Actividad 01 - Características principales de Python

Paradigmas de la Programación

Carlos Augusto Zayas Guggiari

14 de marzo de 2024

Resumen

Con la ayuda del contenido de este documento, el alumno resolverá un sencillo ejercicio mientras empieza a conocer las características principales del lenguaje Python y cómo éste implementa las estructuras lógicas fundamentales.

Contenido

1 Objetivos

1. Conocer algunas características básicas del lenguaje Python:
   - Tipos básicos y Operadores.
   - Entrada y salida estándar.
2. Aprender cómo se implementan en Python las estructuras lógicas y los subprogramas:
   - Secuencia, condición, repetición.
   - Definición de funciones.
3. Comprender el valor del aporte del paradigma estructurado a la programación puramente imperativa.

2 Programación imperativa

2.1 La Commodore 64

Según Guinness World Records, la computadora de escritorio más vendida de todos los tiempos es la **Commodore 64**, que fue fabricada por Commodore International (USA) entre agosto de 1982 y abril de 1994. El número exacto de *C-64* vendidas no está claro: el fundador de Commodore, Jack Tramiel, estimó entre 22 y 30 millones de unidades. La estimación oficial de Commodore era de 17 millones de unidades, mientras que una estimación moderna creíble sitúa la cifra en alrededor de 12,5 millones de unidades. A pesar de ello, la C-64 sigue considerándose el modelo único más exitoso de computadora de escritorio de la historia, incluso con las cifras más conservadoras.

La Commodore 64 con diskettera y monitor (vendidos por separado).

2.2 Un ejemplo en BASIC de hace más de 40 años

El manual que acompañaba a la C-64 se titulaba **Commodore 64 User's Guide** (Guía del Usuario de la Commodore 64) y servía como una introducción al manejo del equipo y sobre todo al lenguaje de programación Microsoft BASIC 2.0 incorporado, que para la época se consideraba lo que ahora se conoce como interfaz del sistema operativo.

Uno de los programas de ejemplo contenidos en el material introductorio mencionado era un juego de adivinanzas, donde el jugador intentaba adivinar un número elegido al azar por la computadora.

Como se puede notar, las líneas del programa están numeradas, como era habitual en los lenguajes de programación de la época, y si el lenguaje no contaba con estructuras lógicas más adecuadas (como en este caso) el número de línea se usaba como referencia para implementar la repetición de sentencias.

Con ayuda del emulador VICE el profesor mostrará en clase el código fuente y la ejecución de una versión del juego.

GUESSING GAME

Since we've gone to some lengths to understand random numbers, why not put this information to use? The following game not only illustrates a

50

good use of random numbers, but also introduces some additional programming theory.

In running this program, a random number, NM, will be generated.

NEW

; INDICATES NO SPACE AFTER QUOTATION MARK

```
1 REM NUMBER GUESSING GAME
2 PRINT "{CLR/HOME}"
5 INPUT "ENTER UPPER LIMIT FOR GUESS ";LI
10 NM = INT(LI*RND(1))+1
15 CN = 0
20 PRINT "I'VE GOT THE NUMBER.":PRINT
30 INPUT "WHAT'S YOUR GUESS"; GU
35 CN = CN + 1
40 IF GU > NM THEN PRINT "MY NUMBER IS
LOWER": PRINT : GOTO 30
50 IF GU < NM THEN PRINT "MY NUMBER IS
HIGHER": PRINT : GOTO 30
60 PRINT "GREAT! YOU GOT MY NUMBER"
65 PRINT "IN ONLY "; CN ;"GUESSES.":PRINT
70 PRINT "DO YOU WANT TO TRY ANOTHER (Y/N)?";
80 GET AN$ : IF AN$="" THEN 80
90 IF RN$ = "Y" THEN 2
100 IF AN$ <> "N" THEN 70
110 END
```

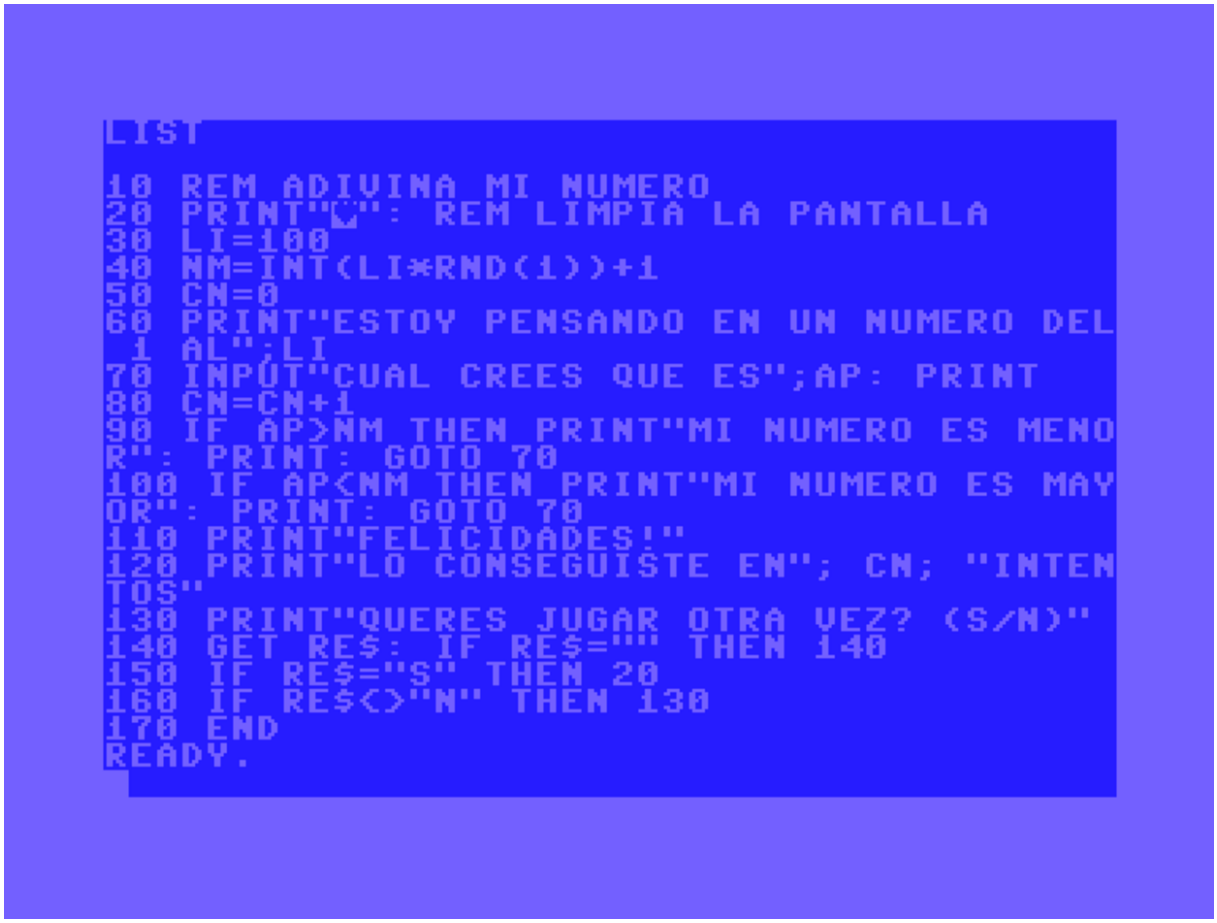
You can specify how large the number will be at the start of the program. Then, it's up to you to guess what the number is.

A sample run follows along with an explanation:

```
ENTER UPPER LIMIT FOR GUESS? 25
I'VE GOT THE NUMBER.

WHAT'S YOUR GUESS? 15
MY NUMBER IS HIGHER

WHAT'S YOUR GUESS? 20
MY NUMBER IS HIGHER

WHAT'S YOUR GUESS? 23
GREAT! YOU GOT MY NUMBER
IN ONLY 3 GUESSES

DO YOU WANT ANOTHER TRY (Y/N)
```

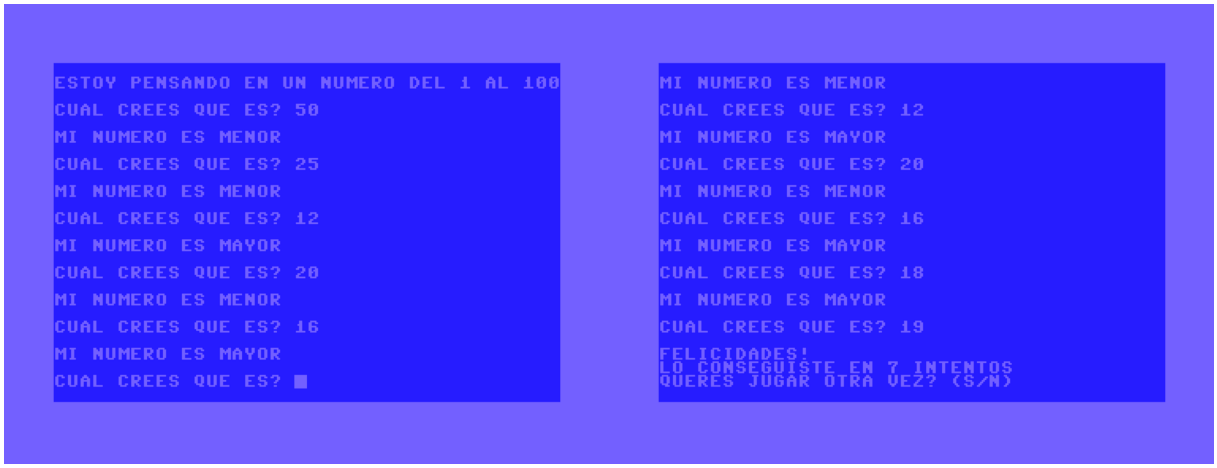
51

El listado en BASIC del juego de adivinanzas, tal y como aparece en la Guía del Usuario de la C-64.



```
LIST

10 REM ADIVINA MI NUMERO
20 PRINT"□": REM LIMPIA LA PANTALLA
30 LI=100
40 NM=INT(LI*RND(1))+1
50 CN=0
60 PRINT"ESTOY PENSANDO EN UN NUMERO DEL
 1 AL";LI
70 INPUT"CUAL CREES QUE ES";AP: PRINT
80 CN=CN+1
90 IF AP>NM THEN PRINT"MI NUMERO ES MENO
R": PRINT: GOTO 70
100 IF AP<NM THEN PRINT"MI NUMERO ES MAY
OR": PRINT: GOTO 70
110 PRINT"FELICIDADES!"
120 PRINT"LO CONSEGUISTE EN"; CN; "INTEN
TOS"
130 PRINT"QUERES JUGAR OTRA VEZ? (S/N)"
140 GET RE$: IF RE$="" THEN 140
150 IF RE$="S" THEN 20
160 IF RE$<>"N" THEN 130
170 END
READY.
```

Versión del juego escrita por el profesor.



```
ESTOY PENSANDO EN UN NUMERO DEL 1 AL 100
CUAL CREES QUE ES? 50
MI NUMERO ES MENOR
CUAL CREES QUE ES? 25
MI NUMERO ES MENOR
CUAL CREES QUE ES? 12
MI NUMERO ES MAYOR
CUAL CREES QUE ES? 20
MI NUMERO ES MENOR
CUAL CREES QUE ES? 16
MI NUMERO ES MAYOR
CUAL CREES QUE ES? ■
```

```
MI NUMERO ES MENOR
CUAL CREES QUE ES? 12
MI NUMERO ES MAYOR
CUAL CREES QUE ES? 20
MI NUMERO ES MENOR
CUAL CREES QUE ES? 16
MI NUMERO ES MAYOR
CUAL CREES QUE ES? 18
MI NUMERO ES MAYOR
CUAL CREES QUE ES? 19

FELICIDADES!
LO CONSEGUISTE EN 7 INTENTOS
QUERES JUGAR OTRA VEZ? (S/N)
```

Sesión de ejemplo con el juego “ADIVINA” para C-64.

3 Ejercicio: "Adivina Un Número"

El ejercicio de esta primera actividad consiste en hacer una versión en Python del juego de adivinanzas, aprovechando las ventajas de la programación estructurada.

El profesor irá mostrando paso a paso las sentencias en Python equivalentes a las que se utilizaron en el BASIC antiguo, de manera a reemplazar los saltos referenciales (GOTO) por las estructuras lógicas adecuadas.

3.1 Ejemplos en Python

3.1.1 Comentarios

Agregar comentarios a nuestro código es una buena costumbre, siempre y cuando éstos sean claros y útiles.

```
# Comentario de una sola línea

a = 1 # Al objeto 1 se le asigna el identificador a

"""
Comentario de más de una línea.
Pueden usarse comillas simples o dobles.
"""
```

3.1.2 Limpiar la pantalla

Python no incluye una sentencia para limpiar la pantalla de la terminal de línea de comandos, pero podemos crear la nuestra invocando al intérprete de línea de comandos del sistema operativo mediante el módulo `os` que forma parte de la Biblioteca Estándar de Python.

```
import os

def clear():
    '''Limpia la pantalla.'''
    os.system('cls' if os.name == 'nt' else 'clear')
```

3.1.3 Imprimir cosas (salida estándar)

La sentencia `print` de Python es bastante versátil, pero por ahora basta con este ejemplo:

```
print("Somos", 3, "elementos.")
```

3.1.4 Pedir cosas (entrada estándar)

La sentencia `input` sirve para solicitar datos al usuario:

```
usuario = input("¿Cómo te llamás?")
edad = int(input("¿Cuántos años tenés?")) # Convierte cadena a entero.
```

3.1.5 Generar un número aleatorio

El módulo `random` forma parte de la Biblioteca Estándar de Python.

```
import random

dado = random.randint(1, 6) # La variable dado va a tener un valor de 1 a 6.
```

3.1.6 Si condicional

Las estructuras condicionales pueden anidarse, aunque por claridad se recomienda modularizar el código.

```
if edad >= 18:
    procesar_inscripcion()
else:
    rechazar_inscripcion()

if opcion == 'A':
    opcion_a()
elif opcion == 'B':
    opcion_b()
else:
    ayuda()
```

3.1.7 Ciclo condicional

El ciclo `while` se repite mientras se cumpla una condición determinada.

```
while continuar == "S":
    hacer_esto_primero()
    hacer_esto_después()
    continuar = input("¿Continuar? S/N")

while True:
    esto_no_acaba()
```

4 Material disponible online para consultas relacionadas con Python

- El Tutorial de Python 3 - PDF
- Aprenda a Pensar como un Programador con Python - PDF
- Inmersión en Python 3 - PDF

Favor enviar sugerencias y correcciones a czayas@pol.una.py