Physics 77/88 - Fall 2024 - Homework 2

Functions, Loops, Lists and Arrays

Submit this notebook to bCourses to receive a credit for this assignment.
due: **Sept 25 2024**
Please upload both, the .ipynb file and the corresponding .pdf

Problem 1 (5P)

a) You need to pull a specific data set out of four lists. Each data set is taged by a date. Use *lambda* to define a function that returns the index of the data set in the list. The lists are:

In [1]:
```
L1 = ['May 24 2021', 'Oct 3 1998', 'Feb 12 2023', 'Jun 11 2013', 'Nov 12 198
L2 = ['Oct 3 1998', 'Jan 21 2012', 'Apr 15 1990', 'Oct 3 1998']
L3 = ['Mar 21 2002', 'Oct 3 2019', 'Jun 12 2013', 'Dec 31 2001', 'Aug 12 199
L4 = ['May 24 2021', 'Oct 3 1998', 'Oct 3 1998', 'Jun 11 2013', 'Oct 3 1998'
```

and the data set you want to pull is the one from 'Oct 3 1998'. (3P)

In [2]:
```
find_date = lambda S: [i for i, s in enumerate(S) if s == 'Oct 3 1998']
```

b) Use *map* to retrieve all indices from all lists for the specific data set **at once** (2P).

In [3]:
```
output = map(find_date, [L1, L2, L3, L4])
print(list(output))
```

```
[[1], [0, 3], [], [1, 2, 4]]
```

Problem 2 (5P)

A prime number (or a prime) is a natural number greater than 1 that is not a product of two smaller natural numbers.
Write a function **IsPrime** using *def* that takes a natural number as input and returns a message in the terminal stating whether the number is a prime or not. Try to write the code as efficient as possible.
Note 1: Check out the pythonian trick of using % to evaluate if a number can be divided by another number without remainder.

In [4]:
```
def isPrime(n):
    for i in range(2, int(n**0.5)+1):
        if n % i == 0:
            print(str(n) + ' is not prime')
```

```
            break
    else:
        print(str(n) + ' is prime')
```

Note 2: Check out the usage of *if not* in Python.
Note 3: Don't use any external library (like e. g. *math* or *numpy*) to solve this problem.

In [5]:
```
for i in range(1, 26):
    isPrime(i)
```

```
1 is prime
2 is prime
3 is prime
4 is not prime
5 is prime
6 is not prime
7 is prime
8 is not prime
9 is not prime
10 is not prime
11 is prime
12 is not prime
13 is prime
14 is not prime
15 is not prime
16 is not prime
17 is prime
18 is not prime
19 is prime
20 is not prime
21 is not prime
22 is not prime
23 is prime
24 is not prime
25 is not prime
```

Problem 3 (5P)

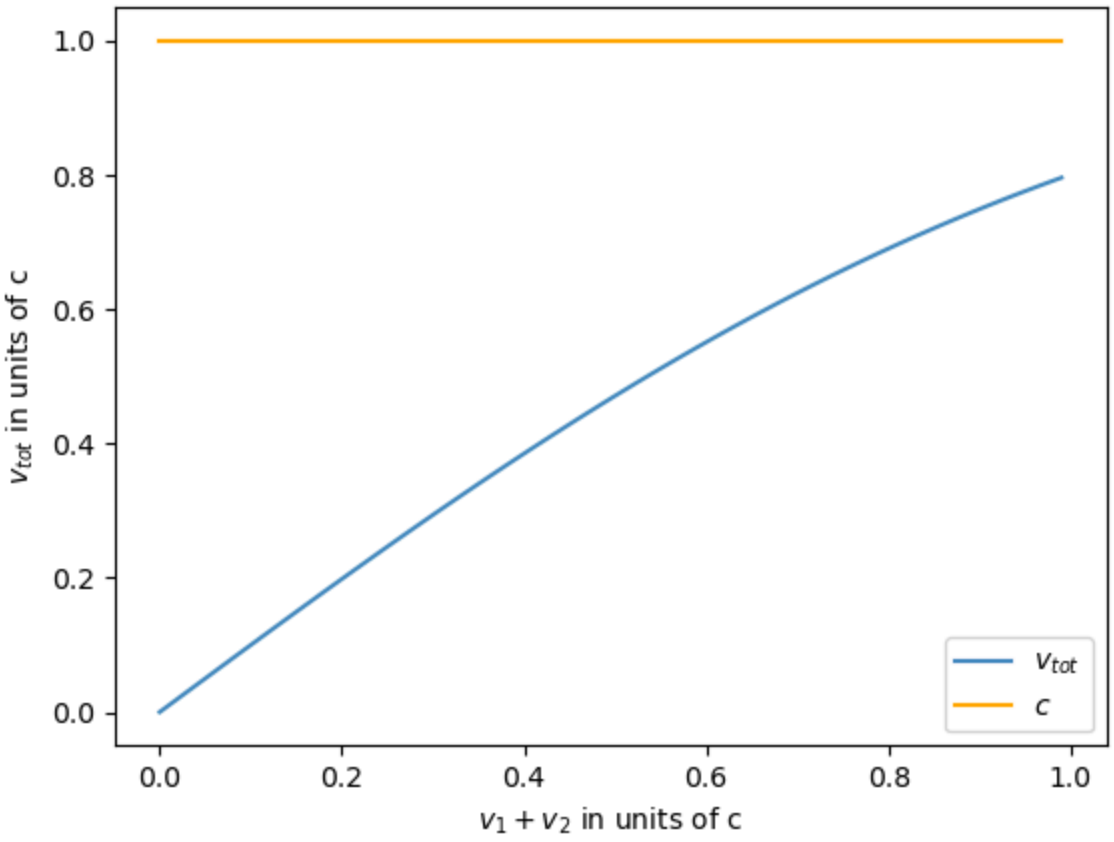
Nothing can move faster than the speed of light $c \approx 300\,000\ \frac{km}{s}$. Therefore, adding up two velocities v_1 and v_2 will always result in a total velocity $v_{tot} < c$. The corresponding equation is:

$$v_{tot} = \frac{v_1 + v_2}{1 + \frac{v_1\, v_2}{c^2}}$$

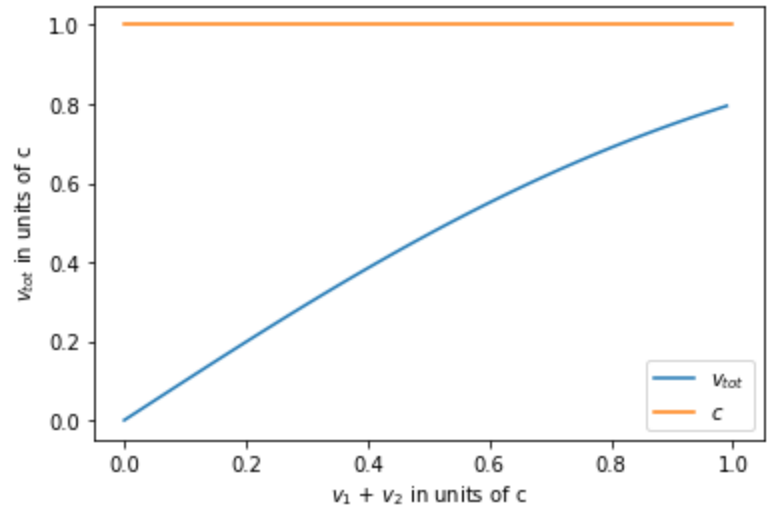
Write a function **Add_velos** using *def* that takes v_1 and v_2 as input arguments and $c = 1$ as default input argument (i. e. the velocities are measured in units of c). The function should return v_{tot} and create a plot showing a diagram with v_{tot} compared to c depending on v_1 and v_2 (see below).

In [6]:
```
import numpy as np
import matplotlib.pyplot as plt

def Add_velos(v1, v2, c = 1):
    v1p = np.arange(0, v1, v1/100)
    v2p = np.arange(0, v2, v2/100)
    cplot = np.repeat(c, 100)

    vtot = (v1 + v2) / (1 + ((v1*v2)/c**2))

    xplot = v1p + v2p
    yplot = (xplot) / (1 + (v1p*v2p/c**2))

    plt.plot(xplot, yplot, '-', color = '#448BBE', label = '$v_{tot}$')
    plt.plot(xplot, cplot, '-', color = '#FFA500', label = '$c$')

    plt.xlabel('$v_1 + v_2$ in units of c')
    plt.ylabel('$v_{tot}$ in units of c')
    plt.legend()
    plt.show()

    print(f'Total velocity is: {vtot:.3f} c')
```

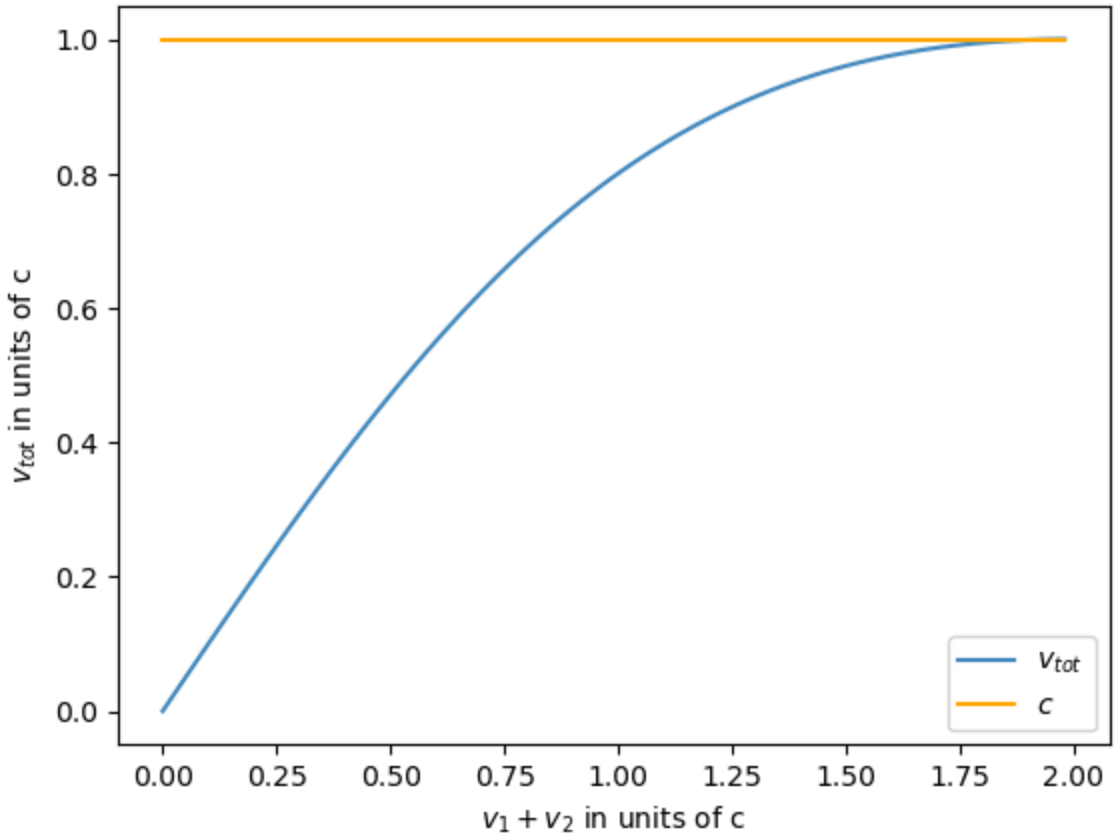
In [7]:
```
Add_velos(0.5,0.5)
```



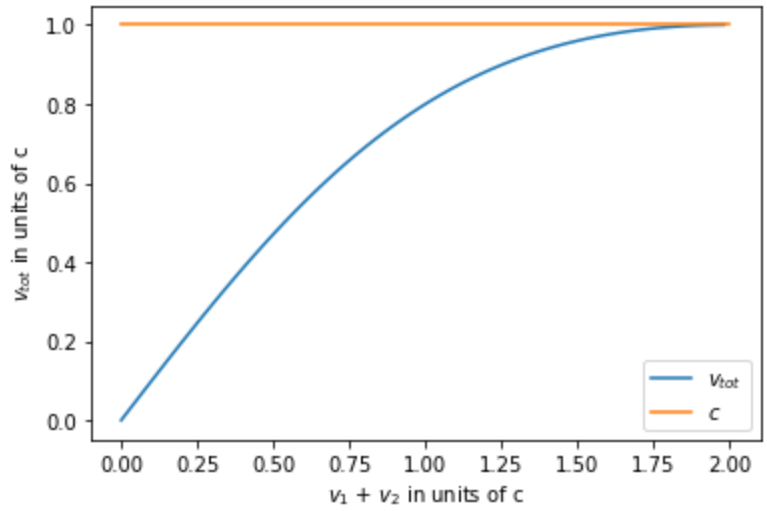

```
Total velocity is: 0.800 c
```



In [8]:
```
Add_velos(1,1)
```




```
Total velocity is: 1.000 c
```



Problem 4 (optional 5P)

Write a function **CountDown** that takes a natural number *N* as input argument. The function should reduce *N* by *1* each time it calls itself *recursively* and stops once it has counted down to zero. The function should also print the current value of *N* to the terminal.

Note: Don't use any external library (like e. g. *math* or *numpy*) to solve this problem.

In [9]:
```
def CountDown(N):
    print(N)
    if N>0:
        CountDown(N-1)
```

In [10]:
```
CountDown(10)
```

```
10
9
8
7
6
5
4
3
2
1
0
```