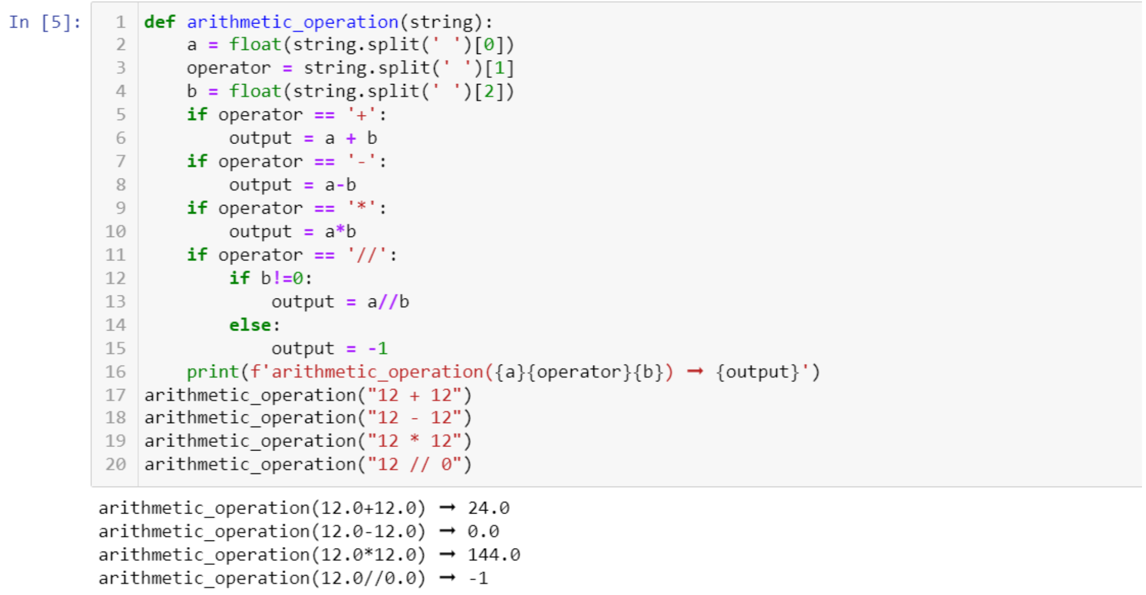
1. Create a function to perform basic arithmetic operations that includes addition, subtraction, multiplication and division on a string number (e.g. "12 + 24" or "23 - 21" or "12 // 12" or "12 * 21").

Here, we have 1 followed by a space, operator followed by another space and 2. For the challenge, we are going to have only two numbers between 1 valid operator. The return value should be a number. eval() is not allowed. In case of division, whenever the second number equals "0" return -1.
For example: "15 // 0" ➞ -1

Examples:
arithmetic_operation("12 + 12") ➞ 24 // 12 + 12 = 24
arithmetic_operation("12 - 12") ➞ 24 // 12 - 12 = 0
arithmetic_operation("12 * 12") ➞ 144 // 12 * 12 = 144
arithmetic_operation("12 // 0") ➞ -1 // 12 / 0 = -1

Ans:



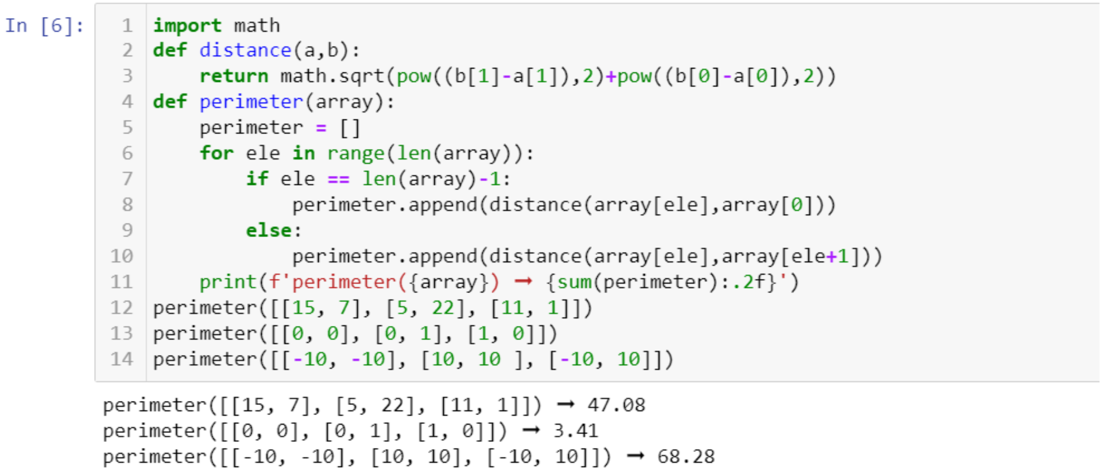
```python
def arithmetic_operation(string):
    a = float(string.split(' ')[0])
    operator = string.split(' ')[1]
    b = float(string.split(' ')[2])
    if operator == '+':
        output = a + b
    if operator == '-':
        output = a-b
    if operator == '*':
        output = a*b
    if operator == '//':
        if b!=0:
            output = a//b
        else:
            output = -1
    print(f'arithmetic_operation({a}{operator}{b}) ➞ {output}')
arithmetic_operation("12 + 12")
arithmetic_operation("12 - 12")
arithmetic_operation("12 * 12")
arithmetic_operation("12 // 0")
```

```
arithmetic_operation(12.0+12.0) ➞ 24.0
arithmetic_operation(12.0-12.0) ➞ 0.0
arithmetic_operation(12.0*12.0) ➞ 144.0
arithmetic_operation(12.0//0.0) ➞ -1
```

2. Write a function that takes the coordinates of three points in the form of a 2d array and returns the perimeter of the triangle. The given points are the vertices of a triangle on a two-dimensional plane.

Examples:
perimeter([[15, 7], [5, 22], [11, 1]]) ➞ 47.08
perimeter([[0, 0], [0, 1], [1, 0]]) ➞ 3.42
perimeter([[-10, -10], [10, 10], [-10, 10]]) ➞ 68.28

Ans:



```
In [6]:
import math
def distance(a,b):
    return math.sqrt(pow((b[1]-a[1]),2)+pow((b[0]-a[0]),2))
def perimeter(array):
    perimeter = []
    for ele in range(len(array)):
        if ele == len(array)-1:
            perimeter.append(distance(array[ele],array[0]))
        else:
            perimeter.append(distance(array[ele],array[ele+1]))
    print(f'perimeter({array}) ➞ {sum(perimeter):.2f}')
perimeter([[15, 7], [5, 22], [11, 1]])
perimeter([[0, 0], [0, 1], [1, 0]])
perimeter([[-10, -10], [10, 10 ], [-10, 10]])
```

```
perimeter([[15, 7], [5, 22], [11, 1]]) ➞ 47.08
perimeter([[0, 0], [0, 1], [1, 0]]) ➞ 3.41
perimeter([[-10, -10], [10, 10], [-10, 10]]) ➞ 68.28
```

3. A city skyline can be represented as a 2-D list with 1s representing buildings. In the example below, the height of the tallest building is 4 (second-most right column).

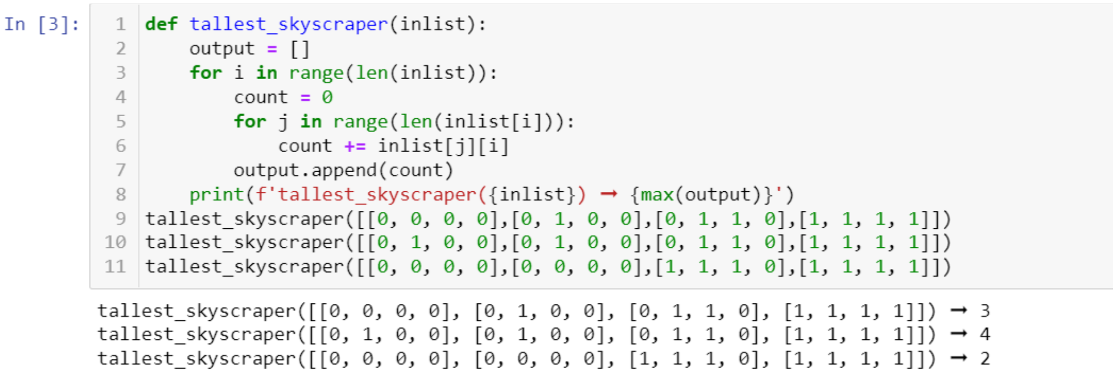
[[0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 1, 0],
[0, 0, 1, 0, 1, 0],
[0, 1, 1, 1, 1, 0],
[1, 1, 1, 1, 1, 1]]

Create a function that takes a skyline (2-D list of 0's and 1's) and returns the height of the tallest skyscraper.

Examples:

tallest_skyscraper([[0, 0, 0, 0],[0, 1, 0, 0],[0, 1, 1, 0],[1, 1, 1, 1]]) ➞ 3
tallest_skyscraper([[0, 1, 0, 0],[0, 1, 0, 0],[0, 1, 1, 0],[1, 1, 1, 1]]) ➞ 4
tallest_skyscraper([[0, 0, 0, 0],[0, 0, 0, 0],[1, 1, 1, 0],[1, 1, 1, 1]]) ➞ 2

Ans:



```
In [3]:
def tallest_skyscraper(inlist):
    output = []
    for i in range(len(inlist)):
        count = 0
        for j in range(len(inlist[i])):
            count += inlist[j][i]
        output.append(count)
    print(f'tallest_skyscraper({inlist}) ➞ {max(output)}')
tallest_skyscraper([[0, 0, 0, 0],[0, 1, 0, 0],[0, 1, 1, 0],[1, 1, 1, 1]])
tallest_skyscraper([[0, 1, 0, 0],[0, 1, 0, 0],[0, 1, 1, 0],[1, 1, 1, 1]])
tallest_skyscraper([[0, 0, 0, 0],[0, 0, 0, 0],[1, 1, 1, 0],[1, 1, 1, 1]])
```

```
tallest_skyscraper([[0, 0, 0, 0], [0, 1, 0, 0], [0, 1, 1, 0], [1, 1, 1, 1]]) ➞ 3
tallest_skyscraper([[0, 1, 0, 0], [0, 1, 0, 0], [0, 1, 1, 0], [1, 1, 1, 1]]) ➞ 4
tallest_skyscraper([[0, 0, 0, 0], [0, 0, 0, 0], [1, 1, 1, 0], [1, 1, 1, 1]]) ➞ 2
```

4. A financial institution provides professional services to banks and claims charges from the customers based on the number of man-days provided. Internally, it has set a scheme to motivate and reward staff to meet and exceed targeted billable utilization and revenues by paying a bonus for each day claimed from customers in excess of a threshold target.

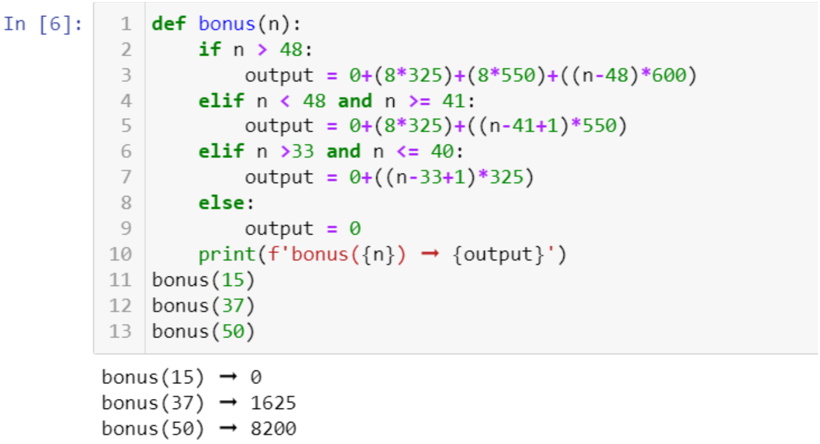
This quarterly scheme is calculated with a threshold target of 32 days per quarter, and the incentive payment for each billable day in excess of such threshold target is shown as follows:

Days	Bonus
0 to 32 days	Zero
33 to 40 days	SGD$325 per billable day
41 to 48 days	SGD$550 per billable day
Greater than 48 days	SGD$600 per billable day

Please note that incentive payment is calculated progressively. As an example, if an employee reached total billable days of 45 in a quarter, his/her incentive payment is computed as follows:
32*0 + 8*325 + 5*550 = 5350
Write a function to read the billable days of an employee and return the bonus he/she has obtained in that quarter.

Examples:
bonus(15) ➞ 0
bonus(37) ➞ 1625
bonus(50) ➞ 8200

Ans:



```
In [6]:
def bonus(n):
    if n > 48:
        output = 0+(8*325)+(8*550)+((n-48)*600)
    elif n < 48 and n >= 41:
        output = 0+(8*325)+((n-41+1)*550)
    elif n >33 and n <= 40:
        output = 0+((n-33+1)*325)
    else:
        output = 0
    print(f'bonus({n}) ➞ {output}')
bonus(15)
bonus(37)
bonus(50)
```

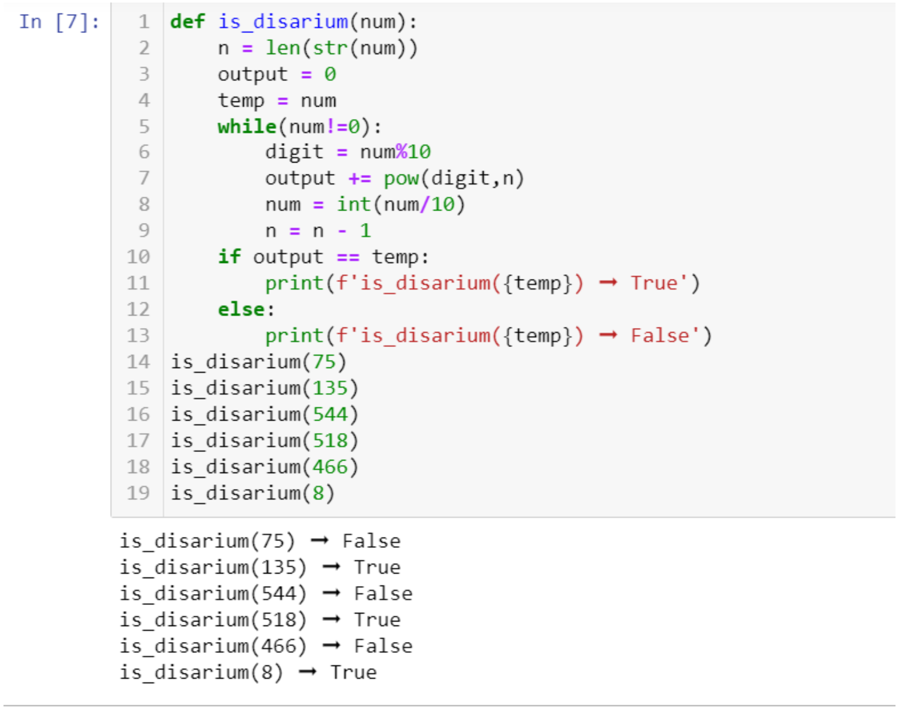
```
bonus(15) ➞ 0
bonus(37) ➞ 1625
bonus(50) ➞ 8200
```

5. A number is said to be Disarium if the sum of its digits raised to their respective positions is the number itself.

Create a function that determines whether a number is a Disarium or not.
Examples:
is_disarium(75) ➞ False
# 7^1 + 5^2 = 7 + 25 = 32
is_disarium(135) ➞ True
# 1^1 + 3^2 + 5^3 = 1 + 9 + 125 = 135
is_disarium(544) ➞ False
is_disarium(518) ➞ True
is_disarium(466) ➞ False

is_disarium(8) ➞ True

Ans:



In [7]:

```
def is_disarium(num):
    n = len(str(num))
    output = 0
    temp = num
    while(num!=0):
        digit = num%10
        output += pow(digit,n)
        num = int(num/10)
        n = n - 1
    if output == temp:
        print(f'is_disarium({temp}) ➞ True')
    else:
        print(f'is_disarium({temp}) ➞ False')
is_disarium(75)
is_disarium(135)
is_disarium(544)
is_disarium(518)
is_disarium(466)
is_disarium(8)
```

```
is_disarium(75) ➞ False
is_disarium(135) ➞ True
is_disarium(544) ➞ False
is_disarium(518) ➞ True
is_disarium(466) ➞ False
is_disarium(8) ➞ True
```