Create a numpy array of 10 rows and 5 columns; set the element at row  $i$  and column  $j$  to be  $i + j$ . Number of rows and columns are set as variables and the array is initialized as an empty array. The nested for loops fill up the array with the corresponding values based on the indices of the current row and column.

```
1 def Q1():
2     rows = 10
3     cols = 5
4
5     # Create uninitialized numpy array as per numpy.org documentation
6     array = np.empty([rows, cols], dtype = int) # Empty array
7
8     for i in range(rows):
9         for j in range(cols):
10             array[i, j] = i + j # The indices start at 0 it seems
11
12     return array
```

### 2.2 Function 2

Add two numpy arrays together.  
Simply summing the array with the plus sign operator is enough.

```
1 def Q2(a, b):
2     return a + b
```

### 2.3 Function 3

Multiply two 2D numpy arrays using matrix multiplication.  
The @ operator from python does matrix multiplication, so nothing else is required here.

```
1 def Q3(a, b):
2     # @ operator to multiply the matrices
3     return a @ b
```

### 2.4 Function 4

For each row of a 2D numpy array, find the column index with the maximum element.  
As per the docs referenced in the code, the argmax function can give the desired output correctly.

```
1 def Q4(a):
2     # From:
3     # docs.vultr.com/python/third-party/numpy/argmax
4     return np.argmax(a, axis = 1)
```

### 2.5 Function 5

Solve the following expression:  $Ax = b$  for  $x$ .  
The dot call with the two given arguments solves the expression.

```

1 def Q5(A, b):
2     # Solving for x using algorithm found on this thread at stackoverflow:
3     # Matrix multiplication, solve Ax = b solve for x
4     return np.dot(np.linalg.inv(A), b)

```

## 2.6 Function 6

Return a simple EasyNN expression  $(a + b)$ .  
We just get both inputs and return their sum.

```

1 def Q6():
2     # [Lecture 3] EasyNN-1.pdf
3     a = nn.Input("a")
4     b = nn.Input("b")
5     return (a+b)

```

## 2.7 Function 7

Return an EasyNN expression for  $a + b \times c$ .  
We just get all three inputs and return the required expression.

```

1 def Q7():
2     # [Lecture 3] EasyNN-1.pdf
3     a = nn.Input("a")
4     b = nn.Input("b")
5     c = nn.Input("c")
6     return (a+b*c)

```

## 2.8 Function 8

Given  $A$  and  $b$ , return an EasyNN expression for  $Ax + b$ .  
 $A$  and  $b$  are constants and  $x$  is an input, we just return the required expression based on that.

```

1 def Q8(A, b):
2     A = nn.Const(A)
3     b = nn.Const(b)
4     x = nn.Input("x")
5     return (A * x + b)

```

## 2.9 Function 9

Given  $n$ , return an EasyNN expression for  $x^n$ .  
Assuming we will only be working with natural numbers, zero included, we just multiply the input as many times as necessary, returning 0 in case  $n == 0$

```

1 def Q9(n):
2     x = nn.Input("x")
3
4     if n == 0:
5         return 1
6
7     aux = x
8     for i in range(1, n):
9         aux = aux * x
10
11    return aux

```

## 2.10 Function 10

Return an EasyNN expression to compute the element-wise absolute value  $|x|$ .

If  $x$  is negative then  $ReLU(x)$  will be zero and  $ReLU(-x)$  will be  $-x$  which is positive

If  $x$  is positive then  $ReLU(x)$  will be  $x$  and  $ReLU(-x)$  will be zero.

In both cases, the sum  $ReLU(x) + ReLU(-x)$  will end up resulting in the absolute value of  $x$

```
1 def Q10():
2     relu = nn.ReLU()
3     x = nn.Input("x")
4     return (relu(x) + relu(-x)) # if x < 0 then relu(-x) = |x| else relu(x) = |x|
```