Copy !

An important concept for Part Four

You will:

- Explore what is copy in python and it's type.
- Explore how to implement it in C

Python - copy

Copy

- Is the module, which provides functions to create shallow and deep copies of objects, for creating a copy of a provided argument/variable.

Types of copy

python import

```
import copy
import numpy as np
```

✓ 0.0s Python

Shallow copy

A ***shallow copy*** in python creates a new object to reference the original nested object.

- If you modify new object, it will affect the new object as well.

```python
def main():
    #initialized parameter
    w_init = [[2],[3]]

    #perform shallow copy
    w = copy.copy(w_init)

    # Before modification of w
    print("\nBefore changinge shallow copied parameter, w \n")
    print(f"Original:  w_init[0][0]:   {w_init[0][0]},   w_init[1][0]:   {w_init[1][0]}")
    print(f"Shallow copy:  w[0][0]:   {w[0][0]},    w[1][0]:   {w[1][0]} \n")

    #modify the initialized parameter
    w[0][0] = 0
    w[1][0] = 0

    # After modification of w
    print("After changinge shallow copied parameter, w \n")
    print(f"Original:  w_init[0][0]:   {w_init[0][0]},   w_init[1][0]:   {w_init[1][0]}")
    print(f"Shallow copy:  w[0][0]:   {w[0][0]},    w[1][0]:   {w[1][0]}")

if __name__=="__main__":
    main()
```

✓ 0.0s Python

```
Before changinge shallow copied parameter, w

Original:  w_init[0][0]:   2,   w_init[1][0]:   3
Shallow copy:  w[0][0]:   2,    w[1][0]:   3

After changinge shallow copied parameter, w

Original:  w_init[0][0]:   0,   w_init[1][0]:   0
Shallow copy:  w[0][0]:   0,    w[1][0]:   0
```

Deep copy

A ***Deep copy*** in python creates a new object and ensuring no shared refrences with the original objects by recursively copies all objects found inside the original object.

- If you modify new object it will not affect the original object.

```python
def main():
    #initialized parameter
    w_init = [[2], [3]]

    #perform deep copy
    w = copy.deepcopy(w_init)

    # Before modification of w
    print("\nBefore changinge deep copied parameter, w \n")
    print(f"Original:  w_init[0][0]:   {w_init[0][0]},   w_init[1][0]:   {w_init[1][0]}")
    print(f"Shallow copy:  w[0][0]:   {w[0][0]},    w[1][0]:   {w[1][0]} \n")

    #modify the initialized parameter
    w[0][0] = 0
    w[1][0] = 0

    # After modification of w
    print("After changinge deep copied parameter, w \n")
    print(f"Original:  w_init[0][0]:   {w_init[0][0]},   w_init[1][0]:   {w_init[1][0]}")
    print(f"Shallow copy:  w[0][0]:   {w[0][0]},    w[1][0]:   {w[1][0]}")

if __name__=="__main__":
    main()
```

✓ 0.0s Python

```
Before changinge deep copied parameter, w

Original:  w_init[0][0]:   2,   w_init[1][0]:   3
Shallow copy:  w[0][0]:   2,    w[1][0]:   3

After changinge deep copied parameter, w

Original:  w_init[0][0]:   2,   w_init[1][0]:   3
Shallow copy:  w[0][0]:   0,    w[1][0]:   0
```

C - copy

Shallow copy

A ***shallow copy*** In C, involves copying the value of a pointer, so that both the original pointer and the copied pointer point to the same memory location.

- This means that changes to one will affect the other.

```c
float **copy(float src[][n]){

    /*
    Shallow copy
    Args:
        src (ndarray (row, col)):   initialized parameter

    Return:
        dest (ndarray (row, col)):  copy of the initilialized parameter
    */


    float **dest = (float **)calloc(m, sizeof(float *));
    if (dest ==NULL){
        perror("Error in allocating the memory");
        free(dest);
    }

    for (int i = 0; i < m; i++)
    {
        dest[i] = src[i];
    }

    return dest;
}
```

```c
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

#define m 2             //row
#define n 1             //col

int **copy(int src[][n]);
int **deepcopy(int src[][n]);

int main(){
    //Initialized paramter
    int w_init[][n] = {3, 2};

    //perform shallow copy
    int **w=copy(w_init);

    printf("\nBefore changinge shallow copied parameter, w \n\n");
    printf("Original:  w_init[0][0]:   %d, w_init[1][0]:   %d \n", w_init[0][0], w_init[1][0]);
    printf("Deep copy:  w[0][0]:   %d, w[1][0]:   %d\n", w[0][0], w[1][0]);

    //change the copied parameter
    w[0][0] = 0;
    w[1][0] = 0;

    printf("\nAfter changinge shallow copied parameter, w\n\n");
    printf("Original:  w_init[0][0]:   %d, w_init[1][0]:   %d \n", w_init[0][0], w_init[1][0]);
    printf("Deep copy:  w[0][0]:   %d, w[1][0]:   %d\n", w[0][0], w[1][0]);


    free(w);


    return 0;
}
```

```
venvsuzanodero@suzans-MacBook-Air Gradient_Descent % gcc copy.c
venvsuzanodero@suzans-MacBook-Air Gradient_Descent % ./a.out

Before changinge shallow copied parameter, w

Original:  w_init[0][0]:   3, w_init[1][0]:   2
Deep copy:  w[0][0]:   3, w[1][0]:   2

After changinge shallow copied parameter, w

Original:  w_init[0][0]:   0, w_init[1][0]:   0
Deep copy:  w[0][0]:   0, w[1][0]:   0
```

Deep copy

A ***Deep copy*** in python creates a new object and ensuring no shared refrences with the original objects by recursively copies all objects found inside the original object.

- If you modify new object it will not affect the original object.

```c
float **deepcopy(float src[][n]){
    /*
    Deep copy
    Args:
        src (ndarray (row, col)):   initialized parameter

    Return:
        dest (ndarray (row, col)):  copy of the initilialized parameter
    */

   float **dest = (float **)calloc(m, sizeof(float *));
   if (dest ==NULL)
   {
    perror("Error in allocating memory");
   }
   for (int i = 0; i < m; i++)
   {
    dest[i] = (float *)calloc(n, sizeof(float));

    if (dest[i]==NULL)
    {
        perror("Error in allocating memory");
        free(dest[i]);
    }
    free(dest);

    for (int j = 0; j < n; j++)
    {
        dest[i][j] = src[i][j];
    }

   }


    return dest;
}
```

```c
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

#define m 2             //row
#define n 1             //col

int **copy(int src[][n]);
int **deepcopy(int src[][n]);

int main(){
    //Initialized paramter
    int w_init[][n] = {3, 2};

    //perform deep copy
    int **w=deepcopy(w_init);

    printf("\nBefore changinge deep copied parameter, w \n\n");
    printf("Original:  w_init[0][0]:   %d, w_init[1][0]:   %d \n", w_init[0][0], w_init[1][0]);
    printf("Deep copy:  w[0][0]:   %d, w[1][0]:   %d\n", w[0][0], w[1][0]);

    //change the copied parameter
    w[0][0] = 0;
    w[1][0] = 0;

    printf("\nAfter changinge deep copied parameter, w\n\n");
    printf("Original:  w_init[0][0]:   %d, w_init[1][0]:   %d \n", w_init[0][0], w_init[1][0]);
    printf("Deep copy:  w[0][0]:   %d, w[1][0]:   %d\n", w[0][0], w[1][0]);

    for (int i = 0; i < m; i++)
    {
        free(w[i]);
    }

    free(w);


    return 0;
}
```

```
venvsuzanodero@suzans-MacBook-Air Gradient_Descent % gcc copy.c
venvsuzanodero@suzans-MacBook-Air Gradient_Descent % ./a.out

Before changinge deep copied parameter, w

Original:  w_init[0][0]:   3, w_init[1][0]:   2
Deep copy:  w[0][0]:   3, w[1][0]:   2

After changinge deep copied parameter, w

Original:  w_init[0][0]:   3, w_init[1][0]:   2
Deep copy:  w[0][0]:   0, w[1][0]:   0
venvsuzanodero@suzans-MacBook-Air Gradient_Descent %
```