Exploración de Python en contextos no científicos

Luca Sarramone[1], Virginia Cifuentes[1], Guillermo H. Rodriguez[1]

[1] Universidad Nacional del Centro (UNICEN), Tandil, Bs.As., Argentina

Resumen Python es un lenguaje de programación que permite la creación de programas robustos en forma rápida y sencilla gracias a su simpleza, en combinación con una comunidad activa que constantemente desarrolla nuevas bibliotecas y frameworks. Este trabajo presenta distintas aplicaciones del lenguaje Python en ámbitos específicos al desarrollo web y la manipulación de imágenes. También detalla las bibliotecas y los frameworks utilizados (por ejemplo, PyGame, Pillow y Django, entre otras). El objetivo es demostrar la versatilidad y facilidad del lenguaje para el desarrollo de software más allá del contexto académico. El enfoque del trabajo es más bien didáctico, permite visualizar la amplia utilización de Python, su fuerte presencia en los cursos universitarios de programación, y su gran potencial para crear aplicaciones robustas en pocas líneas de código.

Keywords: Python, Lenguajes de Programación, Enseñanza de Programación, Desarrollo de Software, Aplicaciones Interactivas

1 Introducción

Aprender a programar por primera vez puede ser un desafío, no solo por el lenguaje de programación en sí mismo o los procesos de pensamiento requeridos para describir un problema para que una computadora pueda entender, sino también por las herramientas que los estudiantes deben usar para desarrollo [1]. La mayoría de las veces, las herramientas adecuadas para principiantes no existen, obligando a los instructores a usar programación de nivel profesional entornos que intimidan a los principiantes y requieren de gran esfuerzo para instalar y configurar correctamente.

En este contexto, estamos frente a una posible contradicción entre la necesidad de la sociedad moderna de contar con desarrolladores calificados que puedan programar, y la reticencia de los estudiantes a aprender programación, lo cual es demostrado por resultados académicos en los primeros años de universidad y también deserción de los mismos. La base de cualquier educación en informática es la capacidad de analizar y desarrollar códigos de programa y todo lo que respalda esta actividad. Dada esta contradicción, las universidades se enfrentan a la necesidad de redefinir sus planes de estudio, prestando especial atención al desarrollo de cursos de programación, cuyo conocimiento debe aplicarse aún más en las actividades profesionales de futuros desarrolladores [2].

La mayoría de las universidades han comenzado a elegir el lenguaje de programación Python, que además es un lenguaje muy activo en la comunidad de código fuente abierto. En 2018, Python ganó el premio “Lenguaje de programación del año” [3], que se otorga a los lenguajes de programación que tienen el mayor crecimiento de calificación por año. Los programas escritos en Python parecen considerablemente más cortos que los programas equivalentes escritos en otros lenguajes de programación populares, como Java, C, C ++, Visual Basic y .NET, debido a los tipos de datos intrínsecos de alto nivel y el tipado dinámico [6].

Siguiendo esta línea, este trabajo de cátedra propone demostrar la facilidad de desarrollar aplicaciones con Python mostrando un abanico de aplicaciones interactivas desarrolladas con Python. El objetivo es evidenciar la versatilidad y facilidad del lenguaje para el desarrollo de software más allá del contexto académico, donde generalmente este lenguaje es usado. Como segundo objetivo es destacar que

Python es más que un lenguaje de prototipado sino que, por el contrario, permite crear aplicaciones robustas y en diferentes dominios de aplicación.

El resto del trabajo se organiza de la siguiente manera: la Sección 2 describe las principales características del lenguaje Python. La Sección 3 presenta y muestra las aplicaciones interactivas desarrolladas. Finalmente, la Sección 4 concluye el trabajo e identifica líneas para trabajos futuros.

2 Python

Python es un lenguaje de programación de alto nivel interpretado y multipropósito creado por Guido Van Rossim. Durante los últimos años se ha convertido en uno de los lenguajes de programación más utilizados para el desarrollo de software, pudiéndose usar en varios sistemas operativos y plataformas, como Windows, Mac OS X, Linux, teléfonos inteligentes y sistemas integrados [4]. Su simpleza lo hace especialmente sencillo de aprender y permite crear grandes programas con unas pocas líneas de código. Todo esto posible gracias a una comunidad muy activa que desarrolla bibliotecas y frameworks de manera constante, facilitando el trabajo a los programadores.

Aunque C / C ++ y Java han sido opciones populares para la enseñanza de programación en las universidades, Python es el lenguaje de programación más popular para la introducción cursos de programación en las mejores universidades de EE. UU [5].

3 Enfoque para desarrollar aplicaciones con Python

El desarrollo de aplicaciones es una de las áreas más abarcativas de la programación, ya que el número de dispositivos donde se puede desarrollar es cada vez mayor. Con el fin de mostrar el gran potencial del lenguaje se tomaron como ejemplo las siguientes 10 aplicaciones.

3.1 PyGame

PyGame[1] es una librería gratuita y de código abierto que permite crear aplicaciones (principalmente videojuegos) construidas sobre Simple DirectMedia Layer (SDL), por lo cual son altamente portables y pueden correr prácticamente en cualquier plataforma o sistema operativo. Para este caso se programó como caso de ejemplo una versión reducida del clásico juego de arcade *Arkanoid*. El código consta de tres partes bien definidas: Objetos, Funciones Auxiliares y Main.

[1] https://www.pygame.org/docs/

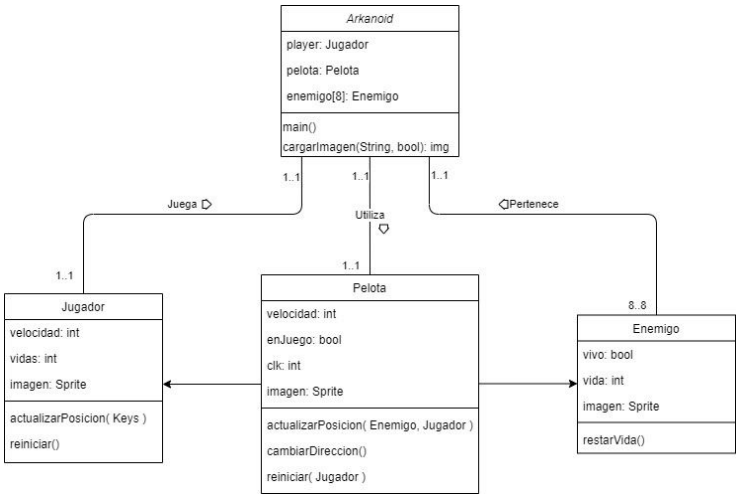

Fig. 1. Diagrama de clases para Arkanoid

Los primeros son los “ladrillos" para la construcción de cualquier aplicación en Pygame ya que permiten describir el comportamiento de cada objeto dentro del videojuego. Para el ejemplo desarrollado se utilizaron tres clases diferentes, cuya relación se puede apreciar en la Fig.1. En primer lugar se encuentra la pelota, que cuenta con una velocidad para cada eje, una posición relativa a la pantalla, un *sprite*[2] y una velocidad de actualización. Cuando la cantidad de ciclos iguala a esta última variable se mueve el objeto una cantidad de pixeles relativa a la velocidad del mismo. En caso de colisionar entonces la dirección se invierte y continua avanzando. Por otro lado se encuentra el jugador que cuenta con las mismas variables que la clase anterior, agregando además una cantidad de vidas. El mismo sólo puede moverse en el eje x mientras alguna de las flechas de dirección esté siendo presionada. Al llegar a los bordes de la pantalla no puede avanzar más y se queda en la última posición válida. Finalmente se encuentra la clase enemigo, que define un *sprite*, una posición fija en pantalla y una cantidad de vida. Cada vez que es golpeado se resta uno a esta última variable y se intercambia el color de su dibujo. La Fig.2 muestra el juego desarrollado en acción.

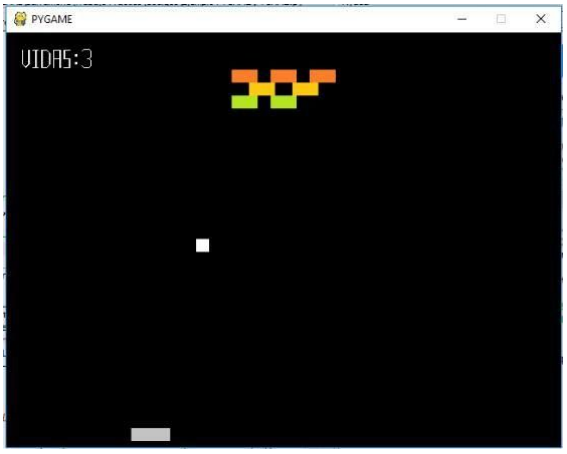

Fig. 2. Juego realizado con PyGame (Arkanoid)

Dentro de la parte de funciones, solo se programó la carga de imágenes que se usarán dentro del juego, para así centralizar el tratamiento de errores. Por último,

[2] Sprite es un término usualmente utilizado en videojuegos para referirse al dibujo o figura que representa a un objeto en la pantalla del jugador.

dentro del Main se encuentra la definición de cada objeto dentro del juego (una pelota, un jugador y 9 enemigos - en la figura se ven 8 porque 1 ha sido destruido por la pelota) junto a un bucle infinito donde se agrupa todo lo relacionado al comportamiento del juego. En cada ciclo reloj se verifican los tiempos de actualización de la pelota y el jugador, además de dibujar los *sprites* en pantalla según la posición indicada por los objetos correspondientes. En caso de que el juego finalice porque se perdieron todas las vidas (en la figura se aprecian 3 vidas) se pasa a la pantalla de *game over*.

3.2 Kivy

Es una biblioteca gratuita y de código abierto que permite desarrollar aplicaciones para distintas plataformas. En este caso se programó una calculadora simple utilizando los objetos predefinidos por la biblioteca.

Existen dos partes bien distinguidas en Kivy: una clase principal donde se retorna el objeto que representa la aplicación, y un conjunto de clases que define los *widgets* que utilizará.

Dada la simpleza del código, solo se utilizó un *widget*[3] que abarca todo el comportamiento de la calculadora, la cual cuenta con un visor (una etiqueta que muestra el resultado de las distintas operaciones) y un teclado (por donde se ingresan los datos a usar). Para poder establecer el orden de los distintos objetos se usó la super clase "gridlayout" a la cual se le asigna un número de columna y organiza los *widgets* como si se tratara de una grilla. Dado que la cantidad de columnas necesarias para los botones difiere de las usadas por el visor se debe crear una subgrilla, de manera tal que el resultado sea vea tal como en la Fig. 3.

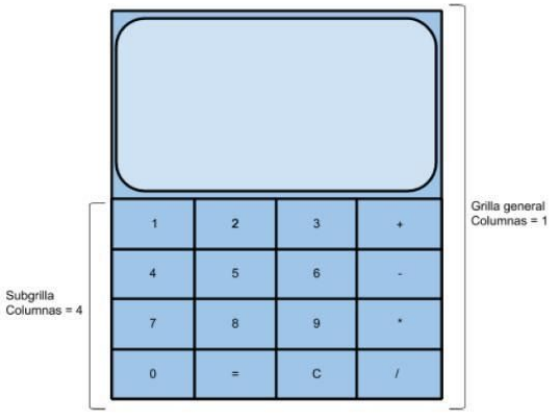

Fig. 3. Pantalla de la calculadora desarrollada con Kivy.

Una vez decidida la organización, se crea la etiqueta que funciona como visor, y las 16 teclas, las cuales se asocian al funcionamiento correspondiente. Utilizando la función "eval" ofrecida por Python, es posible evaluar un string como si fuera una ecuación, por lo que cada tecla sólo debe concatenar el valor que representa a la cadena de caracteres, que se irá representando en el visor a medida que se construye.

Como ya se mencionó, el proyecto se realizó utilizando los objetos predefinidos por la biblioteca, pero en caso de que el trabajo fuera más grande es evidente que codificar de esta manera se vuelve ineficiente. Es por esto que Kivy tiene un lenguaje especial, llamado "KV language", que permite crear el árbol de *widgets* de forma declarativa para luego asociarle su funcionalidad, facilitando así la separación entre la interfaz de usuario y la lógica de la aplicación.

[3] Un widget en una interfaz gráfica es un elemento de interacción, como un botón o una barra de desplazamiento.

3.3 Desarrollo web

El desarrollo web refiere a la creación de sitios o aplicaciones web, donde se combinan conocimiento de HTTP y bases de datos. Python es un lenguaje muy usado en esta área ya que su simpleza permite crear páginas web en pocas líneas de código, lo que permite desarrollar y debuggear los sitios de manera rápida y fácil. Para este caso se tomaron como ejemplo 2 frameworks: *Dash* y *Django*.

3.3.1 Dash[4]

Desarrollado por Plotly y construido sobre Flask[5], Plotly.js[6], and React.js[7], *Dash* es un framework que permite crear aplicaciones web de manera sencilla, desplegarlas en servidores y compartirlas mediante URLs. Para este caso se programó un sencillo ejemplo de aplicación, donde montando un servidor local se puede ver desde navegador un gráfico de barras y una pequeña calculadora.

La estructura de los códigos de *Dash* es sencilla. Primero, se define el *layout* de la aplicación utilizando HTML para los componentes más sencillos y la biblioteca "dash_core_components" para los más complejos (por ejemplo, gráficos). Luego, se define la funcionalidad utilizando *callbacks* mediante la instrucción "@app.callbacks", donde se establece cuáles serán los componentes de entrada y salida involucrados. Seguido, se define la función que ejecuta, teniendo en cuenta que los parámetros y retornos de la misma se corresponden con los *inputs* y *outputs* del *callback* respectivamente.[8]

Se puede desplegar la aplicación en un server local (127.0.0.1:8050) ejecutando el comando:

python DASH.py

3.3.2 Django[9]

Django es un framework de desarrollo web de código abierto, escrito en Python, que respeta el patrón de diseño conocido como Modelo–Vista–Template[10] (MVT). Tiene como objetivo principal ayudar a los diseñadores a crear aplicaciones rápidamente. Dada la complejidad de la biblioteca, el programa de muestra se redujo a crear una pequeña aplicación web donde se pueden realizar encuestas y votarlas. Todos los proyectos de *Django* cuentan con varios archivos *.py*, por lo cual se describe brevemente qué hace cada uno de ellos.

La Fig. 4 ilustra las estructura de archivos y directorios de la aplicación creada en *Django* siguiendo el tutorial[11] de la página. Dentro del directorio raíz se pueden encontrar una carpeta con el nombre del sitio que contiene archivos relacionados a la configuración general del proyecto (zona horaria, idioma, tipo de base, etc.) y otra llamada "polls", que contiene la aplicación de votaciones. El funcionamiento es sencillo: cuando la aplicación recibe una url, recorre los archivos de este último directorio en forma secuencial hasta hacer match con alguna de las entradas

[4] https://plotly.com/dash/
[5] https://flask.palletsprojects.com/en/1.1.x/
[6] https://plotly.com/javascript/
[7] https://es.reactjs.org
[8] https://dash.plotly.com/introduction
[9] https://www.djangoproject.com
[10] https://docs.hektorprofe.net/django/web-personal/patron-mvt-modelo-vista-template/
[11] https://docs.djangoproject.com/en/3.1/intro/tutorial01/

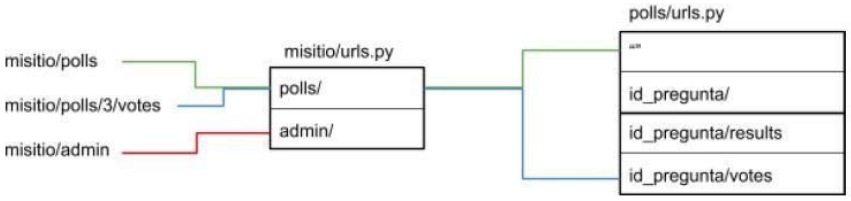

Fig. 4. Estructura de archivos con Django.

Al encontrar la referencia se ejecuta la vista asociada. Todas las vistas están almacenadas en el archivo "views.py" como funciones diferentes que definen el contenido y comportamiento de la página indicada. Para esto, utiliza la carpeta *templates* y los modelos. La primera agrupa un conjunto de archivos HTML que definen el esquema general de la vista. De esta manera, si se necesita modificar cómo se ve la página basta solo con cambiar los HTML y no el código. Por otra parte, los modelos (definidos dentro del archivo "models.py") son los objetos que utilizará la base de datos de la aplicación. Si no se cuenta con una base propia, como es el caso, *Django* viene por defecto equipado con la posibilidad de crearlas en SQLlite[12]. Para modificarla es necesario usar la API de la biblioteca, que se accede mediante el comando:

python manage.py shell

Finalmente, si se desea ver el resultado final es necesario estar situado con la consola dentro de la carpeta raíz del programa y ejecutar el siguiente comando:

python manage.py runserver

3.4 Manipulación de imágenes

Existen gran cantidad de bibliotecas que pueden ser mencionadas en esta categoría, incluso Numpy[13] a pesar de estar orientado a proveer uso para vectores y matrices. Para representar el alcance del lenguaje se decidió tomar los dos extremos: por un lado está *Pillow*, una biblioteca sencilla únicamente para imágenes, y por otro, está *OpenCV*, una biblioteca de gran potencial usada tanto en video como imágenes. Otras bibliotecas interesantes son: Mahotas, SimpleCV (una versión más sencilla de aprender que OpenCV), Scikit -image y SciPy.

3.4.1 Pillow[14]

Es una extensión de la biblioteca PIL pero utilizada en Python 3. Provee soporte para gran cantidad de extensiones y permite procesar imágenes de distintas maneras de forma eficiente, tales como extraer histogramas, aplicar filtros u otras transformaciones.

El código implementado recorre varias funcionalidades guardando los resultados de cada operación. En primer lugar, se carga una imagen y se la muestra por pantalla, lo cual resulta muy útil para comprobar las modificaciones que se aplican sobre la imagen sin la necesidad de guardar los cambios parciales. El mismo archivo es utilizado en las pruebas subsecuentes, primero mostrando sus características y transformando su extensión. Después se le aplican una serie de cambios y se guardan los resultados: se cortan sus dos mitades y se pegan en forma inversa, se rota 180 grados y finalmente se cambia su paleta de colores de RGB a escala de grises. Por

[12] https://www.sqlite.org/index.html
[13] https://numpy.org
[14] https://pillow.readthedocs.io/en/stable/

último se toma otra imagen y se le aplican distintos filtros en cada sector, mostrando en una única imagen los efectos de cada uno.

3.4.2 OpenCV[15]

Es una potente biblioteca de código libre que incluye gran cantidad de algoritmos para el tratamiento de imágenes y video. A pesar de tener una curva de aprendizaje pronunciada, toda su funcionalidad se encuentra completamente explicada en la documentación[16] de la página oficial, por lo cual no es necesario recurrir a terceros para entenderla.

Al igual que el caso anterior se aplicaron sucesivos cambios a diferentes imágenes mostrando las distintas funcionalidades que tiene *OpenCV*. Se comienza con cuestiones sencillas: ajustes de tamaño, suma y fusión de imágenes. En esto es interesante ver los distintos resultados (Fig. 5) ya que, a pesar de que la biblioteca tiene una fuerte base en Numpy, difieren increíblemente en la forma de realizar las operaciones. Por ejemplo, la suma en Numpy es modular mientras que en OpenCV la suma es saturada, como consecuencia el primero tomará el módulo del resultado en base 256, mientras que el otro tomará el máximo valor posible:

```
x = np.uint8([250])
y = np.uint8([10])
cv.add(x,y)      # 250+10 = 260 => 255 - Suma OpenCV
x+y          # 250+10 = 260 % 256 = 4 - Suma Numpy
```

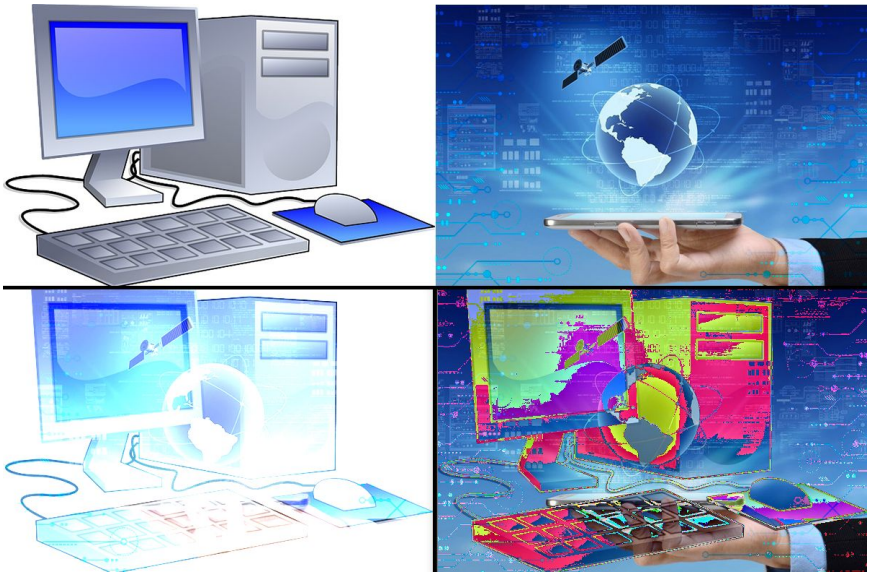

Fig. 5. Resultados de la suma en OpenCV(abajo-izquierda) y Numpy (abajo-derecha)

Luego se utilizan cambios de paleta para poder reconocer sectores de imágenes con colores particulares. Para esto primero se transforma la imagen a formato HSV[17] (*Hue, Saturation, Value*) y se obtienen las zonas donde se encuentran los colores comprendidos en el rango dado por parámetro. El resultado de esta última operación se utiliza para crear una máscara que se aplica sobre la imagen original y así obtener los sectores deseados.

[15] https://opencv.org
[16] https://docs.opencv.org/master/d6/d00/tutorial_py_root.html
[17] HSV es un modelo de representación de colores similar a RGB pero que en lugar de utilizar los colores primarios, usa los componentes de matiz, saturación y brillo.

Finalmente se creó un sencillo detector de movimiento para mostrar las aplicación de la biblioteca en formatos de video. El mismo funciona tomando una imagen limpia como referencia y comparándola con cada *frame*[18] por posibles cambios. Para mayor precisión, las imágenes se transforman a escala de grises y se aplica un filtro gaussiano. Luego se traza el contorno de los objetos en movimiento, aquellos que son muy pequeños se rechazan y se dibujan por pantalla aquellos que superen el límite de validez.

Existen muchas más funcionalidades que no se ven representadas en el código, especialmente los algoritmos relacionados a la detección de "features", machine learning y fotografía computacional. Esto se debe a que muchos de ellos están patentados, por lo cual su uso está restringido en las últimas versiones de *OpenCV*.

3.5 Otros dominios

En este trabajo, también se han desarrollado otras aplicaciones, pero solo mencionaremos brevemente de qué se tratan. En primer lugar, *RE*[19] es un módulo integrado dentro de la instalación de Python que se utiliza para procesar expresiones regulares de forma parecida a Perl[20]. Su funcionamiento es sencillo, utiliza principalmente 2 funciones: match (patrón, cadena) y search (patrón, cadena). El primero, solo busca en el inicio de la cadena y el segundo en toda el string.

En segundo lugar, *Arrow*[21] es una librería que provee funcionalidades para crear, manipular y convertir fechas, tiempos y marcas temporales. A pesar de que Python ya incluye manejo de fechas, existen una gran variedad de tipos y bibliotecas, lo cual muchas veces resulta engorroso. *Arrow* centraliza el funcionamiento en un único módulo, simplificando el uso de las fechas. El programa desarrollado implementa las funcionalidades más interesantes de la biblioteca: extracción de fechas desde un string, extracción de fechas desde un texto y la transformación de las fechas para que sean comprensibles por los humanos (por ejemplo, la resta entre las 14:00 hs y las 12:00 hs devuelve como resultado la frase "quedan 2 horas para las 14:00" en lugar de simplemente 2:00 hs). Además, se muestra las funcionalidades básicas, tales como la definición o desplazamientos de fechas.

En tercer lugar, *TQDM*[22] es una pequeña biblioteca que sirve para mostrar barras de progreso de forma fácil. El código muestra un sencillo ejemplo de su funcionamiento. Lo único que se debe hacer es encerrar la estructura iterable de un *for* con la función "tqdm", de la siguiente manera:

```
for i in tqdm(range(10000)):
    pass;
```

En cuarto lugar, *Request*[23] es una biblioteca simple de Python, utilizada para realizar peticiones HTTP directamente desde código y analizar las respuestas. El programa desarrollado muestra un resumen de la funcionalidad básica de la herramienta y el potencial que tiene. La mayoría de los ejemplos utiliza la página "httpbin.org", un sitio creado por los mismos autores que permite probar distintas configuraciones y métodos. Lo primero que se realiza en el código es una petición y se verifica su estado de 2 maneras distintas: (i) verificando el código de estado, y (ii) con el atributo "ok" del objeto respuesta, que devuelve un booleano verdadero en caso de obtener una respuesta positiva, es decir, con código de estado 400.

[18] Los frames (o fotogramas) son conjuntos de imágenes que al mostrarlas en serie a una velocidad determinada producen la ilusión de movimiento.
[19] https://docs.python.org/3/library/re.html
[20] https://perldoc.perl.org
[21] https://arrow.readthedocs.io/en/stable/
[22] https://tqdm.github.io
[23] https://requests.readthedocs.io/es/latest/

3.6 Análisis de las aplicaciones desarrolladas

La Tabla 1 resume las principales características de las aplicaciones desarrolladas: el identificador (App ID), el dominio de aplicación, y la cantidad de líneas de código (LOC) utilizadas para desarrollar cada aplicación.

Tabla 1. Resumen de aplicaciones interactivas desarrolladas con python.

App ID	Biblioteca utilizada en la aplicación	Dominio de aplicación	Líneas de Código (LOC)
1	PyGame	Desarrollo de videojuegos	197
2	Kyvy	Desarrollo de aplicaciones móviles	176
3	Arrow	Manipulación de fechas	50
4	Dash	Desarrollo Web	110
5	Dyango	Desarrollo Web	127
6	OpenCV	Visión artificial	116
7	Pillow	Procesamiento de imágenes	65
8	RE	Procesamiento de expresiones regulares	76
9	Request	Peticiones HTTP desde código	89
10	TQDM	Mostrar barra de progreso	5
	TOTAL		**1011**

Como se puede apreciar, el número de líneas para cada aplicación es considerablemente bajo, en comparación a lo que demandarían otros lenguajes de programación [6]. En total, para las 10 aplicaciones fueron necesarias 1011 líneas, que en promedio serían aproximadamente 100 líneas de código por aplicación ($\bar{x} = 101,1$; Me=99,5). Claramente, es un número muy bajo si hablamos de crear una aplicación de software, demostrando así la potencialidad del lenguaje Python para el desarrollo de aplicaciones.

4 Conclusiones

Python es un lenguaje sencillo en su estructura y fácil de usar. Permite construir soluciones robustas para resolver grandes problemas, utilizando unas pocas líneas de código. Las bibliotecas desarrolladas son complejas pero, gracias a la gran cantidad de documentación existente, es sencillo comprenderlas de forma autónoma.

La principal dificultad que se presentó durante este trabajo fue la curva de aprendizaje de algunas de las bibliotecas incluidas en el estudio, ya que muchas estaban orientadas a usos complejos que se escapan al alcance de la cátedra. Por otra parte, al coexistir dos versiones de Python (2.x y 3.x), en conjunto con una gran variedad de entornos, se dificulta la instalación de ciertos módulos, más aún si estos encima dependen de otros.

Finalmente, podemos concluir que la programación moderna se está volviendo más accesible para los usuarios comunes. Gracias a la gran cantidad de bibliotecas disponibles, y a la documentación que hay sobre ellas, el conocimiento necesario para desarrollar un programa es cada vez menor. Con el uso de Python, un científico de datos puede montar su propia página web con *Dash* y presentar los resultados de su investigación, sólo comprendiendo las nociones básicas de HTML y siguiendo los tutoriales provistos por la página de la biblioteca, mientras que un iniciado en programación puede crear un detector de movimiento simplemente usando los algoritmos de OpenCV o un aficionado a los videojuegos crear su primer proyecto con PyGame y comenzar a comprender los procesos que utilizan otros motores

gráficos más complejos (por ejemplo Unity[24]) para funcionar. En cuanto a la educación formal, Python está siendo cada vez más utilizado en cursos universitarios para enseñar programación [7][8].

Como trabajo futuro, se planea desarrollar una herramienta interactiva educativa que permita abstraer las bibliotecas Python subyacentes y que permita enseñar a programar a estudiantes, aplanando la curva de aprendizaje.

Referencias

1. Edwards, S. H., Tilden, D. S., & Allevato, A. (2014, March). Pythy: improving the intro-ductory python programming experience. In Proceedings of the 45th ACM technical symposium on Computer science education (pp. 641-646).

2. Prokopyev, M. S., Vlasova, E. Z., Tretyakova, T. V., Sorochinsky, M. A., & Solovyeva, R. A. (2020). Development of a Programming Course for Students of a Teacher Training Higher Education Institution Using the Programming Language Python. Propósitos y Representaciones, 8.

3. TIOBE Index for November (2019). Retrieved from: https://www.tiobe.com/tiobe-index/.

4. López, A. F. J., Pelayo, M. C. P., & Forero, Á. R. (2016). Teaching image processing in engineering using python. IEEE Revista Iberoamericana de Tecnologías del Aprendizaje, 11(3), 129-136.

5. Wainer, J., & Xavier, E. C. (2018). A controlled experiment on Python vs C for an Intro-ductory Programming course: students outcomes. ACM Transactions on Computing Education (TOCE), 18(3), 1-16.

6. Lutz, P. (2000). An empirical comparison of C, C++, Java, Perl, Python, Rexx, and Tcl for a search/string-processing program. Retrieved from: https://www.researchgate.net/publication/36451181_An_empirical_comparison_of_C_C_Java_Perl_Python_Rexx_and_Tcl_for_a_searchstring-processing_program.

7. Grandell, L., Peltomäki, M, Back, J.B. & Salakoski, T. (2006). Why Complicate Things? Introducing Programming in High School Using Python. Retrieved from: https://www.researchgate.net/publication/31596786_Why_Complicate_Things_Introducing_Programming_in_High_School_Using_Python.

8. Mészárosová, E. (2015). Is Python an Appropriate Programming Language for Teaching Programming in Secondary Schools?. *ICTE Journal, 4*(2), 5-14. doi: https://doi.org/10.1515/ijicte-2015-0005

[24] https://unity.com/es