Laboratorio 1 - Pablo Calcumil

September 2, 2020

MAT281 - 2ř Semestre 2020
Profesor: Francisco Alfaro Medina

0.1 Problema 01

0.1.1 a) Calcular el número π

Al escribir la serie, la función queda de la siguiente manera:

```
In [19]: def Calculo_pi(n):
             suma = 0
             for k in range(1,n + 1):
                 suma = ((-1)**(k + 1))/(2*k - 1) + suma
             return 4 * suma
```

```
In [7]: Calculo_pi(3)
```

```
Out[7]: 3.466666666666667
```

```
In [8]: Calculo_pi(100000)
```

```
Out[8]: 3.1415826535897198
```

0.1.2 b) Calcular el número e

Necesitamos una función para los factoriales `factorial()` presentes como denominadores en la serie, por lo tanto las funciones quedan de la siguiente manera:

```
In [15]: def factorial(numero):
             if numero == 1 or numero == 0:
                 return 1
             else:
                 return (numero * factorial(numero - 1))

         def Calculo_e(largo):
             suma = 0
             for k in range(0,largo):
                 denominador = factorial(k)
                 suma = 1/denominador + suma
             return suma
```

```
In [16]: Calculo_e(3)

Out[16]: 2.5

In [18]: Calculo_e(1000)

Out[18]: 2.7182818284590455
```

0.2 Problema 02

Como el problema se centra mayormente en la suma de sus divisores propios, se crea la función para obtener estos `suma_propios()`, y luego la función buscada la que nos dirá si son o no amigos `Amigos()`.

```
In [36]: def suma_propios(numero):
             if numero == 1:
                 return 0
             divisores = []
             numero2 = int(numero/2 + 1)
             for k in range(1,numero2):
                 if numero % k == 0:
                     divisores.append(k)
             return sum(divisores)

         def Amigos(numero1,numero2):
             if suma_propios(numero1) == numero2 and numero1 == suma_propios(numero2):
                 return True
             return False

In [37]: Amigos(220,284)

Out[37]: True

In [38]: Amigos(6,5)

Out[38]: False
```

0.3 Problema 03

La función de `Collatz()` queda de la siguiente manera:

```
In [42]: def Collatz(Numero):
             if Numero <= 1:
                 return 1
             collatz = [Numero]
             while Numero != 1:
                 if Numero % 2 == 1:
                     Numero = int(Numero * 3 + 1)
                     collatz.append(Numero)
                 else:
```

```
                Numero = int(Numero / 2)
                collatz.append(Numero)
        return collatz
```

```
In [46]: Collatz(9)
```

```
Out[46]: [9, 28, 14, 7, 22, 11, 34, 17, 52, 26, 13, 40, 20, 10, 5, 16, 8, 4, 2, 1]
```

0.4 Problema 04

Como la función `Goldbach()` es la suma de dos numeros primos, necesitamos verificar primero que los numeros que sumaremos son numeros primos o no, para eso está la función `esonoprimo()`, y al trabajar al final con solo numeros primos se busca la combinación que cumpla con el objetivo:

```
In [85]: def esonoprimo(numero):
            if numero < 2:
                return 0
            elif numero == 2:
                return 1
            numero2 = int(numero / 2 + 1)
            for k in range(2,numero2):
                if numero % k == 0:
                    return 0
            return 1

        def Goldbach(Numero):
            if Numero < 3:
                return False
            primos = []
            for k in range(2,Numero):
                if esonoprimo(k) == 1:
                    primos.append(k)
            for elemento in primos:
                for number in primos:
                    if elemento + number == Numero:
                        primos.append(elemento)
                        primos.append(number)
                        return (primos[-2],primos[-1])
```

```
In [88]: Goldbach(4)
```

```
Out[88]: (2, 2)
```

```
In [89]: Goldbach(6)
```

```
Out[89]: (3, 3)
```

```
In [90]: Goldbach(8)
```

```
Out[90]: (3, 5)
```

```
In [92]: Goldbach(1000)
```

```
Out[92]: (3, 997)
```