# BASH

1. Dealing with a space separated file.
   If you do arr=($line) then it will convert it to array.
2. (In Bash, strings are automatically treated as integers when used in arithmetic contexts — no explicit type conversion is needed!)
3. Dealing with a csv or some other separation file in bash:

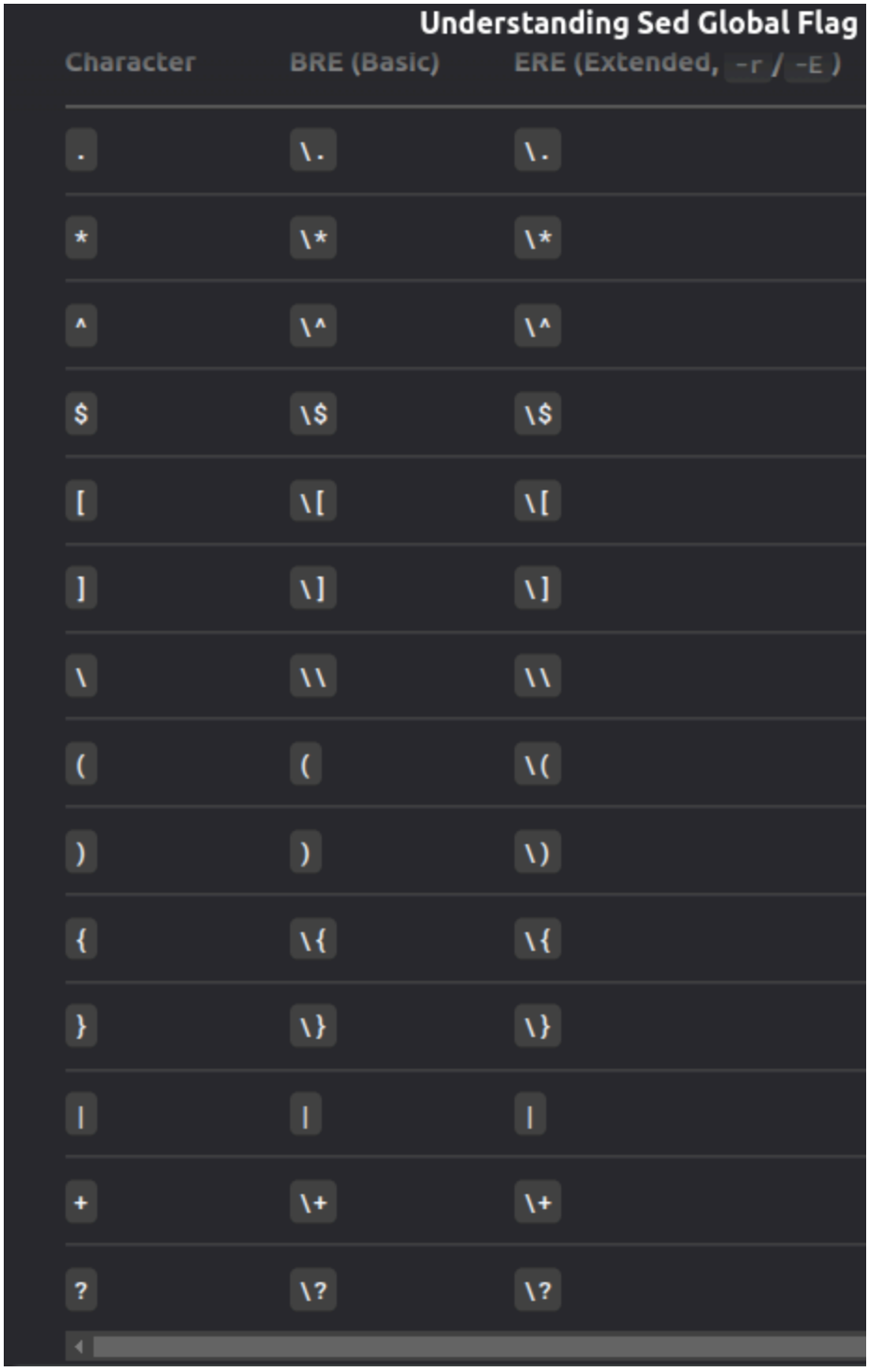
```
while IFS=',' read -ra ar; do
    if [[ ${ar[5]} = "total-marks" ]]; then
        echo "grades" > mygrades.csv
    elif [[ ${ar[5]} -gt 85 ]]; then
        echo "AA" >> mygrades.csv
    elif [[ ${ar[5]} -gt 65 ]]; then
        echo "AB" >> mygrades.csv
    elif [[ ${ar[5]} -gt 45 ]]; then
        echo "BB" >> mygrades.csv
    elif [[ ${ar[5]} -gt 35 ]]; then
        echo "CC" >> mygrades.csv
    else
        echo "F" >> mygrades.csv
    fi
done < remgrades.csv
```

4. sort -t ',' -k7 -k1,1r ug24tmp.csv >> ug24.csv

   In order to sort by multiple keys(also showing how to set reverse order)

# SED

1. For seeing what you need to escape and what not, just refer to the sed official manual(download krke jaana). Open appropriate regex syntax in that usme commands appropriate escaping ke saath hi diye hai.

2. 

   For treating literally

| Character / Operator | Meaning | BRE (default) | ERE (`-E` flag) |
|---|---|---|---|
| `.` | Any single character | ❌ No | ❌ No |
| `*` | Zero or more of previous | ❌ No | ❌ No |
| `+` | One or more | ✅ Yes (`\+`) | ❌ No |
| `?` | Zero or one | ✅ Yes (`\?`) | ❌ No |
| `{}` | Range of repetitions | ✅ Yes (`\{m,n\}`) | ❌ No |
| `()` | Grouping | ✅ Yes (`\(`, `\)`) | ❌ No |
| `\|` | `\|` | Alternation (OR) | ✅ Yes (`\|`) |
| `[]` | Character class | ❌ No | ❌ No |
| `^` | Start of line | ❌ No | ❌ No |
| `$` | End of line | ❌ No | ❌ No |
| `\` | Escape character | ✅ Yes (doubled `\\`) | ✅ Yes |

This is basic list

## AWK

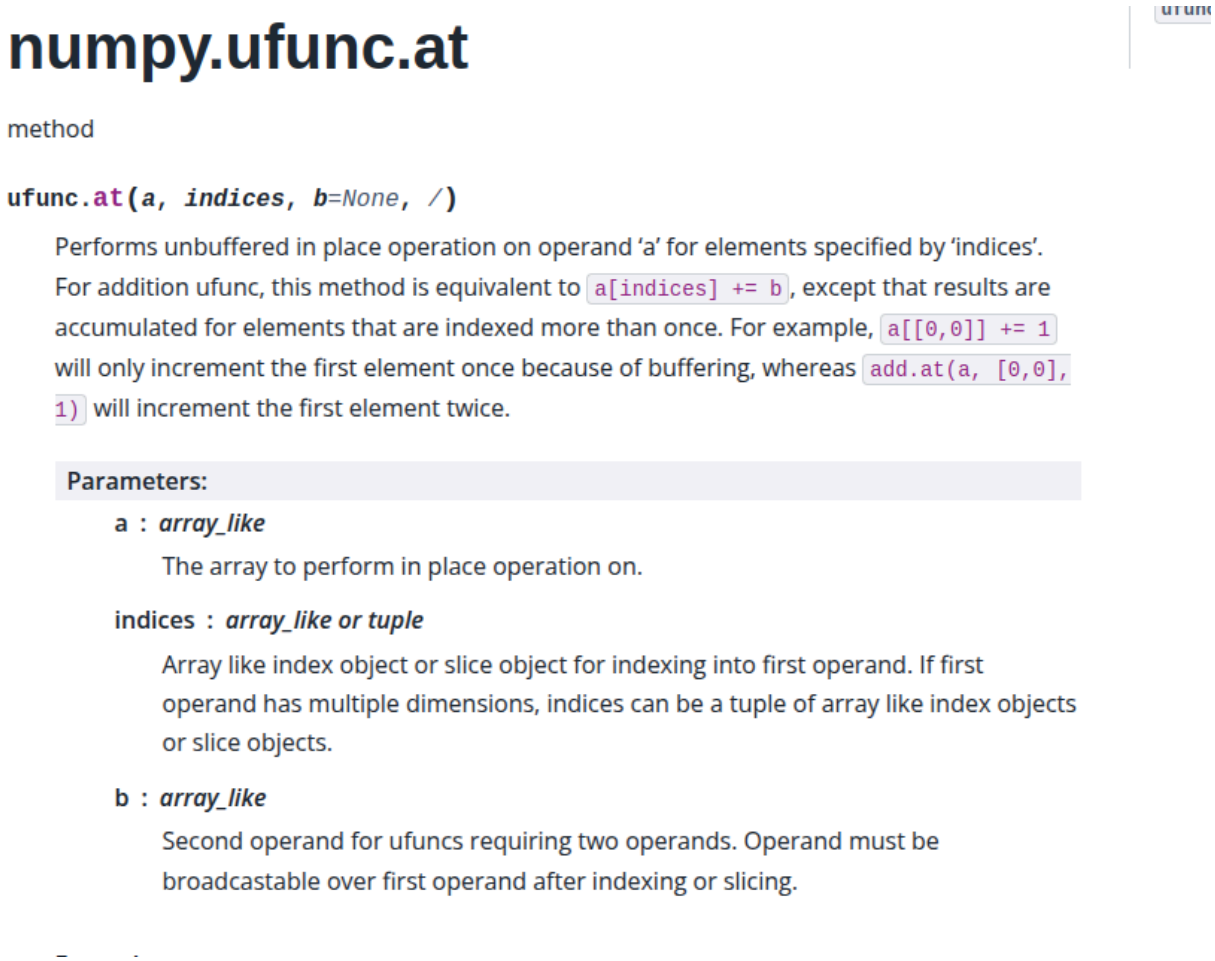
1. For IFS and stuff. Keep in mind that you have to write characters in double quotes only. Writing in single quotes doesn't work.

2. V.v.v.v.imp: for (key in array) is always unordered. If you want in some order DON'T USE IT. Rather use numbers. Like for (i=0; i<n; i++).
3. For string concatenation, you just have to separate the strings by space and write, no + in between.

## PYTHON

1. 
```
4  '''
5  import sys
6  # Please write your code in this file.
7  n=int(sys.argv[1])
```
can be used to take command line argument inputs.

2. Lots of min max functions in sorting searching section of documentation. Some in mathematical functions section too.
3. Mai bolta hu for broadcasting socho mt zyada, always align them properly and then do operations.



# numpy.ufunc.at

method

**ufunc.at(*a, indices, b=None, /*)**

Performs unbuffered in place operation on operand 'a' for elements specified by 'indices'. For addition ufunc, this method is equivalent to `a[indices] += b`, except that results are accumulated for elements that are indexed more than once. For example, `a[[0,0]] += 1` will only increment the first element once because of buffering, whereas `add.at(a, [0,0], 1)` will increment the first element twice.

**Parameters:**

**a** : *array_like*

The array to perform in place operation on.

**indices** : *array_like or tuple*

Array like index object or slice object for indexing into first operand. If first operand has multiple dimensions, indices can be a tuple of array like index objects or slice objects.

**b** : *array_like*

Second operand for ufuncs requiring two operands. Operand must be broadcastable over first operand after indexing or slicing.

4.
5. np.add.at can be really useful…See endsemques code to see how it is used. Even argmin, norm, loadtxt are nice functions.

6. 
Saved memory full
print("".join(string_list))
explain
The expression print("".join(string_list)) is used to join elements of a list of strings (string_list) into a single string and then print the resulting string. Here's a detailed breakdown:
Explanation:
1. "".join():
• The join() method in Python is used to join the elements of an iterable (in this case, string_list) into a single string.
• The "" (empty string) before .join() is the separator that will be used between each element of the list. Since it's an empty string, there will be no separator between the elements. If you used a space " " or comma ", ", that would be inserted between the elements.
2. string_list:
• This is the list of strings that you want to join. Each element in the list must be a string (if it's not, you will get a TypeError).
• For example, string_list = ['Hello', 'world', 'this', 'is', 'Python'].
3. Result of join():
• The join() method will concatenate all elements of string_list into one continuous string. For the list ['Hello', 'world', 'this', 'is', 'Python'], the result would be 'HelloworldthisisPython' because there's no separator (as we used "").
4. print():