Colección de actividades Aprender Conectados

Nivel Secundario

Programación

Cartas y digiamigos

Actividad N° 10

APRENDER CONECTADOS

Ministerio de Educación, Cultura, Ciencia y Tecnología
Presidencia de la Nación

Autoridades

Presidente de la Nación
Mauricio Macri

Jefe de Gabinete de Ministros
Marcos Peña

Ministro de Educación, Cultura, Ciencia y Tecnología
Alejandro Finocchiaro

Secretario de Gobierno de Cultura
Pablo Avelluto

Secretario de Gobierno de Ciencia, Tecnología e Innovación Productiva
Lino Barañao

Titular de la Unidad de Coordinación General del Ministerio de Educación, Cultura, Ciencia y Tecnología
Manuel Vidal

Secretaria de Innovación y Calidad Educativa
Mercedes Miguel

Subsecretario de Coordinación Administrativa
Javier Mezzamico

Directora Nacional de Innovación Educativa
María Florencia Ripani

ISBN en trámite

Este contenido fue producido por el Ministerio de Educación, Cultura, Ciencia y Tecnología de la Nación en el marco del Plan Aprender Conectados

Introducción

El Plan Aprender Conectados es la primera iniciativa en la historia de la política educativa nacional que se propone implementar un programa integral de alfabetización digital, con una clara definición sobre los contenidos indispensables para toda la Argentina.

En el marco de esta política pública, el Consejo Federal de Educación aprobó, en 2018, los Núcleos de Aprendizajes Prioritarios (NAP) de Educación Digital, Programación y Robótica (EDPR) para toda la educación obligatoria, es decir, desde la sala de 4 años hasta el fin de la secundaria. Abarcan un campo de saberes interconectados y articulados, orientados a promover el desarrollo de competencias y capacidades necesarias para que los estudiantes puedan integrarse plenamente en la cultura digital, tanto en la socialización, en la continuidad de los estudios y el ejercicio de la ciudadanía, como en el mundo del trabajo.

La incorporación de Aprender Conectados en la Educación Secundaria permite poner a disposición estudiantes y docentes, tecnología y contenidos digitales que generan nuevas oportunidades para reconocer y construir la realidad: abre una ventana al mundo, facilita la comunicación y la iniciación a la producción digital.

La sociedad está cambiando a un ritmo más acelerado que nuestro sistema educativo y la brecha entre las propuestas pedagógicas que presentan las escuelas y la vida de los estudiantes se amplía cada vez más. Garantizar el derecho a aprender en el siglo XXI implica que todos los estudiantes puedan desarrollar las capacidades necesarias para actuar, desenvolverse y participar como ciudadanos en esta sociedad cada vez más compleja, con plena autonomía y libertad.

En este marco, Aprender Conectados presenta actividades, proyectos y una amplia variedad de recursos educativos para orientar la alfabetización digital en la educación obligatoria en todo el país. La actividad que se presenta a continuación y el resto de los recursos del Plan son un punto de partida sobre el cual cada docente podrá construir propuestas y desafíos que inviten a los estudiantes a disfrutar y construir la aventura del aprender.

María Florencia Ripani
Directora Nacional de Innovación Educativa

Objetivos generales

Núcleos de Aprendizajes Prioritarios

Educación Digital, Programación y Robótica – Educación secundaria	Ofrecer situaciones de aprendizaje que promuevan en los alumnas y alumnos: • La comprensión general del funcionamiento de los componentes de hardware y software, y la forma en que se comunican entre ellos y con otros sistemas, entendiendo los principios básicos de la digitalización de la información y su aplicación en la vida cotidiana. • El desarrollo de proyectos creativos que involucren la selección y la utilización de múltiples aplicaciones, en una variedad de dispositivos, para alcanzar desafíos propuestos, que incluyan la recopilación y el análisis de información. • La creación, la reutilización, la reelaboración y la edición de contenidos digitales en diferentes formatos, entendiendo las características y los modos de representación de lo digital. • La resolución de problemas a partir de su descomposición en partes pequeñas, aplicando diferentes estrategias, utilizando entornos de programación tanto textuales como icónicos, con distintos propósitos, incluyendo el control, la automatización y la simulación de sistemas físicos.

Objetivos de aprendizaje

Esta actividad permitirá introducir al lenguaje de programación Python y está orientada a desarrollar conocimientos iniciales vinculados con los siguientes objetivos de aprendizaje:

- Aprender a generar aleatoriedad en el código.
- Profundizar los conceptos de funciones y estructura de datos.
- Profundizar los conceptos de estructuras de selección y bucles.
- Profundizar en el manejo de gráficos.
- Realizar un programa que cargue sus datos desde una base de datos.

Materiales y recursos

- Computadora
- Python 2.x instalado

Desafío

¿Recuerdan los juegos de cartas para chicos donde se realizan comparaciones de atributos como fuerza o velocidad para ver quién gana? El desafío de esta actividad es crear un juego de cartas digital utilizando los personajes de Digiaventuras.

< Inicio >

Disparador

"¿Recuerdan los juegos de cartas para chicos donde se realizan comparaciones de atributos como fuerza o velocidad para ver quién gana? ¿Les gustaría crear uno para que lo jueguen los chicos de primaria?"

En esta actividad vamos a crear un juego de cartas donde cada carta será un personaje de la serie de educación digital Digiaventuras. Cada personaje tendrá una serie de atributos con un valor entre 0 y 100. El juego comienza con cada jugador eligiendo una carta al azar de su mazo. A continuación, cada jugador elige un atributo para su carta. Finalmente se comparan los valores de cada atributo entre las cartas de los jugadores. Cuando algún jugador gana en ambos atributos, gana la partida. Para que la partida sea justa, los participantes deberían jugar con la misma lista de cartas.

Las líneas de código son presentadas dentro de este documento con el siguiente formato para su fácil identificación, y copiado.

```
# Soy un comentario en el código
print 'Soy una línea de código'
Soy una línea de código
```

Las líneas de color azul son la respuesta al código introducido en las líneas anteriores y se presentan como un ejemplo del resultado a obtener. No deben ser copiadas ni ejecutadas en IDLE.

También se incluyen comentarios en gris, precedidos por el símbolo numeral. Estos comentarios son notas que dan claridad al código, pero que Python ignora y no son ejecutados.

El ejercicio se propone para ser escrito línea por línea, en IDLE. Es importante que todos los estudiantes estén frente a una computadora con IDLE, de Python, abierto al momento de comenzar la actividad y que, a medida que avanzan, ejecuten el programa para comprobar que su código esté bien y corregir errores, si los hubiera.

Al final de este documento se presentan todas las líneas del programa juntas, que pueden ser guardadas en un archivo desde IDLE, para ser ejecutado.

< Desarrollo >

La creación del programa se puede dividir en los siguientes pasos:

1. Creación de base de datos de cartas.
2. Importación de librerías externas.
3. Configuración de la ventana.
4. Carga de datos externos.
5. Presentación de instrucciones e interacción con el usuario.
6. Impresión en pantalla de las cartas.

El docente recuerda a los alumnos acerca de la importancia de utilizar los comentarios e invita colocar los códigos para que Python reconozca todos los acentos y signos en los sistemas operativos Linux y Windows. También se sugiere ingresar los datos del autor y el nombre del programa.

```
# -*- coding: cp1252 -*-
# -*- coding: 850 -*-
# -*- coding: utf-8 -*-
# Creador: Pedro Perez
# Arte Digital
```

Recordá utilizar comentarios de forma frecuente en el que expliques tu código. ¡El programador que lo lea te lo agradecerá!

1) Creación de base de datos de cartas.

Para agilizar la actualización de los atributos de las cartas existentes, para agregar nuevas cartas de forma simple y eficiente, e incluso para compartir nuestro mazo con otros jugadores, es importante que los atributos de las cartas de nuestro mazo estén definidos de forma externa al programa. Los lugares donde almacenamos datos para un programa de forma externa, los denominamos bases de datos. Una base de datos puede ser desde un sistema computacional complejo, hasta un simple archivo de texto. Para la resolución de esta actividad, utilizaremos un archivo de texto donde cada línea representa una carta, o una entrada en nuestra base de datos, y cada línea tiene separados sus atributos por una coma. A este tipo de archivos se los llama CSV por sus siglas en inglés (comma separated values) y pueden ser abiertos como simple texto o como hoja de cálculo.

Definimos los atributos de las cartas como:

NOMBRE: El nombre de nuestro Digiamigo.

FUERZA: Un valor entre 0 y 100 que representa la fuerza de nuestro digiamigo.

INTELIGENCIA: Un valor entre 0 y 100 que representa la inteligencia de nuestro digiamigo.

DESTREZA: Un valor entre 0 y 100 que representa la destreza de nuestro digiamigo.

VELOCIDAD: Un valor entre 0 y 100 que representa la velocidad de nuestro digiamigo.

IMAGEN: El nombre de un archivo de imagen GIF que represente a nuestro digiamigo.

Para las imágenes de las cartas del ejercicio ejemplo utilizamos el siguiente banco de imágenes de Digiaventuras:
https://drive.google.com/drive/folders/1GyvilbdbPEhmjfnoQBlBWELqrh9Q6dnU

Los estudiantes pueden crear sus propios dibujos o fotos, o utilizar imágenes y fotos de licencia libre obtenidos en internet. Lo importante es que el archivo de imagen sea extension .GIF. Si la extensión del archivo de imagen fuera otra, se puede convertir a GIF con un software de edición de imágenes o utilizando servicios de conversión de archivos en línea.

A continuación presentamos la base de datos de cartas que utilizaremos para el ejemplo de esta actividad.

cartas.csv

```
cactus, 40, 34, 18, 49, cactus.gif
elio, 43, 28, 31, 12, elio.gif
luna, 26, 54, 54, 20, luna.gif
mundo, 32, 65, 63, 25, mundo.gif
supertablet, 34, 64, 18, 56,supertablet.gif
ojo, 40, 12, 20, 79, ojo.gif
robot, 42, 73, 29, 37, robot.gif
rociobot, 19, 88, 15, 47, rociobot.gif
```

¿Te animás a agregar otros robots nuevos y personalizados?

2) Importación de librerías externas.

Una vez más vamos a trabajar con gráficos, por lo que importamos la librería Turtle, y en función de que buscamos obtener cartas al azar, vamos a importar la función “choice()” de la librería “random”. Esta librería nos provee de funciones para obtener números y selecciones al azar. “choice()” nos devuelve un elemento al azar de una lista ordenada de elementos.

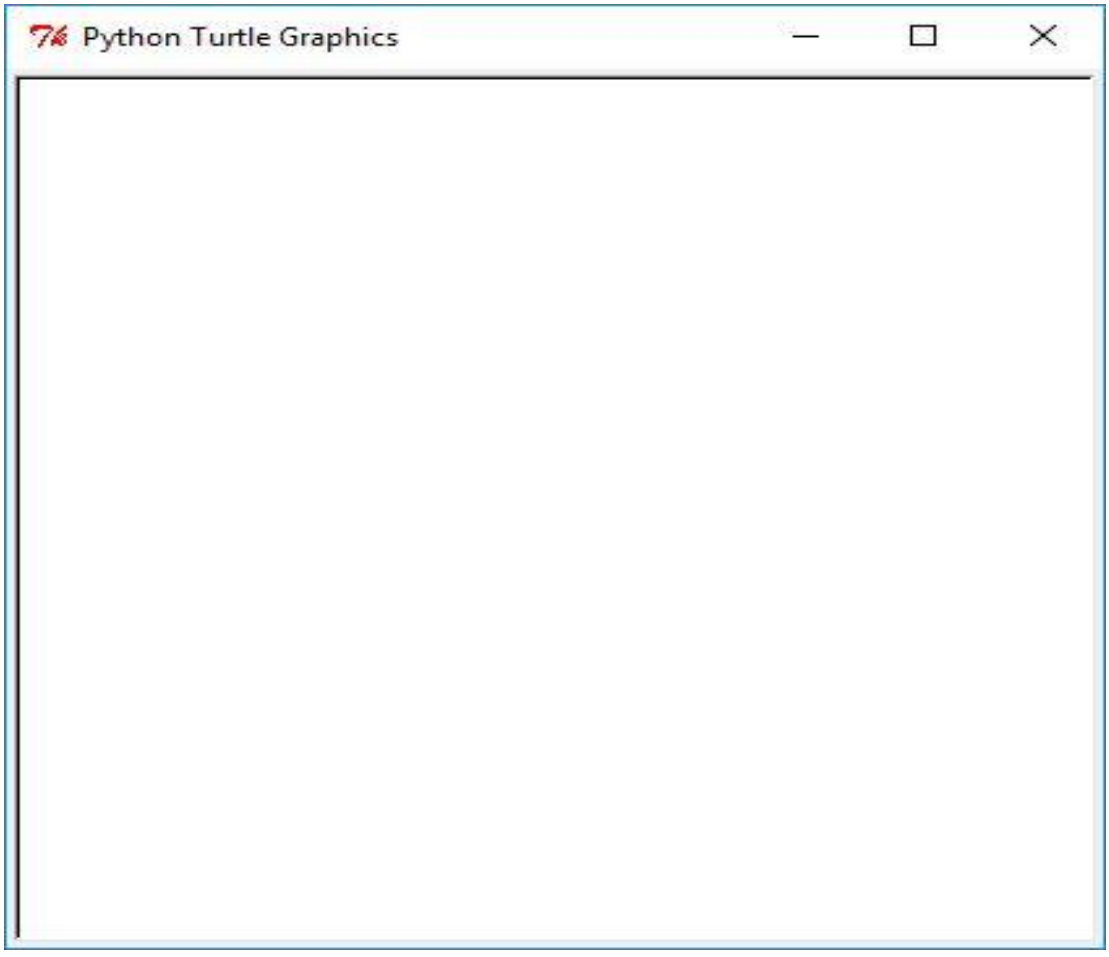
```
from turtle import *
from random import choice
```

¿Se te ocurre cómo optimizar aún más este programa? ¿Se podrían importar solo las funciones de Turtle que usamos en vez de la librería completa?

3) Configuración de la ventana.

Comenzaremos por crear y configurar la ventana donde mostraremos las cartas.

```
# Configuramos la pantalla
screen = Screen() # Inicializamos nuestra ventana
screen.bgcolor('white') # Definimos el color de fondo como blanco.
penup() # Levantamos el lápiz para no dibujar la pantalla.
hideturtle() # Escondemos el puntero para que no se vea.
```



Resultado al ejecutar el código de ejemplo

4) Carga de datos externos.

Guardaremos los datos que recuperemos del archivo separado por comas en un diccionario de nombre “digiamigos”. Comenzamos por inicializar el diccionario y abrir el archivo con la función “open()”.

```
# Inicializamos el diccionario "digiamigos" y abrimos el archivo desde
donde cargaremos las cartas.
digiamigos = {}
archivo = open('cartas.csv', 'r')
```

Para leer el archivo línea a línea, utilizaremos la función “splitlines()” que nos devuelve el contenido del archivo en una lista de líneas. Esa lista la iteramos con un *loop for* para obtener los datos de las cartas de a una por vez.

Dentro del *loop for* vamos a separar los atributos en variables, utilizando la función de texto “split()” que nos permite dividir un texto en variables.

Por último, guardamos los contenidos de estas variables como una entrada en nuestro diccionario “digiamigos”.

```
# Cada carta es una línea en el archivo.
# Cada atributo está separado por comas.
# Por cada línea en el archivo, guardamos el nombre
# y los atributos de la carta dentro de nuestro diccionario.
for linea in archivo.read().splitlines():
  # El parámetro ', ' le indica a la función split() que
  # el separador de atributos es la coma.
  nombre, fuerza, inteligencia, destreza, velocidad, imagen =
linea.split(', ')
  # Guardamos los atributos recuperados
  # como una entrada en el diccionario.
  digiamigos[nombre] = [fuerza, inteligencia, destreza, velocidad,
imagen]
  # Registramos la imagen en nuestra pantalla para poder utilizarla
  screen.register_shape(imagen)

# Ya una vez fuera del loop cerramos el archivo de texto.
archivo.close()
```

¿Como harías para agregar otros atributos a las cartas? ¿Este es el único archivo que tendrías que modificar?

Si imprimimos el contenido del diccionario digiamigos, veremos que todas nuestras cartas fueron cargadas:

```
>>> print digiamigos
{'rociobot': ['19', '88', '15', '47', 'imagenes\\rociobot.gif'],
'elio': ['43', '28', '31', '12', 'imagenes\\elio.gif'],
'supertablet': ['34', '64', '18', '56', 'imagenes\\supertablet.gif'],
'mundo': ['32', '65', '63', '25', 'imagenes\\mundo.gif'],
'robot': ['42', '73', '29', '37', 'imagenes\\robot.gif'],
'luna': ['26', '54', '54', '20', 'imagenes\\luna.gif'],
'ojo': ['40', '12', '20', '79', 'imagenes\\ojo.gif'],
'cactus': ['40', '34', '18', '49', 'imagenes\\cactus.gif']}
```

¿Pueden describir cómo está compuesta la estructura de datos propuesta para las cartas?

5) Presentación de instrucciones e interacción con el usuario.

Necesitamos que nuestro programa continúe ejecutándose de forma continua, ya que cada partida puede consistir de varias rondas hasta que se defina el ganador. Una manera de realizar acciones de forma continua es con un *loop* infinito. Un *loop* infinito nunca deja de ejecutarse porque siempre se cumple su condición. Un ejemplo es el *loop while* con el parámetro "*True*" que significa verdadero en inglés.

Una vez dentro del *loop*, ejecutamos las acciones del programa. Comenzamos por imprimir las instrucciones de uso a la vez que recibimos una instrucción por teclado. Esta doble acción se puede realizar de forma simple con raw_input.

```
# Imprimimos las instrucciones de uso y recibimos la instrucción por
teclado.
while True:
  avatar = raw_input("\n Presiona ENTER para obtener\
 tu carta al azar. \n\
 Escribe el nombre de cualquier Digiamigo para ver sus atributos.\n")
```

¿Sabés qué son y qué hacen todos esos símbolos "\"?

Llegó el momento de evaluar el dato ingresado por el usuario. Nos encontramos con tres posibles escenarios.

- Presiono Enter para que el programa seleccione una carta al azar de su baraja.
- Escribió el nombre de alguno de los personajes de la baraja para ver sus atributos.
- Escribió una cadena de caracteres diferente al nombre de un personaje en la baraja.

Consideremos la primera opción. Si el usuario presiona Enter, la variable "avatar" quedará vacía. Evaluamos esta condición con "if", y si cumple, seleccionamos un avatar al azar.

```
# Si presionan enter, elegimos una carta al azar.
  if avatar == "":
    # Aquí utilizamos la función choice para aleatorizar la elección
    # de la carta.
    avatar = choice(list(digiamigos.keys()))
    # Mostramos por pantalla el nombre del
    # digiamigo seleccionado al azar.
    print(avatar)
```

6) Impresión en pantalla de las cartas.

En este punto del programa, la variable "avatar" o tiene el nombre de un digiamigo como contenido (ya sea seleccionado al azar o ingresado por el usuario), o tiene una cadena de caracteres diferentes. Podemos evaluar fácilmente esta condición utilizando el condicional "if" junto con la palabra clave "in". "*if*" compara la cadena de caracteres ingresada por el usuario con la "*key*" o "llave" de cada elemento del diccionario "digiamigos". Si coincide, se ejecuta el bloque de código dentro del bucle. Si la condición es cierta, significa que la cadena de caracteres ingresados por el usuario coinciden con el nombre de un digiamigo, y por lo tanto podemos mostrar la carta en pantalla. Para esto guardamos los atributos de la carta en la variable "atributos", definimos una tipografía y la guardamos dentro de la variable "tipografia" y utilizamos estos datos junto a las herramientas que nos da la librería *turtle* para escribir y dibujar en pantalla la imagen y los atributos de nuestra carta.

```
# Imprimimos en pantalla la imagen y los atributos de la carta
seleccionada.
  if avatar in digiamigos:
    #
    atributos = digiamigos[avatar]
    tipografia = ('Courier', 14, 'bold')
    clear()
    goto(0, 130)
    shape(atributos[4])
    setheading(90)
    stamp()
    setheading(-90)
```

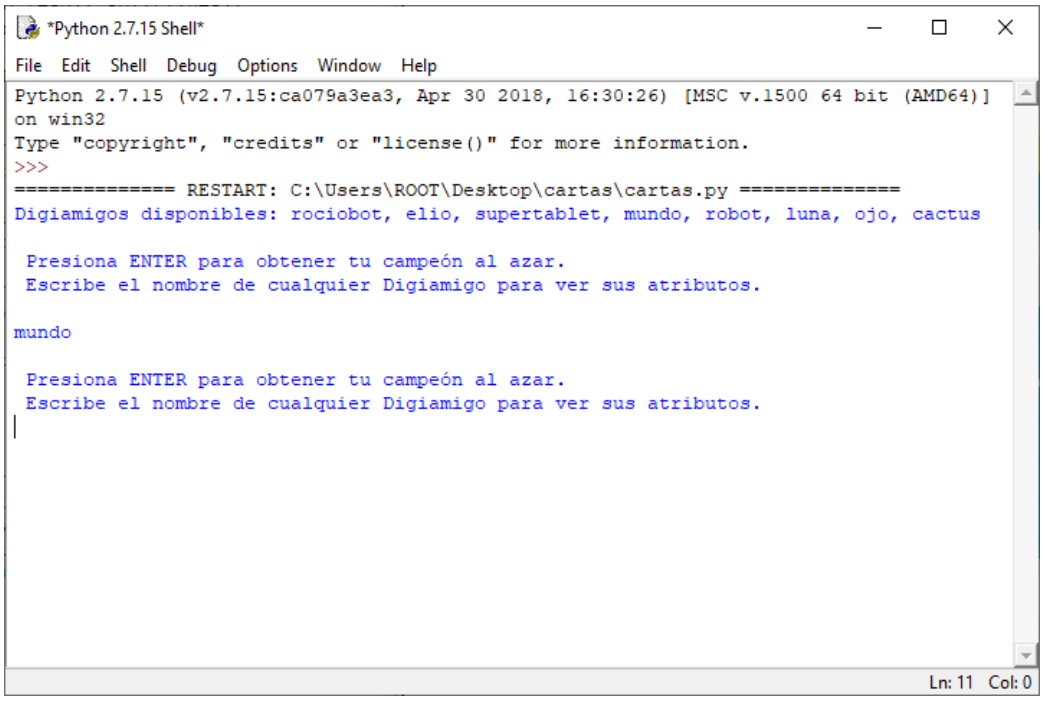
```
 forward(280)
    write('Nombre: ' + avatar, font=tipografia, align='center')
    forward(25)
    write('Fuerza: ' + atributos[0], font=tipografia,
align='center')
    forward(25)
    write('Inteligencia: ' + atributos[1], font=tipografia,
align='center')
    forward(25)
    write('Destreza: ' + atributos[2], font=tipografia,
align='center')
    forward(25)
    write('Velocidad: ' + atributos[3], font=tipografia,
align='center')
```

¿Podés explicar línea a línea qué acciones está realizando el código con la librería turtle?

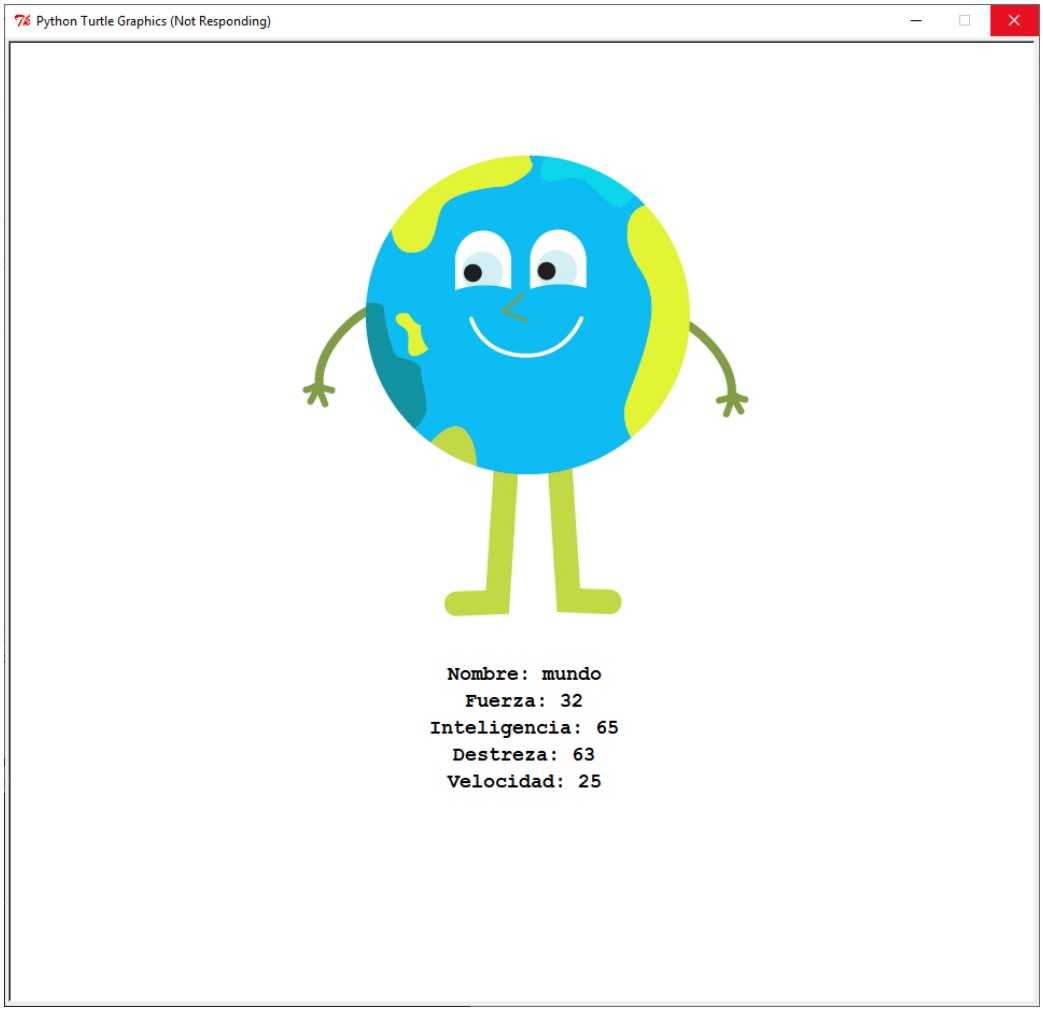
Finalmente, utilizamos la palabra clave "*else*" que, cuando es utilizada a continuación de un bloque de condición "*if*", nos permite ejecutar un código para todos los casos en que NO se cumpla la condición del "*if*". De esta manera alertamos al usuario que ingresó mal el dato por teclado.

```
# Si la carta seleccionada no existe, informamos al usuario del error.
  else:
    print("El nombre ingresado no corresponde a una carta del mazo.")
```

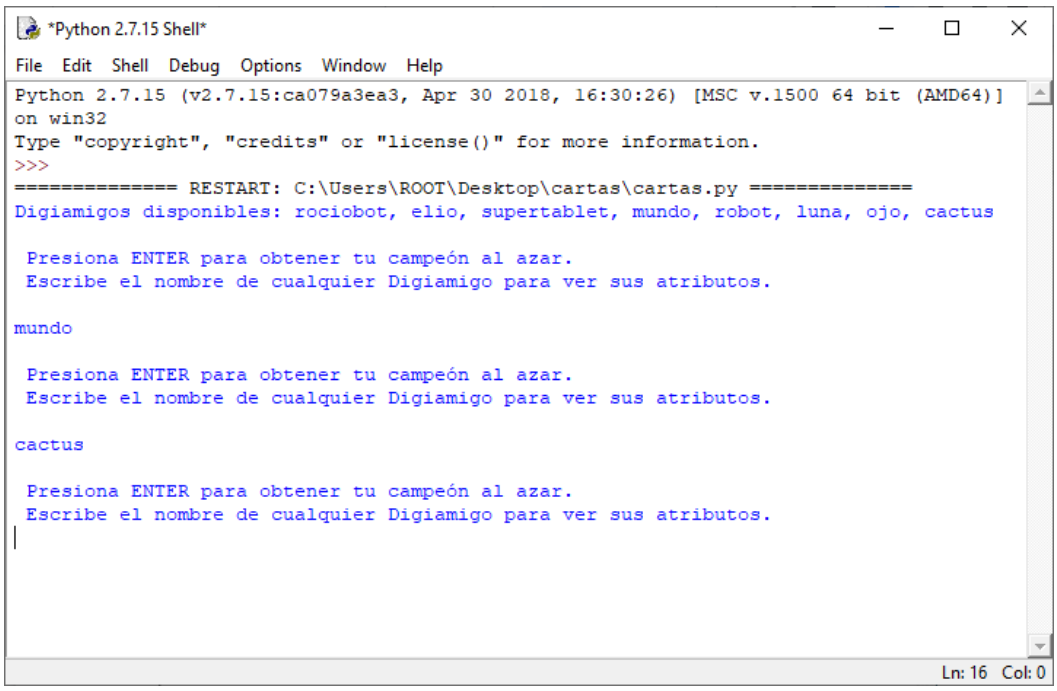
Como todo el código principal del programa está dentro del *loop* "*while*" infinito, cuando termina esta última condición, todo se vuelve a ejecutar, volviendo a presentar al usuario con la opción de presionar *Enter* o elegir un personaje de la lista de digiamigos.



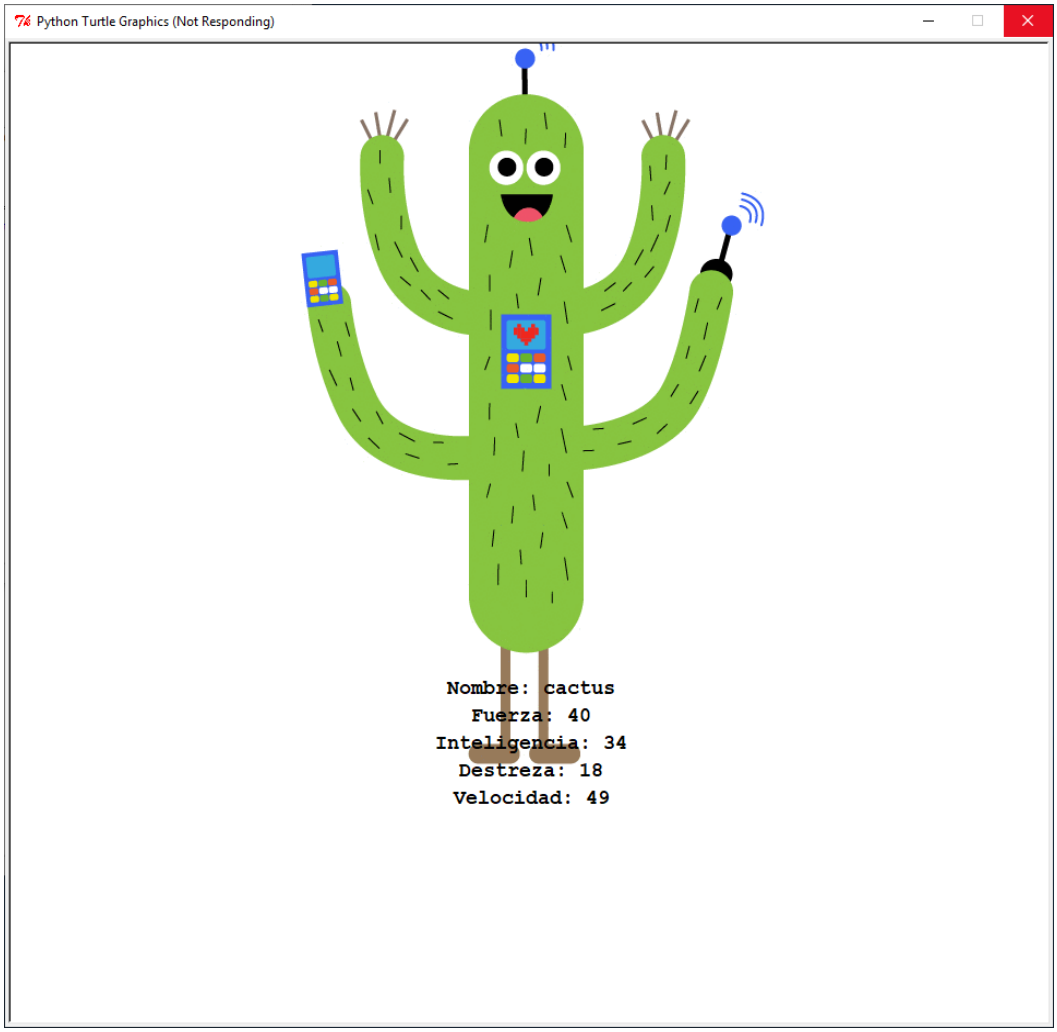
Ejemplo de ejecución del programa.



Ejemplo de ejecución del programa.



Ejemplo de ejecución del programa.



Código completo

A continuación presentamos el código completo para ser copiado en un nuevo archivo de IDLE. Podés guardarlo con el nombre cartas.py

```
# -*- coding: cp1252 -*-
# -*- coding: 850 -*-
# -*- coding: utf-8 -*-
# Creador: Pedro Perez
# Cartas Digiaventuras

#Importamos las librerías externas
from turtle import *
from random import choice

# Configuramos la pantalla
screen = Screen() # Inicializamos nuestra ventana
screen.bgcolor('white') # Definimos el color de fondo como
blanco.
penup() # Levantamos el lápiz para no dibujar la pantalla.
hideturtle() # Escondemos el puntero para que no se vea.


# Inicializamos el diccionario "digiamigos" y abrimos el archivo
desde donde cargaremos las cartas.
digiamigos = {}
archivo = open('cartas.csv', 'r')

# Cada carta es una línea en el archivo.
# Cada atributo está separado por comas.
# Por cada línea en el archivo, guardamos el nombre
# y los atributos de la carta dentro de nuestro diccionario.
for linea in archivo.read().splitlines():
  nombre, fuerza, inteligencia, destreza, velocidad, imagen =
linea.split(', ')
  digiamigos[nombre] = [fuerza, inteligencia, destreza,
velocidad, imagen]
  screen.register_shape(imagen)
archivo.close()

# Imprimimos en pantalla la lista de cartas disponibles.
print('Digiamigos disponibles: ' + ', '.join(digiamigos.keys()))
```

```
# Imprimimos las instrucciones de uso y
# recibimos la instrucción por teclado.
while True:
  avatar = raw_input("\n Presiona ENTER para obtener tu
campeón al azar. \n\
 Escribe el nombre de cualquier Digiamigo para ver sus
atributos.\n")

# Si presionan enter, elegimos una carta al azar.
  if avatar == "":
    avatar = choice(list(digiamigos.keys()))
    print(avatar)

# Imprimimos en pantalla la imagen y
# los atributos de la carta seleccionada.
  if avatar in digiamigos:
    atributos = digiamigos[avatar]
    tipografia = ('Courier', 14, 'bold')
    clear()
    goto(0, 130)
    shape(atributos[4])
    setheading(90)
    stamp()
    setheading(-90)
    forward(280)
    write('Nombre: ' + avatar, font=tipografia,
align='center')
    forward(25)
    write('Fuerza: ' + atributos[0], font=tipografia,
align='center')
    forward(25)
    write('Inteligencia: ' + atributos[1], font=tipografia,
align='center')
    forward(25)
    write('Destreza: ' + atributos[2], font=tipografia,
align='center')
    forward(25)
    write('Velocidad: ' + atributos[3], font=tipografia,
align='center')

# Si la carta seleccionada no existe, informamos al usuario
del error.
  else:
    print("El nombre ingresado no corresponde a una carta del
mazo.")
```

Después de guardar el archivo, podés ejecutarlo desde IDLE, presionando la tecla “F5”.

< Cierre >

El docente pide a los estudiantes que se agrupen para probar sus proyectos. Si ven que algo del programa no funciona, trabajan juntos para encontrar el error y solucionarlo.
Luego, se sugiere hacer una puesta en común para intercambiar sus impresiones sobre lo que aprendieron, lo que más les gustó hacer y sobre los desafíos que se les presentaron.

Evaluación:
El docente puede evaluar el proyecto tanto a través de la observación, durante el desarrollo de las actividades, como en relación al programa final. Puede copiar los proyectos y evaluarlos en detalle, pero también puede utilizar la instancia de puesta en común.

Se tendrán en cuenta los objetivos específicos de la actividad, como otros aspectos vinculados a la creatividad, la cooperación entre pares y el aprendizaje a partir de la exploración y el error.

A continuación, se presentan preguntas orientadoras:

- ¿Logró resolver los desafíos propuestos?
- ¿Pudo crear la base de datos?
- ¿Trabajó en grupo cuando se encontró con problemas?
- ¿Escribió el programa propuesto de forma lógica y ordenada, solucionando los errores de sintaxis?
- ¿Creó alguna carta o atributo nuevo?
- ¿Comprendió las lógicas de iteración propuestas en la actividad?

El proceso de evaluación de la evolución del aprendizaje podrá continuar con la actividad propuesta a continuación.

Para seguir aprendiendo

Se sugiere extender el software para que se pueda jugar de a dos jugadores en una misma computadora, realizando cada jugador una selección al azar de un mazo y luego el programa realizando la comparación de atributos de forma automática.

Finalmente sugerimos al docente plantear una actividad en grupo, en la que, con la ayuda de los estudiantes, se piense en un programa que resuelva una problemática en particular y que se valga de las herramientas aprendidas en este ejercicio (creación de funciones, creación de base de datos externa, carga de imágenes personalizadas, dibujo de formas en pantalla, manipulación del puntero, utilización de bucles, etc.).

El proyecto será completamente libre, para que puedan aplicar los aprendizajes construidos.