

# Laboratorios de computación salas A y B

---

*Profesor:* Marco Antonio Quintana

*Asignatura:* Estructura de datos y algoritmos I

*Grupo:* 17

*No de Práctica(s):* 11

*Integrante(s):* Pascual Hernández Daniel Alfonso

*No. de Equipo de  
cómputo empleado:* ----

*No. de Lista o Brigada:* 27

*Semestre:* Segundo

*Fecha de entrega:* 27/mayo/2020

*Observaciones:*

CALIFICACIÓN: \_\_\_\_\_

## Objetivo

Poder crear algoritmos de efectividad usando algunos metodos de diseño

## Introducción

Fuerza bruta: se refiere hacer todas las combinaciones posibles, como en una contraseña de 3 dígitos, usamos la combinación de números desde 000 hasta 999

Algoritmos ávidos: se toma una decisión óptima a partir de los procesos

Botton – up: separar el problema general en subproblemas y de la solución de estos encontrar en del general

Top-dow: vamos calculado desde n hacia abajo guardando los resultados para no repetirlos

Incremental:vamos dando un resultado de manera paulatina ,el cual debe ser correcto y en cada iteración va dando más información de la respuesta

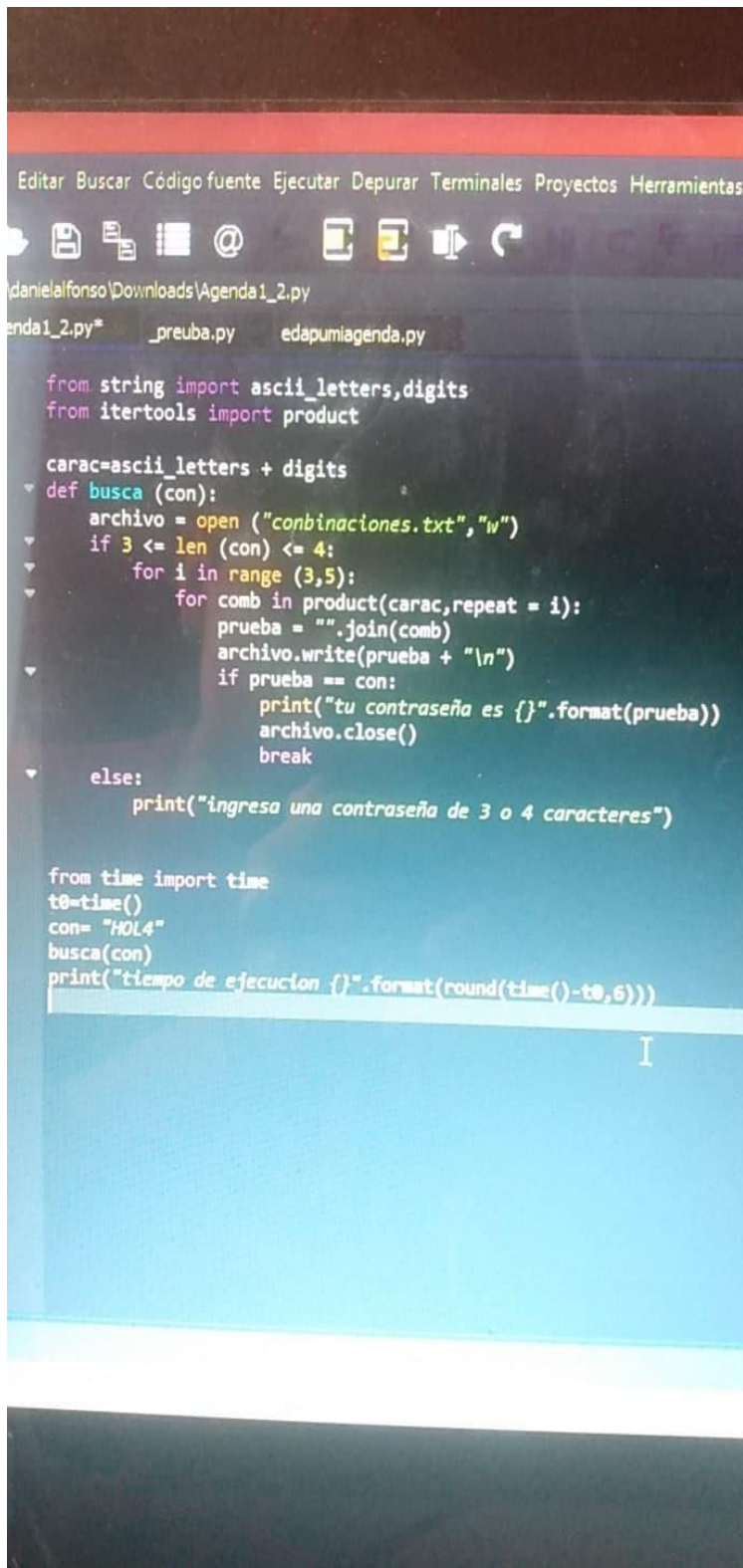
Divide y venceras: de un problema grande crear pequeños problemas e ir solucionando uno a uno

Mención y graficas en tiempo de ejecución: usar las gráficas que nos otorgan pyton para el análisis

Modelo RAM :contamos el tiempo de ejecución y cuantas veces se hizo alguna iteración

## Desarrollo

# Programa 1

A photograph of a computer screen displaying a Python IDE. The IDE has a menu bar with 'Editar', 'Buscar', 'Código fuente', 'Ejecutar', 'Depurar', 'Terminales', 'Proyectos', and 'Herramientas'. Below the menu is a toolbar with icons for file operations. The file explorer on the left shows the file path 'danielalfonso\Downloads\Agenda1\_2.py' and a list of files: 'Agenda1\_2.py\*', '\_preuba.py', and 'edapumiagenda.py'. The main editor window shows a Python script. The script imports 'ascii\_letters' and 'digits' from 'string', and 'product' from 'itertools'. It defines a function 'busca(con)' that opens a file 'combinaciones.txt' in write mode. The function checks if the length of 'con' is between 3 and 4. If so, it iterates from 3 to 5, and for each iteration, it generates all possible combinations of 'con' using 'product'. It then checks if the generated combination matches 'con'. If it does, it prints the password and closes the file. If not, it continues the loop. If the length of 'con' is not between 3 and 4, it prints a message to enter a password of 3 or 4 characters. At the bottom, it imports 'time', gets the start time 't0', calls 'busca(con)', and prints the execution time in seconds, rounded to 6 decimal places.

```
from string import ascii_letters,digits
from itertools import product

carac=ascii_letters + digits
def busca (con):
    archivo = open ("combinaciones.txt","w")
    if 3 <= len (con) <= 4:
        for i in range (3,5):
            for comb in product(carac,repeat = i):
                prueba = "".join(comb)
                archivo.write(prueba + "\n")
                if prueba == con:
                    print("tu contraseña es {}".format(prueba))
                    archivo.close()
                    break
    else:
        print("ingresa una contraseña de 3 o 4 caracteres")

from time import time
t0=time()
con= "HOL4"
busca(con)
print("tiempo de ejecucion {}".format(round(time()-t0,6)))
```

Explorador de variables Ayuda

Terminal 1/A

```
danielalfonso/Downloads')
Traceback (most recent call last):

File "C:/Users/danielalfonso/Downloads/Agenda1.py", line 1, in <module>
    busca(con)

File "C:/Users/danielalfonso/Downloads/Agenda1.py", line 2, in busca
    archivo.write(prueba + "\n")

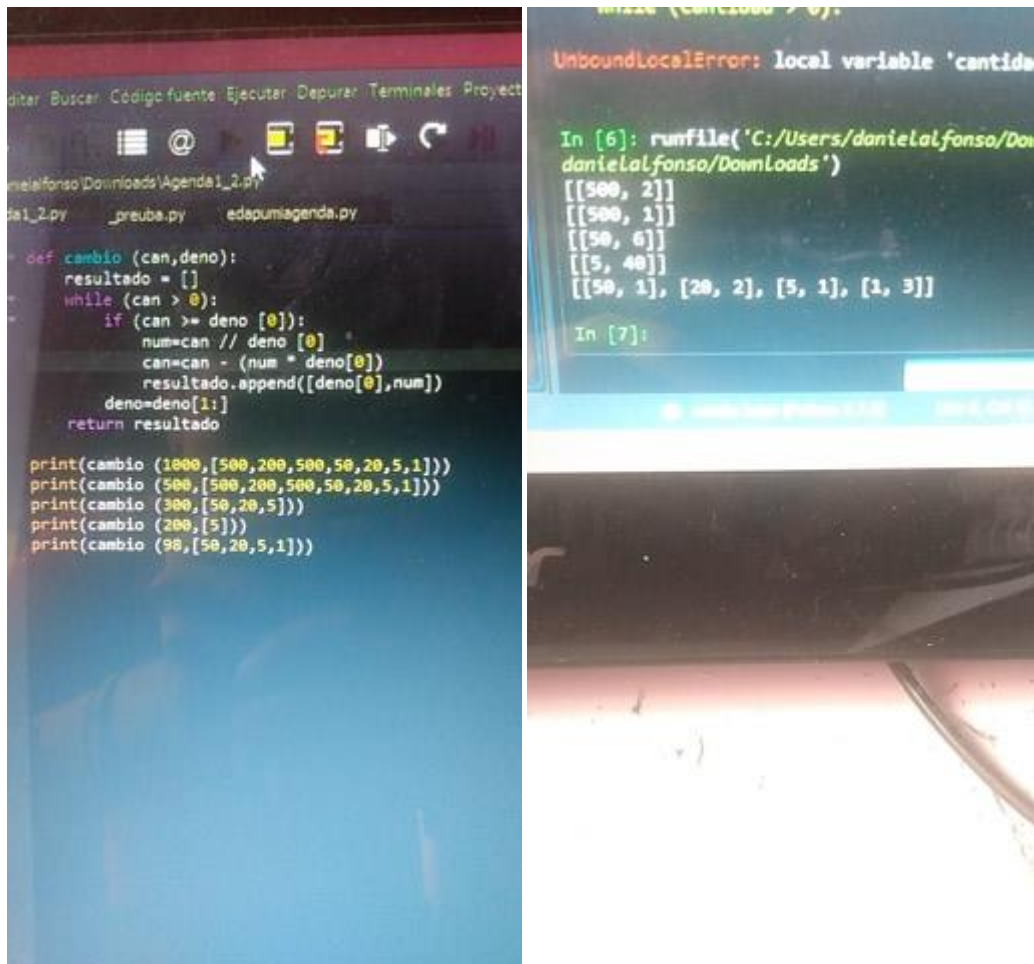
NameError: name 'archivo' is not defined

In [4]: runfile('C:/Users/danielalfonso/Downloads/Agenda1.py', wdir='C:/Users/danielalfonso/Downloads')
tu contrase a es H0L4
tiempo de ejecucion 59.363735

In [9]:
```

conda: base (Python 3.7.6) Line 25, Col 1

## Programa 2

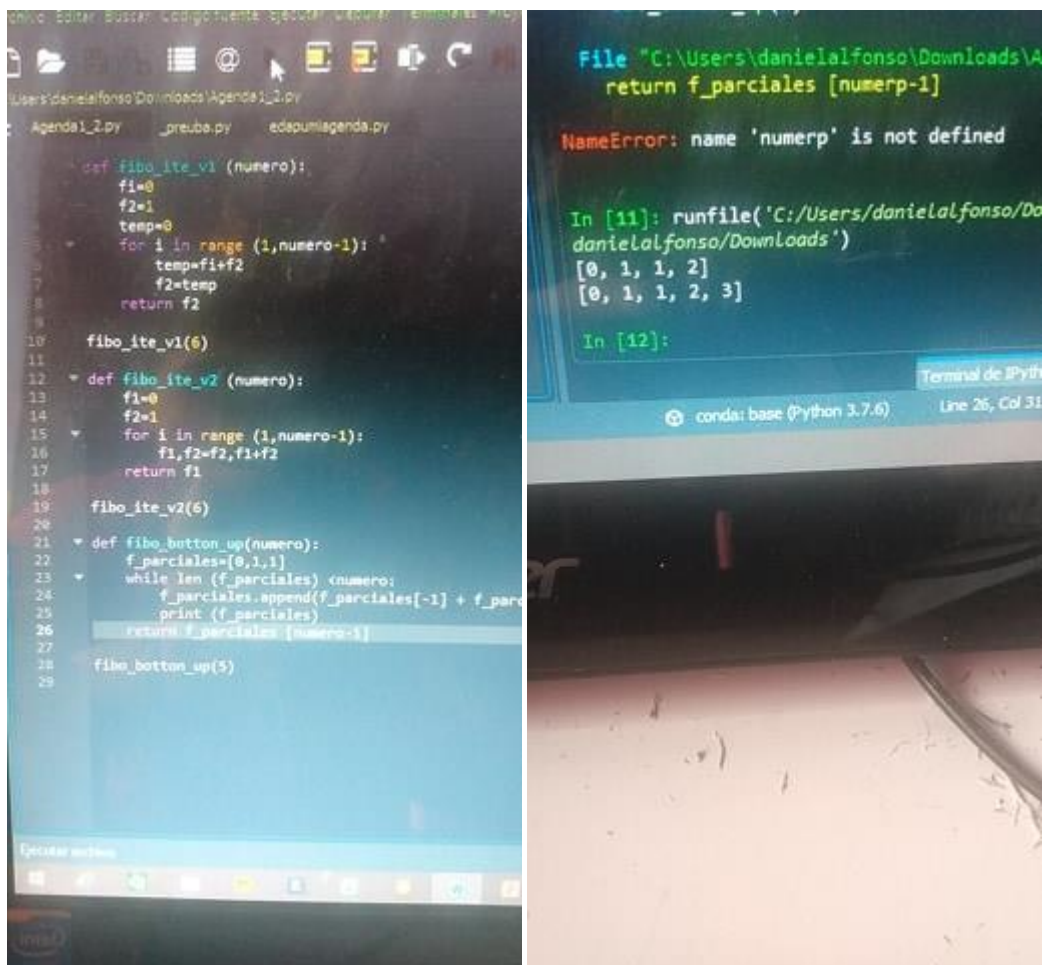


The image consists of two screenshots from a Python IDE. The left screenshot shows the source code editor with a file named 'Agenda1\_2.py'. The code defines a function 'cambio' that takes 'can' and 'deno' as arguments. It uses a while loop to process a list of denominations. The right screenshot shows the IPython console output, displaying the results of the 'cambio' function calls for various inputs, including a list of denominations [500, 200, 500, 50, 20, 5, 1].

```
def cambio (can,deno):  
    resultado = []  
    while (can > 0):  
        if (can >= deno [0]):  
            num=can // deno [0]  
            can=can - (num * deno[0])  
            resultado.append([deno[0],num])  
            deno=deno[1:]  
        return resultado  
  
print(cambio (1000,[500,200,500,50,20,5,1]))  
print(cambio (500,[500,200,500,50,20,5,1]))  
print(cambio (300,[50,20,5]))  
print(cambio (200,[5]))  
print(cambio (90,[50,20,5,1]))
```

```
UnboundLocalError: local variable 'cantidad'  
  
In [6]: runfile('C:/Users/danielalfonso/Down  
danielalfonso/Downloads')  
[[500, 2]]  
[[500, 1]]  
[[50, 6]]  
[[5, 40]]  
[[50, 1], [20, 2], [5, 1], [1, 3]]  
  
In [7]:
```

## Programa3



The image shows a Python IDE with a file explorer at the top displaying files: `Agenda1_1.py`, `_preuba.py`, and `edapumilagenda.py`. The main editor window contains the following Python code:

```
def fibo_ite_v1(numero):  
    f1=0  
    f2=1  
    temp=0  
    for i in range(1,numero-1):  
        temp=f1+f2  
        f2=temp  
    return f2  
  
fibo_ite_v1(6)  
  
def fibo_ite_v2(numero):  
    f1=0  
    f2=1  
    for i in range(1,numero-1):  
        f1,f2=f2,f1+f2  
    return f1  
  
fibo_ite_v2(6)  
  
def fibo_bottom_up(numero):  
    f_parciales=[0,1,1]  
    while len(f_parciales)<numero:  
        f_parciales.append(f_parciales[-1] + f_parciales[-2])  
        print(f_parciales)  
    return f_parciales[numero-1]  
  
fibo_bottom_up(5)
```

Below the code editor is a terminal window titled "Terminal de IPython" with the following output:

```
File "C:\Users\danielalfonso\Downloads\A  
return f_parciales [numerp-1]  
  
NameError: name 'numerp' is not defined  
  
In [11]: runfile('C:/Users/danielalfonso/Da  
danielalfonso/Downloads')  
[0, 1, 1, 2]  
[0, 1, 1, 2, 3]  
  
In [12]:
```

The terminal also shows the environment as `conda: base (Python 3.7.6)` and the current line and column as `Line 26, Col 31`.

## Programa 4

```
agenda1_2.py
agenda1_2.py
preuba.py
edspumagenda.py

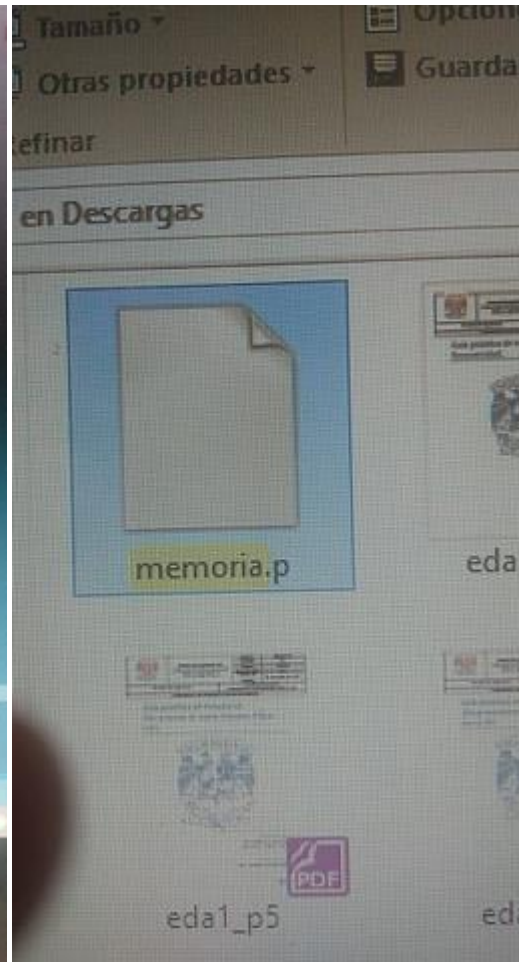
memoria={1:0,2:1,3:1}
def fibo_ite_v2(numero):
    f1=0
    f2=1
    for i in range(1,numero-1):
        f1,f2=f2,f1+f2
    return f1

fibo_ite_v2(6)

def fibo_top_dow(numero):
    if numero in memoria:
        return memoria[numero]
    f=fibo_ite_v2(numero-1) + fibo_ite_v2(numero-2)
    memoria[numero]=f
    return memoria[numero]

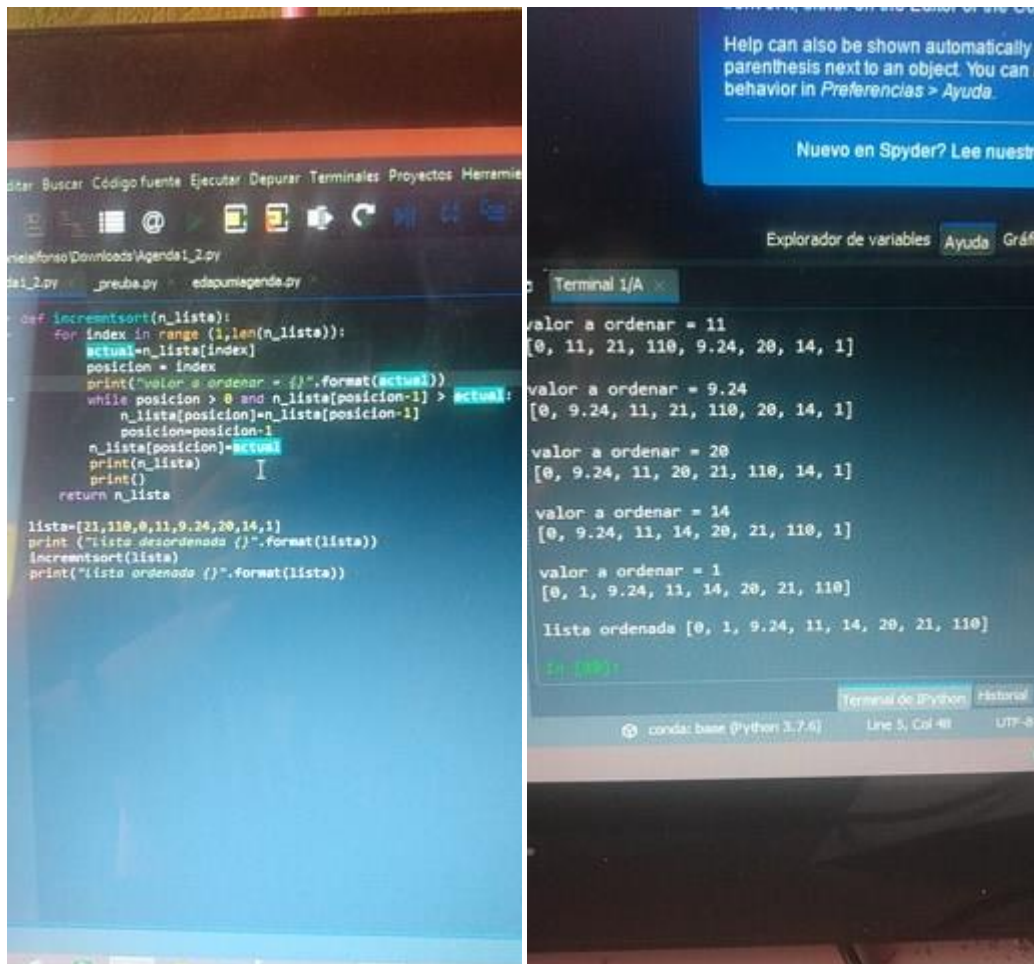
fibo_top_dow(12)
memoria
fibo_top_dow(8)
memoria
import pickle
archivo=open('memoria.p','wb')
pickle.dump(memoria,archivo)
archivo.close()

archivo=open('memoria.p','rb')
memoria_de_archivo=pickle.load(archivo)
archivo.close()
memoria
memoria_de_archivo
```





## Programa 5



The image shows a screenshot of the Spyder IDE interface. The left pane displays a Python script named `edapumilagenda.py` with a bubble sort algorithm. The right pane shows the execution output in the `Terminal 1/A` window.

**Python Code:**

```
def incrementosort(n_lista):
    for index in range(1, len(n_lista)):
        actual = n_lista[index]
        posicion = index
        print("valor a ordenar = {}".format(actual))
        while posicion > 0 and n_lista[posicion-1] > actual:
            n_lista[posicion] = n_lista[posicion-1]
            posicion = posicion-1
        n_lista[posicion] = actual
        print(n_lista)
        print()
    return n_lista

lista = [21, 110, 0, 11, 9.24, 20, 14, 1]
print("Lista desordenada {}".format(lista))
incrementosort(lista)
print("Lista ordenada {}".format(lista))
```

**Terminal Output:**

```
valor a ordenar = 11
[0, 11, 21, 110, 9.24, 20, 14, 1]

valor a ordenar = 9.24
[0, 9.24, 11, 21, 110, 20, 14, 1]

valor a ordenar = 20
[0, 9.24, 11, 20, 21, 110, 14, 1]

valor a ordenar = 14
[0, 9.24, 11, 14, 20, 21, 110, 1]

valor a ordenar = 1
[0, 1, 9.24, 11, 14, 20, 21, 110]

Lista ordenada [0, 1, 9.24, 11, 14, 20, 21, 110]
```

The bottom status bar indicates the environment is `conda: base (Python 3.7.4)` at `Line 5, Col 48` with `UTF-8` encoding.



## Programa 6

```
def quicksort_aux(lista, inicio, fin):
    if inicio < fin:
        pivote = particion(lista, inicio, fin)
        quicksort_aux(lista, inicio, pivote-1)
        quicksort_aux(lista, pivote+1, fin)

def particion(lista, inicio, fin):
    pivote = lista[inicio]
    print("el valor del vipote {}".format(pivote))
    izquierda = inicio+1
    derecha = fin
    print("indice izquierdo {}".format(izquierda))
    print("indice derecho {}".format(derecha))

    bandera = False
    while not bandera:
        while izquierda <= derecha and lista[izquierda] <= pivote:
            izquierda = izquierda+1
        while lista[derecha] >= pivote and derecha >= izquierda:
            derecha = derecha-1
        if derecha < izquierda:
            bandera = True
        else:
            temp = lista[izquierda]
            lista[izquierda] = lista[derecha]
            lista[derecha] = temp
    print(lista)

    temp = lista[inicio]
    lista[inicio] = lista[derecha]
    lista[derecha] = temp
    return derecha

lista = [21, 10, 0, 11, 9, 1, 20, 14, 1]
print("lista desordenada {}".format(lista))
quicksort(lista)
print("lista ordenada {}".format(lista))
```

Terminal 1/A

```
indice izquierdo1
indice derecho 8
[21, 10, 0, 11, 9, 1, 20, 14, 24]
el valor del vipote 14
indice izquierdo1
indice derecho 6
[14, 10, 0, 11, 9, 1, 20, 21, 24]
el valor del vipote 1
indice izquierdo1
indice derecho 4
[1, 0, 10, 11, 9, 14, 20, 21, 24]
el valor del vipote 10
indice izquierdo3
indice derecho 4
[0, 1, 10, 9, 11, 14, 20, 21, 24]
lista ordenada [0, 1, 9, 10, 11, 14, 20, 21, 24]
```

codigo: base (Python 3.7.6) Line 25, Col 9

## Programa 7



```
File Edit Search Code Fuente Ejecutar Debugar Terminales Proyecto Herramientas
pydenialforso Downloads\Agenda1_1.py
Agenda1_1.py _preuba.py edaquiagenda.py

#Ayuda inline
from matplotlib.pyplot import plt
from mpl_toolkits.mplot3d import Axes3D

import random
from time import time

from quicksort import quicksort_time

datos = [i*i*100 for i in range(1,11)]
tiempo_is = []
tiempo_qs = []

for i1 in datos :
    lista_is = random.sample((range(0,100000000),i1))
    lista_qs = lista_is.copy()

    t0 = time()
    insertion_sort_time(lista_is)
    tiempo_is.append(round((time()-t0),6))

    t0 = time()
    insertion_sort_time(lista_qs)
    tiempo_qs.append(round((time()-t0),6))

print("tiempos parciales en ejecucion en INSERT SORT (i)[5] ")
print("tiempos parciales en ejecucion en QUICK SORT (i)[5] ")

print("tiempos total en ejecucion en INSERT SORT (i)[5] ")
print("tiempos total en ejecucion en QUICK SORT (i)[5] ")
```

## Programa 7

```

import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

times=0
def insertion_sort_graph(n_lista):
    global time
    for index in range(1, len(n_lista)):
        times += 1
        actual = n_lista[index]
        position = index
        while position > 0 and n_lista[position-1] > actual:
            times += 1
            n_lista[position] = n_lista[position-1]
            position = position - 1
        n_lista[position] = actual
    return n_lista

TAM=101
eje_x=list(range(1, TAM, 1))
eje_y=[]
lista_variable=[]

for num in eje_x:
    lista_variable=random.sample(range(0,1000),num)
    times=0
    lista_variable=insertion_sort_graph(lista_variable)
    eje_y.append(times)

fig=plt.subplots(facecolor='w', edgecolor='k')
ax=plt.subplot(eje_x, eje_y, marker='o', color='b', linestyle='None')
ax.set_xlabel('x')
ax.set_ylabel('y')
ax.grid(True)
ax.legend(['insertion sort'])
ax.title(['insertion sort'])
plt.show()
```

## Programa 8



```
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

times=0
def insertSort_graph(n_lista):
    global time
    for index in range(1,len(n_lista)):
        times += 1
        actual = n_lista[index]
        posicion = index
        while posicion > 0 and n_lista[posicion-1] > actual :
            times+=1
            n_lista[posicion]=n_lista[posicion-1]
            posicion=posicion-1
        n_lista[posicion]=actual
    return n_lista

TAM=101
eje_x=list(range(1,TAM,1))
eje_y=[]
lista_variable=[]

for num in eje_x:
    lista_variable=random.sample(range(0,1000),num)
    times=0
    lista_variable=insertSort_graph(lista_variable)
    eje_y.append(times)

fig,ax=plt.subplots(facecolor='w',edgecolor='k')
ax.plot(eje_x,eje_y,marker='o',color='b',linestyle='None')

ax.set_xlabel('x')
ax.set_ylabel('y')
ax.grid(True)
ax.legend(['insertion sort'])
ax.title(['insertion sort'])
```

## Conclusiones:

Los objetivos se pudieron cumplir de manera adecuada, logre ver cosas muy interesantes para poder desarrollar mis códigos de formas dintas dependiendo las necesidades del mismo, incluso practica con el mismo problema pero que la solución sea de manera distinta.

## Referencias

<http://lcp02.fi-b.unam.mx/>