Programa 14

Descripción del problema:

Se desea realizar un programa en el que el usuario vaya introduciendo números hasta que introduzca algo que no lo sea. Una vez introducidos, se añadirá a cada número la información de si es primo o no, el número de cifras que tiene y si es el mínimo o el máximo. Al final, sacará todo por pantalla. Es necesario usar un diccionario, listas y funciones.

Materia nueva:

En este caso, nos falta por saber cómo conocer la longitud de una cadena (es una de las múltiples funciones que nos quedan en el tintero). Dicha función es:

len(string**)** Aquí hay una curiosidad. Si la mayor parte eran string**.función**, ¿por qué no hay ningún punto? Pues sencillamente porque esta función se puede usar para un montón de cosas aparte de una Cadena de caracteres. De momento la usaremos para conocer cuántos dígitos tiene el número que el usuario ha introducido.

NOTA:

- **Modo avanzado:** Recuerda que, para cada variable nueva tipo diccionario, es necesario declararla para que no apunte a la misma dirección de memoria.
- **Modo terrestre:** Mete la declaración en el bucle para que se declare y sea nueva cada vez que se usa.

Solución

Algoritmo:

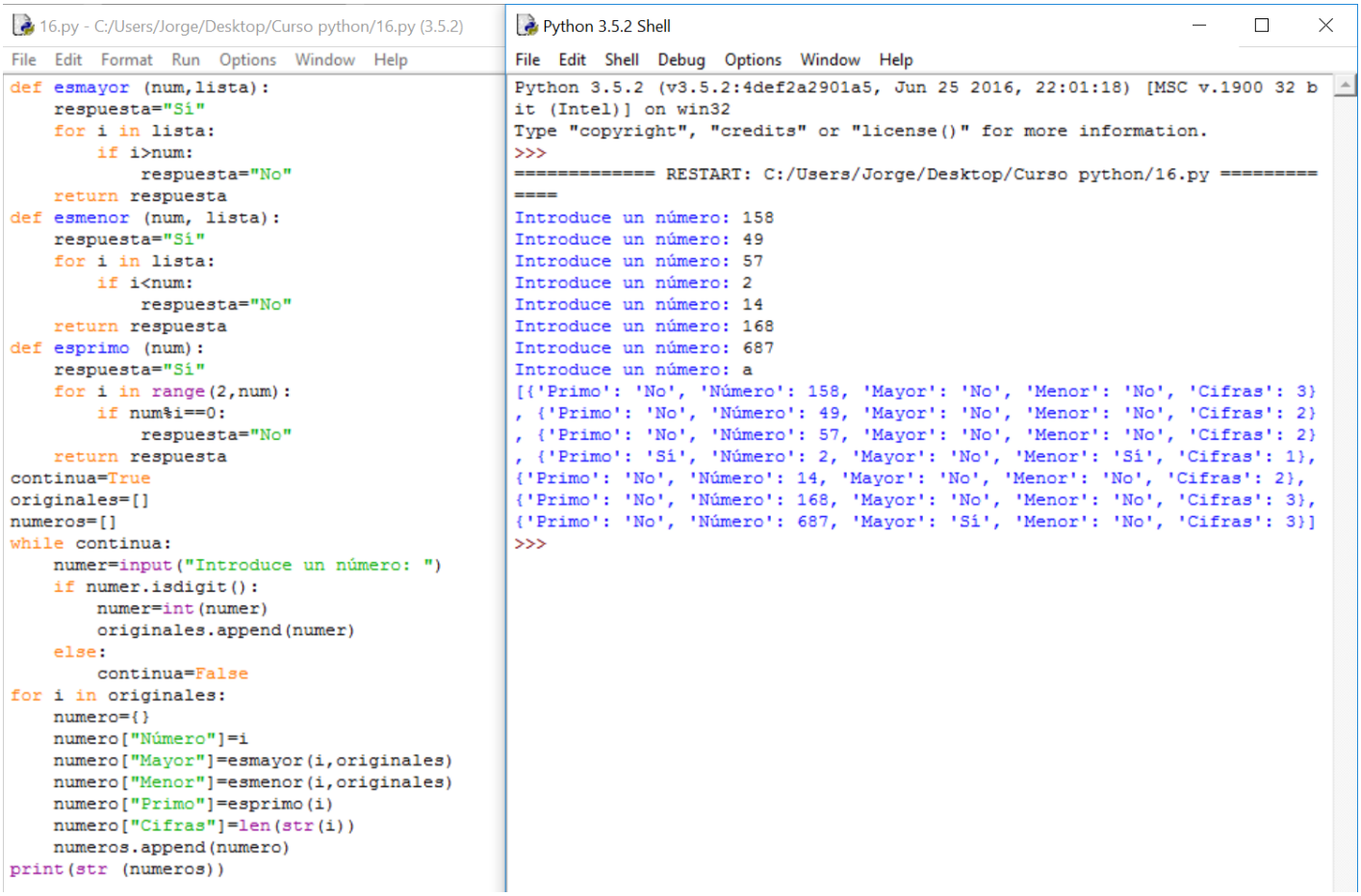
1.- Pedir números hasta que haya una introducción que no se pueda convertir en uno. Ir guardándolos en una lista

2.- Para cada uno de ellos:

- 2.1.- Guardarlo bajo la "Key": "Número"
- 2.2.- Comprobar si es menor, mayor y primo y guardarlo bajo las "Keys": "Menor", "Mayor" y "Primo"
- 2.3.- Contar con la función **len(** s **)** la cantidad de caracteres del número pasado a Cadena de caracteres y guardarlo bajo la "Key": "Cifras"

3.- Sacar la lista de diccionarios por la pantalla.

Solución:



```
16.py - C:/Users/Jorge/Desktop/Curso python/16.py (3.5.2)
File  Edit  Format  Run  Options  Window  Help
def esmayor (num,lista):
    respuesta="Sí"
    for i in lista:
        if i>num:
            respuesta="No"
    return respuesta
def esmenor (num, lista):
    respuesta="Sí"
    for i in lista:
        if i<num:
            respuesta="No"
    return respuesta
def esprimo (num):
    respuesta="Sí"
    for i in range(2,num):
        if num%i==0:
            respuesta="No"
    return respuesta
continua=True
originales=[]
numeros=[]
while continua:
    numer=input("Introduce un número: ")
    if numer.isdigit():
        numer=int(numer)
        originales.append(numer)
    else:
        continua=False
for i in originales:
    numero={}
    numero["Número"]=i
    numero["Mayor"]=esmayor(i,originales)
    numero["Menor"]=esmenor(i,originales)
    numero["Primo"]=esprimo(i)
    numero["Cifras"]=len(str(i))
    numeros.append(numero)
print(str (numeros))
```

```
Python 3.5.2 Shell
File  Edit  Shell  Debug  Options  Window  Help
Python 3.5.2 (v3.5.2:4def2a2901a5, Jun 25 2016, 22:01:18) [MSC v.1900 32 b
it (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: C:/Users/Jorge/Desktop/Curso python/16.py =========
====
Introduce un número: 158
Introduce un número: 49
Introduce un número: 57
Introduce un número: 2
Introduce un número: 14
Introduce un número: 168
Introduce un número: 687
Introduce un número: a
[{'Primo': 'No', 'Número': 158, 'Mayor': 'No', 'Menor': 'No', 'Cifras': 3}
, {'Primo': 'No', 'Número': 49, 'Mayor': 'No', 'Menor': 'No', 'Cifras': 2}
, {'Primo': 'No', 'Número': 57, 'Mayor': 'No', 'Menor': 'No', 'Cifras': 2}
, {'Primo': 'Sí', 'Número': 2, 'Mayor': 'No', 'Menor': 'Sí', 'Cifras': 1},
{'Primo': 'No', 'Número': 14, 'Mayor': 'No', 'Menor': 'No', 'Cifras': 2},
{'Primo': 'No', 'Número': 168, 'Mayor': 'No', 'Menor': 'No', 'Cifras': 3},
{'Primo': 'No', 'Número': 687, 'Mayor': 'Sí', 'Menor': 'No', 'Cifras': 3}]
>>>
```

Explicación:

En este caso, en las órdenes:

numero["Mayor"]=esmayor(i,originales) y esmenor(i,originales) podemos ver que se ha trasladado a la función una lista entera.

Podría parecer que sólo admitiría variables simples pero, realmente, no existe límite de datos o variables a la hora de pasar a las funciones.

Sin embargo, cada función usa una lista diferente. Sí, en Python no se permite usar el original de ninguna variable enviada a una función. Si nos pusiéramos puristas, esto duplicaría la cantidad de memoria reservada en el ordenador a nuestro programa; es verdad, pero la cantidad de memoria y la capacidad de cálculo han dejado de ser un problema hace mucho tiempo.

Respecto a la función **len(**s**)**, ha habido que pasarlo primero a Cadena de caracteres para poder realizarlo.

Revision #2
Created 1 February 2022 11:42:15 by Equipo CATEDU
Updated 1 February 2022 11:42:17 by Equipo CATEDU