VIDEOJUEGOS
INTRODUCCIÓN A PYTHON

CLASE 5

Contenido

“

Este espacio curricular permite fortalecer el espíritu crítico y la actitud creativa del futuro graduado. Lo alienta a integrar los conocimientos previos, recreando la práctica en el campo real, al desarrollar un proyecto propio.

”

PRORAMACIÓN I

Uno de los primeros avances que se tiene en programación es cuando comenzamos a trabajar con funciones, el cambio de nuestro código es enorme. Hace un uso correcto de las funciones, nos permite desarrollar códigos mas legibles y mas mantenibles.

En este apartado, vamos a extender nuestro conocimiento en el manejo de funciones. Incorporamos el uso de módulos y librerías que son muy útiles para asegurar la mantenibilidad del código. Por otro lado, trataremos ligeramente el sistema de Python para gestionar los errores del código en tiempo de ejecución.

Introducción

Terminando ya con esta primera aproximación de Python y la programación estructurada, hablaremos sobre los módulos y como estos pueden convertirse en nuestras librerías de trabajo. Cuando desarrollamos, estos es muy útil puesto que nos permite crear nuestras primeras herramientas para futuros trabajos. También, vamos a debatir sobre la importancia de controlar los datos de entrada de nuestros juegos. Por último, un tema de regalo. Veremos como persistir la información que nos resulte útil en el juego en archivos de texto.

Librerías

Las librerías de Python son extensiones que podemos agregar en nuestros proyectos. Es verdad que todo puede ser resuelto con nuestro trabajo y esfuerzo, pero a veces es mejor usar las librerías que otro creo para acelerar nuestro desarrollo. Y en la mayoría de las veces, porque lo hicieron mejor de lo que uno podría 😊.

Para profundizar tu conocimiento en el lenguaje Python, no esta de mas visitar algunas de estas librerías.

¿Qué son las librerías?

Son archivos que contienen un conjunto de funciones que otro escribió (esto es cuestionable) que están disponibles para usarlas en nuestros proyectos.

Las librerías de Python son referencias a un conjunto de funciones que te ayudarán a mejorar y/o completar tu videojuego. En Python, las librerías cuentan con un conjunto de módulos que permiten el acceso parcial a funciones especificas

NumPy

Es una librería indispensable cuando necesitas trabajar con operaciones matemáticas, de algebra y matrices

Pandas

Muy utilizada para trabajar con bases de datos, facilita el análisis de los datos y es muy fácil de usar.

Scikit-learn

Una librería interesante para trabajar con machine learning, no se cuanto aporta a los videojuegos, pero no está de más saber que existe.

PyGame

En este caso, indispensable si tu intención es crear videojuegos sencillos. Una librería que te va a llevar al siguiente nivel.

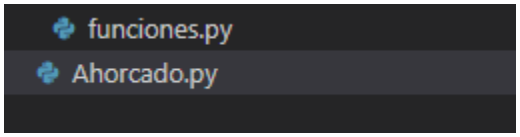
Módulos

Para facilitar el mantenimiento y la legibilidad de nuestros programas en Python, existen los módulos. Nuestro código fuente se puede subdividir, y en cada división podemos colocar un subconjunto de nuestras funciones. La división es arbitraria, pero lo lógico es colocar en un mismo módulo las funciones que tienen una relación concreta entre sí. Esto claro, para que buscarlas luego sea mucho más sencillo.

Siguiendo lo anteriormente dicho, cada archivo físico en Python es un módulo. Estos pueden encontrarse en el directorio raíz de nuestro proyecto o dentro de carpetas creadas jerárquicamente.

Creando módulos

Simplemente creamos un archivo.py y dentro de ellos colocamos nuestras funciones



El siguiente paso, es importar el modulo dentro de nuestro archivo principal.

```
import funciones
```

La limitación, si pudiera llamarse así, de usar un **import** es que tenemos que utilizar el namespace (en este caso nombre del archivo) para referir a cada función dentro del módulo.

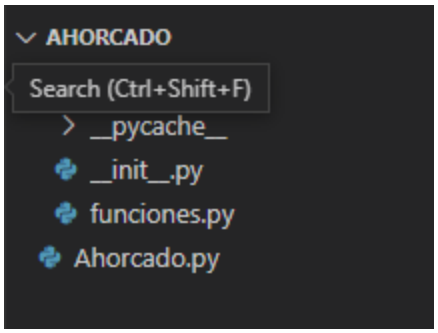
```
funciones.limpiarPantalla()
```

Si deseamos evitar esa aclaración, podemos utilizar la construcción from-import *

```
#imortando librerías
from Lib.funciones import *

#Llamando funciones
limpiarPantalla()
```

Continuando con el uso de módulos, podemos para mayor legibilidad colocar los archivos.py dentro de una carpeta. La declaración es muy simple, usamos los nombres de las carpetas como parte del namesapce y de esa forma podemos importar los módulos.



Para que esto ultimo funcione, solo debemos tener una precaución. El primer archivo de la carpeta debe ser un archivo vacío llamado __init__.py. Cuando usamos el __ini__.py a esta carpeta la llamamos paquete.

La importación del modulo se vería como el código siguiente

```
import Lib.funciones
```

también podemos Usar el from

```
from Lib.funciones import *
```

Control de errores

El control de errores es sumamente importante para que nuestro videojuego sea robusto. Debemos prestar principal atención a dos cosas, los datos de entrada y posibles errores en el código al momento de ejecutar nuestro código fuente.

Los errores en Python se llaman excepciones (Exception) Para controlar errores no esperados en el código usamos la función Try.

Exception

Son los errores personalizados que tiene Python, el lenguaje tiene una enorme lista de enumeradores para estos errores.

Python proporciona una lista de excepciones estándar como:

Exception: es la base de todas las excepciones

ArithmeticError: clase base para todos los errores que se producen en un cálculo aritmético

FloatingPointError: se genera cuando falla un cálculo de punto flotante

ZeroDivisionError: Se produce cuando se hace una división por cero

EOFError: error cuando se alcanza el final del archivo.

ImportError: cuando falla la declaración del import.

NameError: se genera cuando no se encuentra el nombre el namespace local o globa,

IOError: cuando falla la operación de entrada y salida,

ValueError: cuando tiene el tipo de valor de argumento incorrecto.

Para más información te recomiendo que visites https://docs.python.org/3/library/exceptions.html

Try- Except:

En Python se utiliza un operador Try para controlar las excepciones en un bloque de código. Cuando se produce un error, lo atrapa el **except**. En este último, se describe el tratamiento que se le dará al error.

```
try:
    a=10/0
except:
    print("No se puede dividir por cero")
```

podemos personalizar el **except** para que el motor de excepciones trabaje menos, indicando que error nos interesa atrapar. La estrategia es ir siempre de lo particular a lo general

```
try:
    a=10/0
except ZeroDivisionError:
     print("No se puede dividir por cero")
except:
     print("algo salio mal")
```

podemos declarar mas de una excepción en el **except** separando las excepciones por coma. además, tenemos la cláusula **else** que se ejecuta cuando no ha ocurrido ningún error en el bloque try.

```
try:
    a=10/0
except ZeroDivisionError:
    print("No se puede ")
except:
    print("algo salió mal")
else:
    print("todo está bien")
```

y, por último, tenemos la cláusula **finally**. Se utiliza para ejecutar un código al terminar el try se haya encontrado un error o no.

```
try:
    a=10/0
except ZeroDivisionError:
    print("No se puede ")
except:
    print("algo salió mal")
else:
    print("todo está bien")
finally:
    print("siempre")
```

Archivos

Unas de las funcionalidades de Python es manipular archivos de texto (.txt). Esta tarea es muy sencilla, y podemos realizarla con las instrucciones print o write.

La instrucción open, nos permite crear o actualizar archivos de texto. Para ello debemos indicar el modo. Las opciones son las siguientes:

r: abre el archivo en modo lectura

r+: abre en modo lectura y escritura, pero no se puede modificar

w: abre el archivo en modo escritura, se reemplaza el archivo original si existe

a: abre en modo escritura, pero permite agregar y se conserva el contenido original

a+: modo escritura solo permite agregar contenido al final.

En conclusión, si solo deseas agregar contenido al final sin borrar usas a+, y si solo necesitas leer el contenido puedes usar r.

Creando y agregando contenido a un archivo

Utilizamos la instrucción open, respetando la siguiente sintaxis:

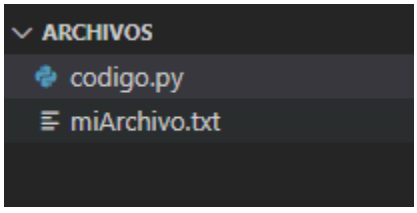
[nombre del archivo]=open("[nombre.txt]","[modo]")

Copiando contenido en el archivo con print

Posterior a crear la variable que representa al archivo, podemos usar la instrucción print para agregar contenido al documento.

```
f=open("miArchivo.txt","w")
print("escribiendo", file=f)
f.close()
```

es muy importante cerrar el archivo con close.



Copiando contenido en el archivo con write

```
f=open("miArchivo.txt","w")
f.write("contenido")
f.close()
```

lectura del contenido del archivo

podemos recorrer cada línea del archivo como si fuese una lista

```
f=open("miArchivo.txt","r")
for i in f:
    print(i)
f.close()
```

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS D:\salaPython\archivos> & C:/Python310/python.exe d:/salaPython/archivos/codigo.py
1

2

3

4

PS D:\salaPython\archivos>
```

Bibliografía

Sweigart, A. (2012). Making Games with Python & Pygame: A Guide to Programming with Graphics, Animation, and Sound. Recuperado de https://inventwithpython.com/makinggames.pdf

Pygame. (s.f.). [Página oficial de pygame]. Recuperado 2 julio, 2019, de https://www.pygame.org/news