UNIVERSIDAD AUTONOMA DE NUEVOLEON

FACULTAD DE CIENCIAS FISICO MATEMATICAS

‘‘ALGORITMOS DE ORDENAMIENTO’’

ALUMNA: MELINA PLATAS SANCHEZ

MATRICULA: 1740967

GRUPO: 001

AULA: 120

MATERIA: MATEMATICAS COMPUTACIONALES

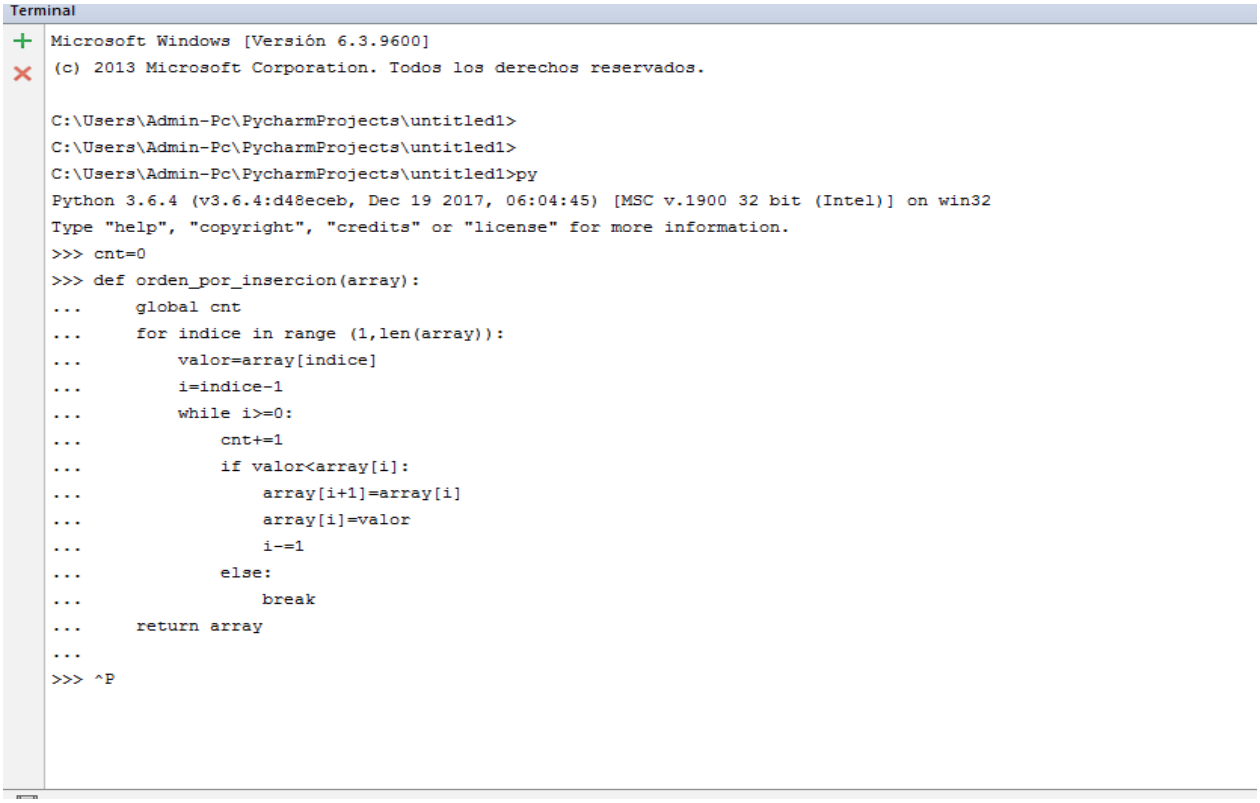
MAESTRO: LIC. JOSE ANASTACIO HERNANDEZ SALDAÑA

FECHA: 11 DE ABRIL 2018

Reporte de algoritmos

El algoritmo de **inserción** es tener dos arreglos en el cual uno no este ordenado y el otro sí. Así comenzamos con el algoritmo, tomamos el primer elemento el cual está ordenado e iteramos sobre el arreglo desordenado tomando un elemento a la vez y cuando tomemos ese elemento lo insertamos en un arreglo ordenado, así podremos tener el elemento que está desordenado en el arreglo ordenado de tal modo que ese elemento esté en su posición que deba de ir, de este modo el algoritmo acaba cuando el arreglo desordenado esté vacío sin ningún elemento.

El algoritmo que encontré para el algoritmo de incersión es el siguiente:



Quicksort es el algoritmo de ordenación más rápido conocido, su tiempo de ejecución promedio es O(n log (n)), siendo en el peor de los casos O(n2), caso altamente improbable. El hecho de que sea más rápido que otros algoritmos de ordenación con tiempo promedio de O(n log (n)) viene dado por que QuickSort realiza menos operaciones ya que el método utilizado es el de partición. Una explicación de este algoritmo es que se elige un elemento v de la lista L de elementos al que se le llama pivote. Se particiona la lista L en tres listas: L1 - que contiene todos los elementos de L menos v que sean menores o iguales que v. L2 - que contiene a v. L3 - que contiene todos los elementos de L menos v que sean mayores o iguales que v. Se aplica la recursión sobre L1 y L3. Se unen todas las soluciones que darán forma final a la lista L finalmente ordenada. Como L1 y L3 están ya ordenados, lo único que tenemos que hacer es concatenar L1, L2 y L3.

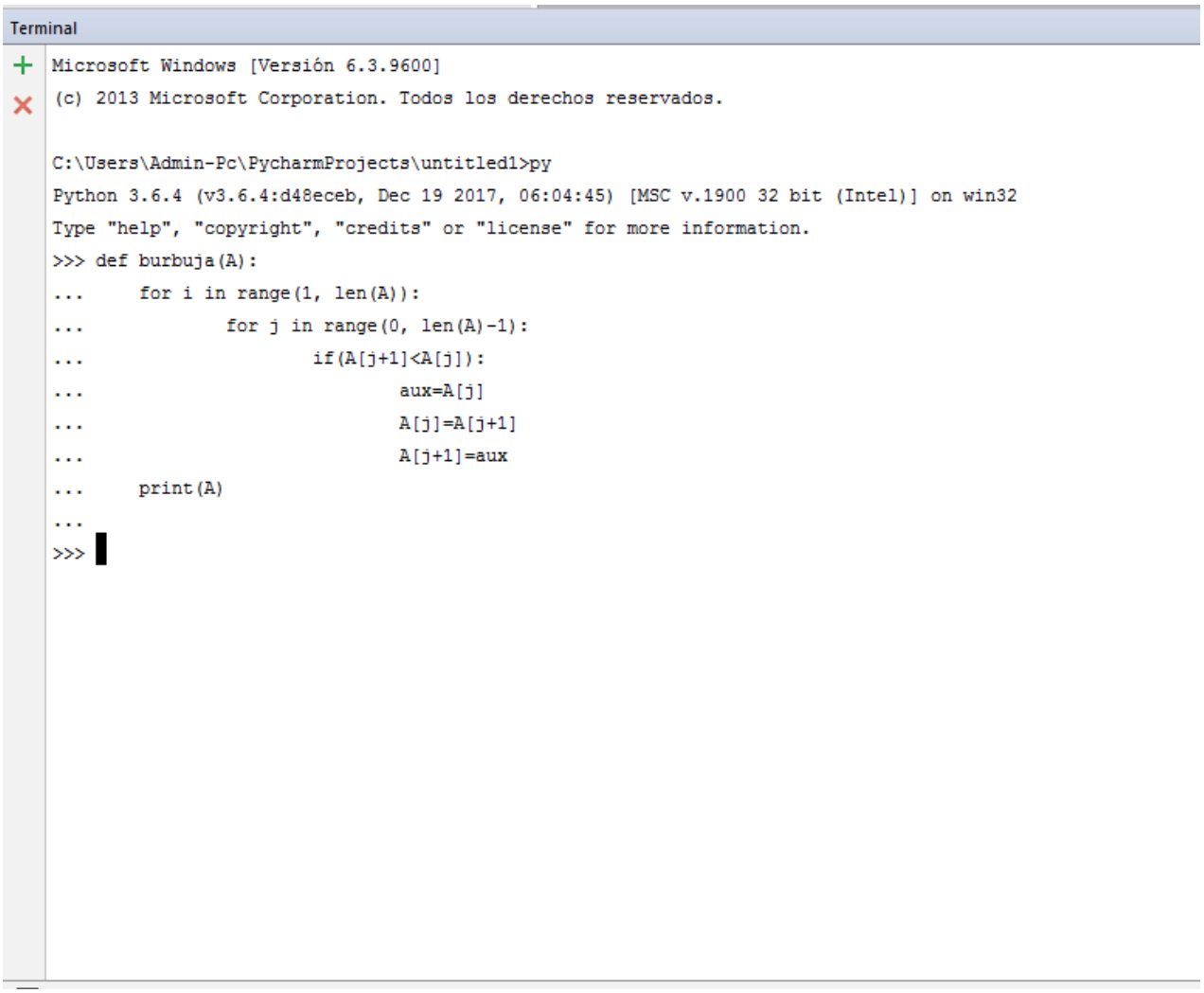
El algoritmo que encontré para el algoritmo de quickort es el siguiente:

```
Microsoft Windows [Versión 6.3.9600]
(c) 2013 Microsoft Corporation. Todos los derechos reservados.

C:\Users\Admin-Pc\PycharmProjects\untitled1>py
Python 3.6.4 (v3.6.4:d48eceb, Dec 19 2017, 06:04:45) [MSC v.1900 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> def quicksort(arr):
...     global cnt
...     if len(arr) <= 1:
...             return arr
...     p=arr.pop(0)
...     menores, mayores=[], []
...     for e in arr:
...             cnt+=1
...             if e <=p:
...                     menores.append(e)
...             else:
...                     mayores.append(e)
...     return quicksort(menores) +[p] + quicksort(mayores)
...
>>>
```

Este algoritmo llamado **Burbuja** me tocó junto con mi compañero Sarai en explicar a la clase y lo que sé ahora es que funciona comparando en él dos elementos adyacentes, si el segundo elemento es menor al primero entonces son intercambiados, en caso contrario se evalúa el segundo elemento y el siguiente elemento adyacente a su izquierda. Ya cuando esté terminada la primera iteración se vuelve a realizar el mismo procedimiento sin embargo en cada iteración hecha, un elemento de derecha a izquierda, ya no es evaluado ya que contiene el número mayor de cada iteración. El algoritmo terminará una vez que el número de iteraciones sea igual al número de elementos que contenga.

El algoritmo que encontré para el algoritmo de burbuja el siguiente:



En este último algoritmo que es **selection** lo que hace es que la función principal, es la encargada de recorrer la lista, ubicando el mayor elemento al final del segmento y luego reduciendo el segmento a analizar. También busca el mayor de todos los elementos de la lista. Poner el mayor al final (intercambiar el que está en la última posición de la lista con el mayor encontrado). Buscar el mayor de todos los elementos del segmento de la lista entre la primera y la anteúltima posición. Poner el mayor al final del segmento (intercambiar el que está en la última posición del segmento, o sea anteúltima posición de la lista, con el mayor encontrado). Se termina cuando queda un único elemento sin tratar: el que está en la primera posición de la lista, y que es el menor de todos porque todos los mayores fueron reubicados.

El algoritmo que encontré para el algoritmo de selection es el siguiente: