Codificación y Programación.

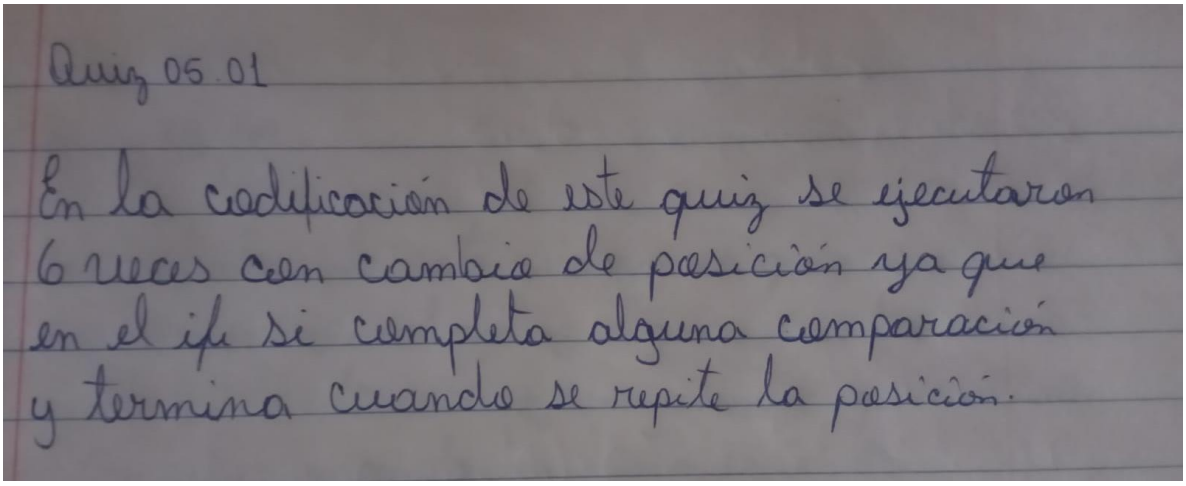
Quiz Capítulos #5, #6 Y #7.

Q. 05-01. ¿Cuántas comparaciones se ejecutaron en el siguiente proceso de clasificación por inserción?

```
def bubblesort(S):
    n = len(S)
    for i in range(n):
        print(S)
        for j in range(n - 1):
            if S[j] > S[j + 1]:
                S[j], S[j + 1] = S[j + 1], S[j]
```

```
S = [50, 30, 40, 10, 20]
bubblesort(S)
print(S)
```

```
[50, 30, 40, 10, 20]
[30, 40, 10, 20, 50]
[30, 10, 20, 40, 50]
[10, 20, 30, 40, 50]
[10, 20, 30, 40, 50]
[10, 20, 30, 40, 50]
```

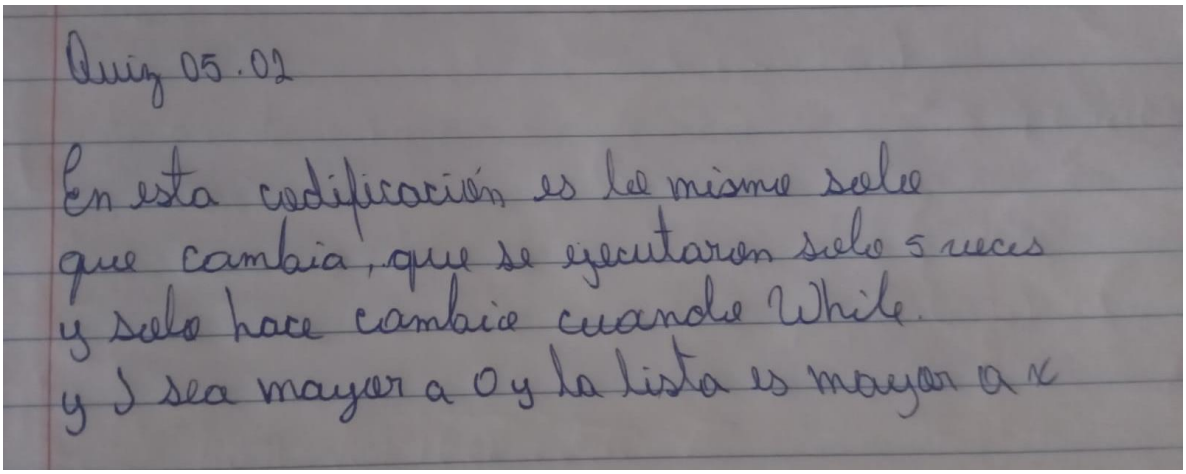


Q. 05-02. ¿Cuántas comparaciones se ejecutaron en el siguiente proceso de clasificación por inserción?

```python
def insertionsort2(S):
    n = len(S)
    for i in range(1, n):
        print(S)
        x = S[i]
        j = i - 1
        while j >= 0 and S[j] > x:
            S[j + 1] = S[j]
            j -= 1
        S[j + 1] = x
```

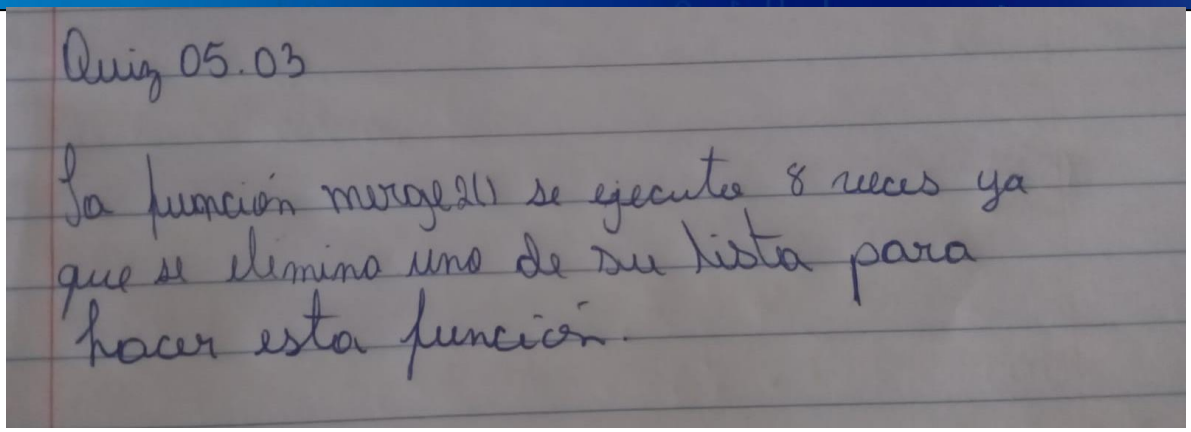
```python
S = [50, 30, 40, 10, 20]
insertionsort2(S)
print(S)
```

```
[50, 30, 40, 10, 20]
[30, 50, 40, 10, 20]
[30, 40, 50, 10, 20]
[10, 30, 40, 50, 20]
[10, 20, 30, 40, 50]
```



Q. 05-03. ¿Cuántas veces se ejecutó la función merge2() en el siguiente proceso de clasificación por fusión?

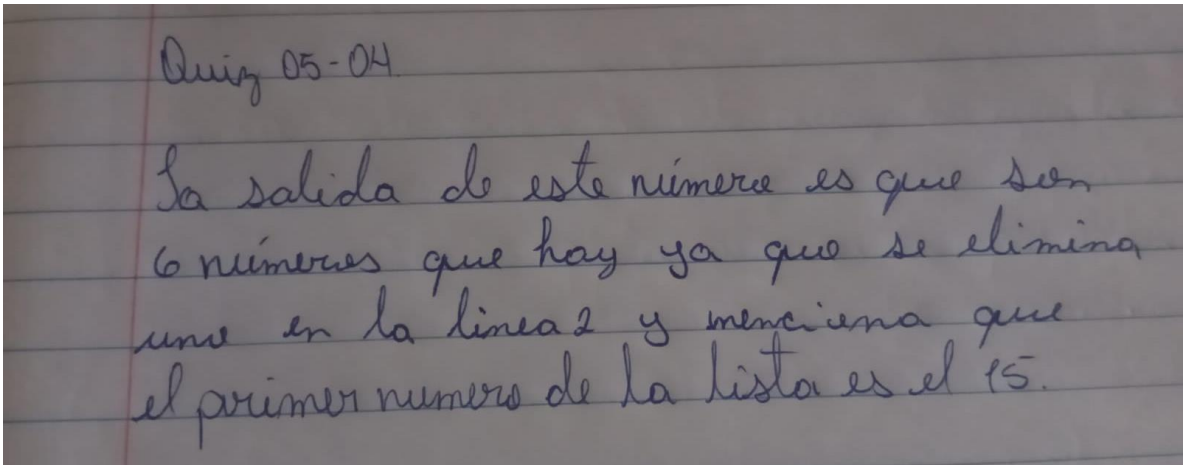
```python
S = [6, 2, 11, 7, 5, 4, 8, 16, 10, 3]
mergesort2(S, 0, len(S) - 1)
print(S)
```



Q. 05-04. Dada la lista a continuación, escriba la salida después de ejecutar la función partición1().

```
S = [15, 10, 12, 20, 25, 13, 22]
partition1(S, 0, len(S) - 1)
print(S)
```

```
[15, 10, 12, 20, 25, 13, 22] 0 6 pivot =  15
[13, 10, 12, 15, 25, 20, 22]
```



Q. 06-01. Diseñe un algoritmo que halle la función factorial de cualquier número empleando recursividad

In [4]:

```
#Quiz 06.01

n=int(input("ingrese un numero : "))

def imprimir(x):
    #print(x)
    if x==0:
        return 1
    else:
        return x*imprimir(x-1)

print(imprimir(n))
```

```
ingrese un numero : 5
120
```

Q. 06-02. Diseñe un algoritmo que halle la función factorial de cualquier número empleando memoización

In [6]:
```
#quiz 06.02
def factorial_1(n):

    if n in factorial_1.__dict__:
        return factorial_1.__dict__[n]

    if n > 1:
        fac = n * factorial_1(n - 1)
    else:
        fac = 1

    factorial_1.__dict__[n] = fac
    return fac

n=int(input("ingrese un numero : "))
print(factorial_1(n))
```

```
ingrese un numero : 5
120
```

Q. 06-03. Diseñe un algoritmo que codifique la secuencia de Fibonacci empleando recursividad

In [5]:
```
#Quiz 06.03
fib=int(input("Ingrese un numero: "))

def fibonacci(n):
    if n==0:
        return 0
    elif n==1:
        return 1
    else:
        return fibonacci(n-1)+fibonacci(n-2)
print(fibonacci(fib))

```

```
Ingrese un numero: 35
9227465
```

Q. 06-04. Ejercicio lógico: en cada palabra, reemplace las letras por un número, teniendo en cuenta que para cada palabra separada por un espacio la suma de sus dígitos es un número al cuadrado Encuentra el número representado por cada letra.

A MERRY XMAS TO ALL

A	M	E	R	R	Y	X	M	A	S	T	O	A	L	L

- **Código de referencia**

```
from math import sqrt, floor
import time

def is_square_digitsum(n):
    s = 0
    while n > 0:
        s += n % 10
        n //= 10
    if sqrt(s) == int(sqrt(s)):
        return True
    return False

def find_all_squares():
    sqrs = [[] for _ in range(5)]
    for i in range(1, floor(sqrt(10 ** 5)) + 1):
        n = i * i
        if not is_square_digitsum(n):
            continue
        s = str(n)
```

- **Código de referencia**

```
        if len(s) == 3 and s[1] != s[2]:
            continue
        if len(s) == 5 and s[2] != s[3]:
            continue
        if len(s) in [4, 5] and len(set(s)) != 4:
            continue
        sqrs[len(s) - 1].append(n)
    return sqrs

def promising(s, n, dic):
    for i in range(len(s)):
        digit = int(str(n)[i])
        for key, value in dic.items():
            if key == s[i] and value != digit:
                return False
            if value == digit and key != s[i]:
                return False
    return True
```

- **Código de referencia**

```
def solve(words, dic, squares):
    global solved
    if (len(words) == 0):
        solved = dic
    else:
        s = words[0]
        candidates = squares[len(s) - 1]
        for n in candidates:
            if promising(s, n, dic):
                newdic = dic.copy()
                for i in range(len(s)):
                    newdic[s[i]] = int(str(n)[i])
                solve(words[1:], newdic, squares)

def main():
    squares = find_all_squares()
    # print(squares)
```

- **Código de referencia**

```
    words = ['A', 'TO', 'ALL', 'XMAS', 'MERRY']
    dic = {}
    solve(words, dic, squares)
    for word in words:
        print(word, end=": ")
        for c in word:
            print(solved[c], end="")
        print()

start = time.time()
solved = {}
main()
end = time.time()
print("Elapsed Time: ", end - start, " seconds")
```

Q. 07-01. Convierta los siguientes datos del diccionario en un objeto dataframe usando Pandas. (El objeto del marco de datos se denomina df.)

```
d = {'col1': [1, 2], 'col2': [3, 4], 'col3': [5, 6], 'col4': [7, 8]}
```

In [45]:
```
#Quiz 07.01
import pandas as pd
import numpy as np

df=pd.DataFrame({"col1":[1,2],"col2":[3,4],"col3":[5,6],"col4":[7,8]})

df
```

Out[45]:

	col1	col2	col3	col4
0	1	2	5	7
1	2	4	6	8

Q. 07-02. Para el dataframe creado en la Pregunta 1, cree un nuevo dataframe que consista sólo en los datos de la columna con el nombre de columna 'col4'. (El dataframe se denomina new_df.)

In [164]:
```
#Quiz 07.02
import pandas as pd
import numpy as np

df=pd.DataFrame({"col1":[1,2],"col2":[3,4],"col3":[5,6],"col4":[7,8]})

df_new = df.drop(['col1', 'col2','col3'], axis=1)

df_new

```

Out[164]:

	col4
0	7
1	8

Q. 07-03. El índice del dataframe creado en la Pregunta 1 es 0,1. Escriba un comando para cambiar el nombre de estos índices a primero y segundo.

In [163]:
```
#Quiz 07.03
import numpy as np
import pandas as pd
from pandas import Series, DataFrame

df=pd.DataFrame({"col1":[1,2],"col2":[3,4],"col3":[5,6],"col4":[7,8]})

df.index = ['Primero', 'Segundo']

df

```

Out[163]:

	col1	col2	col3	col4
Primero	1	3	5	7
Segundo	2	4	6	8

Q. 07-04. Escriba un comando para buscar datos faltantes en el dataframe df creado en la Pregunta 1 e imprima el resultado. (Sin embargo, los datos que faltan deben devolverse como verdaderos).

In [165]:
```
#Quiz 07.04
import pandas as pd

import numpy as np

new_df = {'col1':[1, 2], 'col2': [3, 4], 'col3':[ 5,6],'col4': [7, 8,]}

df = pd.DataFrame(new_df)

df.notnull()
```

Out[165]:

	col1	col2	col3	col4
0	True	True	True	True
1	True	True	True	True

Q. 07-05. Escriba un comando para verificar el resumen de las estadísticas descriptivas (desviación estándar, valor mínimo, moda, etc.) del marco de datos df creado en la pregunta 1 e imprima el resultado.

In [130]:
```
#Quiz 07.05
import pandas as pd
import statistics

df=pd.DataFrame({"col1":[1,2],"col2":[3,4],"col3":[5,6],"col4":[7,8]})
#count
df['col5'] = df.count(axis=1)
df
```

Out[130]:

	col1	col2	col3	col4	col5
0	1	3	5	7	4
1	2	4	6	8	4

In [131]:
```
#Quiz 07.05
import pandas as pd
import statistics

df=pd.DataFrame({"col1":[1,2],"col2":[3,4],"col3":[5,6],"col4":[7,8]})
#sum
df['col5'] = df.sum(axis=1)
df
```

Out[131]:

	col1	col2	col3	col4	col5
0	1	3	5	7	16
1	2	4	6	8	20

In [168]:
```
#Quiz 07.05
import pandas as pd
import statistics

df=pd.DataFrame({"col1":[1,2],"col2":[3,4],"col3":[5,6],"col4":[7,8]})

#mean
df['col6'] = df.mean(axis=1)
df
```

Out[168]:

	col1	col2	col3	col4	col6
0	1	3	5	7	4.0
1	2	4	6	8	5.0

In [167]:
```
#Quiz 07.05
import pandas as pd
import statistics

df=pd.DataFrame({"col1":[1,2],"col2":[3,4],"col3":[5,6],"col4":[7,8]})


#median
df['col7'] = df.median(axis=1)
df
```

Out[167]:

	col1	col2	col3	col4	col7
0	1	3	5	7	4.0
1	2	4	6	8	5.0

In [162]:
```
#Quiz 07.05
import pandas as pd
import statistics

df=pd.DataFrame({"col1":[1,2],"col2":[3,4],"col3":[5,6],"col4":[7,8]})

#mode
a= df.mode(axis=1)
print(a)
df
```

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

Out[162]:

	col1	col2	col3	col4
0	1	3	5	7
1	2	4	6	8

In [137]:
```
#Quiz 07.05
import pandas as pd
import statistics

df=pd.DataFrame({"col1":[1,2],"col2":[3,4],"col3":[5,6],"col4":[7,8]})

#std
df['col9'] = df.std(axis=1)
df
```

Out[137]:

	col1	col2	col3	col4	col9
0	1	3	5	7	2.581989
1	2	4	6	8	2.581989

In [138]:
```
#Quiz 07.05
import pandas as pd
import statistics

df=pd.DataFrame({"col1":[1,2],"col2":[3,4],"col3":[5,6],"col4":[7,8]})

#min
df['col10'] = df.min(axis=1)
df
```

Out[138]:

	col1	col2	col3	col4	col10
0	1	3	5	7	1
1	2	4	6	8	2

In [139]:
```
#Quiz 07.05
import pandas as pd
import statistics

df=pd.DataFrame({"col1":[1,2],"col2":[3,4],"col3":[5,6],"col4":[7,8]})

#maximum
df["col11"]=df.max(axis=1)
df
```

Out[139]:

	col1	col2	col3	col4	col11
0	1	3	5	7	7
1	2	4	6	8	8

In [159]:
```
#Quiz 07.05
import pandas as pd
import statistics

df=pd.DataFrame({"col1":[1,2],"col2":[3,4],"col3":[5,6],"col4":[7,8]})

#absolute
a=df.abs()
print(a)
df
```

```
   col1  col2  col3  col4
0     1     3     5     7
1     2     4     6     8
```

Out[159]:

	col1	col2	col3	col4
0	1	3	5	7
1	2	4	6	8

In [141]:
```
#Quiz 07.05
import pandas as pd
import statistics

df=pd.DataFrame({"col1":[1,2],"col2":[3,4],"col3":[5,6],"col4":[7,8]})

#product
df["col13"]=df.prod(axis=1)
df
```

Out[141]:

	col1	col2	col3	col4	col13
0	1	3	5	7	105
1	2	4	6	8	384

In [160]:
```
#Quiz 07.05
import pandas as pd
import statistics

df=pd.DataFrame({"col1":[1,2],"col2":[3,4],"col3":[5,6],"col4":[7,8]})

#cumulative sum
a=df.cumsum(axis=1)
print(a)
df
```

```
   col1  col2  col3  col4
0     1     4     9    16
1     2     6    12    20
```

Out[160]:

	col1	col2	col3	col4
0	1	3	5	7
1	2	4	6	8

In [161]:

```
#Quiz 07.05
import pandas as pd
import statistics

df=pd.DataFrame({"col1":[1,2],"col2":[3,4],"col3":[5,6],"col4":[7,8]})

#comulative product
a=df.cumprod(axis=1)
print(a)
df

```

```
   col1  col2  col3  col4
0     1     3    15   105
1     2     8    48   384
```

Out[161]:

	col1	col2	col3	col4
0	1	3	5	7
1	2	4	6	8