e-ISSN 2444-8729
https://doi.org/10.14201/eks2017182147162

Python como primer lenguaje de programación textual en la Enseñanza Secundaria

Python as First Textual Programming Language in Secondary Education

José Carlos García Monsálvez

Profesor de Enseñanza Secundaria - Especialidad de Informática. Conselleria de Educación, Investigación, Cultura y Deporte - Generalitat Valenciana, España. garcia_josmon@gva.es

Resumen

Con la reciente introducción de la Programación en el currículo preuniversitario, se abre una oportunidad para incluir conceptos fundamentales de las Ciencias de la Computación. Este artículo presenta el origen y evolución de Python, las principales características que lo configuran como un lenguaje idóneo, así como una revisión y clasificación de herramientas educativas disponibles en su ecosistema. Dichas herramientas abarcan un amplio y variado abanico de recursos; desde libros interactivos hasta librerías que facilitan la creación de elaborados artefactos *software* por parte de los alumnos. En este trabajo se presenta una propuesta multidisciplinar para utilizar el lenguaje de programación Python en todos los niveles de Secundaria.

Palabras Clave

Python; Enseñanza Secundaria; Currículum; Computación; Informática; Pensamiento computacional; Introducción a la programación

Abstract

With the recent introduction of Programming in the K-12 curricula there is an opportunity to include Computer Science fundamental concepts. This paper presents the origin and evolution of Python as well as their main features that configure it as an ideal programming language. We also review and classify some educational tools in the Python ecosystem. Such tools cover a wide-open spectrum of resources from interactive books to libraries which ease the construction of student elaborated software artefacts. This work presents a multidisciplinary proposal to use the Python programming language in all levels of Secondary Stage.

Keywords

Python; K-12 levels; Curricula; Computing; Informatics; Computational Thinking; Introductory programming

Recepción: 25-04-2017 Revisión: 10-05-2017 Aceptación: 25-05-2017 Publicación: 30-06-2017

1. Introducción

1.1. Situación actual y previsible futuro

Debido al aumento de la importancia de las Ciencias de la Computación en la sociedad actual, tanto en la vida cotidiana como en el ámbito laboral (con un peso muy determinante en el I+D+i), muchos países han decidido incorporarlas en las etapas obligatorias de su currículo (Hubwieser, et al., 2015; García-Peñalvo, et al., 2016). Dicha incorporación varía desde la troncalidad de la asignatura de Computación, tanto en Primaria como Secundaria, en el Reino Unido o la estrategia de más largo recorrido como Israel (Hazzan, 2008; Ragonis, 2010; Gal-Ezer, 2014) hasta el escaso tratamiento que se da actualmente en España.

A pesar de la proliferación de iniciativas en nuestro entorno (Balanskat, et al., 2015), ello no se ha traducido en un avance curricular a nivel nacional (MEC) y todo queda en decisiones unilaterales de las comunidades autónomas (en base a las competencias transferidas en enseñanza). Así pues, contamos con casos puntuales en Andalucía, Comunitat Valenciana, Madrid, Cataluña y Navarra (que analizaremos con más detalle en el apartado 4).

La proliferación y avance del resto de iniciativas, tanto en el entorno de la Unión Europea como en otros países desarrollados, empujará a la lógica conclusión de introducir a nivel nacional (posiblemente en la futura ley de educación que se pretende consensuar) una nueva asignatura. Está por determinar la dedicación (asignatura troncal u optativa, dedicación semanal), el enfoque (ciencias de la computación *vs.* pensamiento computacional *vs.* programación) y modelo (basado en alguna de las experiencias de las comunidades autónomas o en casos de éxito de otros países como Israel o Estonia).

1.2. Ciencias de la computación y pensamiento computacional

Tal como se indica en *The great principles of computing* (Denning, 2003), la Computación puede ser el cuarto gran dominio científico, junto con la Física, la Biología y las Ciencias Sociales. En dicho artículo, Denning propone 7 categorías: "computation, communication, coordination, recollection, automation, evaluation and design". Así pues, las ciencias de la computación tienen entidad suficiente como para reclamar una asignatura troncal en –al menos– toda la enseñanza secundaria.

En los últimos años el *Pensamiento Computacional* se ha convertido en tema central dentro del marco de iniciativas a nivel global que pretenden llevar las Ciencias de la Computación a la enseñanza preuniversitaria (Riesco, 2014; Tedre y Denning, 2016). Hay que tener claro que Computación y

Pensamiento Computacional no son conceptos equivalentes a pesar de tener claros solapamientos. El campo de la Computación es mucho más amplio y podemos destacar como ejemplo "[that] we need to show our students what their programs are controlling before they can understand how to design programs that produce intended effects" (Tedre y Denning, 2016).

1.3. Pensamiento computacional y programación

Aunque no hay una definición académica aceptada sobre pensamiento computacional, en la extensa revisión de la literatura (García-Peñalvo, et al., 2016, cap. 2) del proyecto TACCLE3 (García-Peñalvo, 2016a) tenemos la definición más reciente del propio García-Peñalvo (2016): "the application of high level of abstraction and an algorithmic approach to solve any kind of problems".

Por otro lado, aunque algunas iniciativas pueden caer en el error de equiparar pensamiento computacional con programación, desde luego, la forma más efectiva de desarrollar dicho pensamiento computacional implica la introducción de la programación (Compañ-Rosique, et al., 2015).

No obstante, no hay resultados concluyentes sobre la transferencia del pensamiento computacional a otras áreas distintas a la programación (Guzdial, 2015a). Sin embargo, parecen evidentes las ventajas directas para niveles posteriores en la educación (Ciclos Formativos de la familia de Informática o con módulos –de otras familias profesionales– relacionados con la programación, así como primeros cursos de Universidad en carreras de Ciencia e Ingeniería –obviamente también en las carreras de Informática–) y en otros ámbitos; no solo el sector de las TIC, sino cualquier actividad que requiera un uso continuado de manipulación y procesamiento de datos (principalmente –aunque no en exclusividad– las relacionadas con Ciencia e Ingeniería).

2. Selección del lenguaje de programación

Uno de los elementos clave en la introducción a la programación es el lenguaje escogido; siendo los otros elementos: el currículo, la pedagogía y las herramientas empleadas en el proceso de enseñanza+aprendizaje (Pears, et al., 2007). La selección del lenguaje de programación se ha estudiado únicamente en 22 estudios primarios que van desde los años cincuenta hasta 2012 (Stefik, et al., 2015).

Aunque se puede seguir un proceso formal y bien estructurado para la selección del lenguaje empleado en la introducción a la programación (Parker, et al., 2006), ya existen multitud de experiencias y estudios que avalan las ventajas de Python como candidato bien posicionado.

Tanto en el ámbito universitario (Guo, 2013; Ateeq, et al., 2014; Peña, 2015a; Peña, 2015b; Koulouri,

2015; Murphy, et al., 2016) como en enseñanza secundaria (Elkner, 2000; Grandell, et al., 2006; Mészárosová, 2015) se analizan las características y aplicación de Python como lenguaje introductorio (que enunciaremos en el apartado 3.2).

La curva de aprendizaje, sintaxis simple y legibilidad de Python es reconocida en todos los casos como idónea para la introducción de la programación. Tanto es así que en ocasiones se le denomina "pseudocódigo ejecutable".

2.1. Ciencias de la computación y pensamiento computacional

Cada vez con mayor frecuencia, la primera aproximación programación para los estudiantes tanto de Secundaria como –en algunos casos– de Primaria suele basarse en un entorno de programación visual por bloques (Scratch, Alice, etc.). A pesar de facilitar la introducción de la programación en etapas más tempranas, estos lenguajes no permiten un desarrollo más allá de su entorno visual (salvo interacciones simples mediante la informática física con herramientas como S4A[1]).

Por ello, aparecen cada vez más estudios que investigan la transición de la programación visual por bloques a la programación textual (Armoni, 2015; Weintrop, 2015; Brown, 2016). Por las características que veremos más abajo, Python es escogido en varios estudios como destinatario para dicha transición a la programación textual (Dorling y White, 2015) incluso después de haber probado con la transición de programación visual a otros lenguajes como Java (Tabet, et al., 2016).

3. Python

3.1. Origen y evolución

Guido van Rossum creó Python en diciembre de 1989, basándose en el lenguaje de programación ABC que ayudó a implementar en los años 80. ABC tenía como objetivo principal la facilidad de aprendizaje para personas ajenas a las Ciencias de la Computación[2]. Con esa herencia tan clara, no debe extrañarnos que sea tan popular en los cursos introductorios de programación. Si acaso, debería extrañarnos que no lo haya sido antes.

Conocido como lenguaje de *script* durante años, no fue hasta finales de los años 90 cuando se empezó a plantear su aplicabilidad en el aprendizaje de la programación, como primer lenguaje. Python es realmente un lenguaje de propósito general y ha ido ganando popularidad en varios ámbitos como

1 Scratch for Arduino: http://s4a.cat/index_es.html
2 http://python-history.blogspot.com/

el desarrollo rápido de aplicaciones web, administración de sistemas, ciencia de datos, computación científica (donde domina con diferencia), inteligencia artificial, internet de las cosas, etc.[3].

3.2. Características idóneas

En los estudios comparativos (Grandell, 2006; Ateeq, 2014; Koulori, 2014; Mészárosová, 2015; Peña, 2015b), las principales características por las que destacan a Python son:

- Sintaxis simple.
- Alta legibilidad (sangrado obligatorio).
- Entorno amigable de desarrollo (intérprete interactivo).
- Abstracciones de más alto nivel (mayor nivel de expresividad).
- Potente librería estándar y gran cantidad de módulos de terceros (actualmente son más de 100.000).
- Multi-paradigma (imperativo, POO y funcional).
- Disponibilidad de recursos educativos abiertos.
- *Software* libre y comunidad entusiasta.

No en vano es el lenguaje más popular en los cursos introductorios de programación de las universidades en Estados Unidos (Guo, 2014), el que más resultados devuelve en una búsqueda de cursos introductorios de programación en plataformas de MOOC[4], y es el escogido por la Fundación Raspberry Pi para su programación principal (Tollervey, 2015).

3.3. Herramientas educativas

La popularidad de Python ha propiciado también la elaboración de herramientas visuales, tanto para programar directamente (Bart, 2016) como para visualizar la ejecución de programas (Guo, 2013; Tang, 2014), o incluso estudiar en entornos táctiles como tabletas y *smartphones* (Ihantola, et al., 2014).

La comunidad de Python genera una creciente cantidad de herramientas y recursos educativos abiertos. Listamos a continuación una breve clasificación[5]:

3 https://www.python.org/about/success/
4 buscando en https://www.class-central.com/
5 Un listado más exhaustivo en: https://github.com/quobit/awesome-python-in-education

- Entornos interactivos: try.jupyter.org, Trinket.io, Snakify.org, codesters.com, pythonroom.com, repl.it, etc.
- Libros interactivos: interactivepython.org (varios libros disponibles), learnpython.com, codeclub[6], etc.
- Juegos/competiciones: checkio.org, empireofcode.com, codecombat.com, codeandconquer.co, etc.
- Ejercicios: exercism.io, practicepython.org, pybit.es, etc.
- Libros (libres): Think Python, Invent With Python, Dive Into Python 3, etc.
- *Hardware*: Raspberry Pi, Arduino, BBC:microbit, PyBoard, ESP8266, etc.
- Documentación: docs.python.org/3, PyMOTW3, Python Guide, Full Stack Python, etc.
- Librerías: PyGame, Python Arcade Library, Processing (Python Mode), VPython, Pilas Engine, etc.
- Herramientas para crear cursos: Jupyter Notebook, PyCharm Edu, Runestone Interactive, etc.
- Vídeo tutoriales: Dan Bader (dbader.org), Harrison Kinsley (pythonprogramming.net), Khan Academy, The Hello World Program, etc.

4. Análisis del currículo

Como indicamos en la introducción, existen a nivel autonómico varias iniciativas que introducen la programación en el currículo. Por un lado, tenemos el caso de Andalucía, que ha creado una nueva asignatura optativa en 2º de Bachillerato denominada "Programación y Computación" (aparte de TIC II definida en la LOMCE) y que comenzó a impartirse en el curso 2016-2017 por los profesores de la especialidad de Informática disponibles en los centros de secundaria.

Por otro lado, está el caso de Navarra, que introduce por primera vez en España la programación en Primaria (en los dos últimos cursos de Matemáticas)[7]. Aunque el más publicitado fue el de la Comunidad de Madrid, con la asignatura de libre configuración autonómica denominada "Tecnología, Programación y Robótica". Esta asignatura comenzó a implantarse en el curso 2015-2016 en 1º y 3º ESO, y en el siguiente curso 2016-2017 finaliza la implantación en el resto de la ESO (2º y 4º). Esta asignatura introduce la programación entre otros temas muy variados propios de Tecnología y se imparte con profesores de dicha especialidad (y que han recibido un curso de programación).

6 https://codeclubprojects.org/en-GB/python/
7 http://codigo21.educacion.navarra.es/normativa/nuevo-curriculo/

Con diferencia, el caso del currículo de la Comunidad Valenciana es diferente al resto en varios aspectos: tiene más años de recorrido, se ha ofertado en todos los niveles de la ESO y Bachillerato desde un principio y ha ido adaptando sus contenidos con el paso de los años. Además, es impartido por profesores de la especialidad de Informática. Por todo ello, pasamos a analizar con más detalle sus características y currículo actual.

4.1. Comunidad Valenciana

La Comunidad Valenciana fue pionera en la introducción de la optativa de Informática en la Enseñanza Secundaria Pública Española, puesto que elaboró un currículum para cada curso de la ESO y Bachillerato en 1995 (Orden de 9[8] y 10[9] de mayo de 1995, respectivamente). El siguiente paso que se dio fue la obligatoriedad de la oferta de dicha optativa en todos los centros de Enseñanza Secundaria (ORDEN de 31 de mayo de 2004, de la Conselleria de Cultura, Educación y Deporte)[10] lo cual generó una demanda de profesores de la especialidad de Informática (en la actualidad, 1000 docentes).

El hecho de que la asignatura sea optativa influye en una serie de problemas:

- *Discontinuidad*: los alumnos que optan por la asignatura no tienen ninguna precondición y, por lo tanto, en cualquiera de los niveles de la ESO y Bachillerato puede haber alumnos que jamás han cursado la asignatura en años anteriores.
- *Brecha de género*: aunque no existen estadísticas al respecto en Secundaria, la optatividad facilita la tendencia general reflejada en los estudios sobre la brecha de género en las Ciencias de la Computación.
- *Nivelación*: debido a la discontinuidad antes mencionada, la disparidad de niveles difícilmente permite que los grupos alcancen de manera homogénea los indicadores de éxito para cada curso.

El último paso dado en la Comunitat Valenciana corresponde a la última actualización de currículo en 2015 (DECRETO 87/2015, de 5 de junio, del Consell)[11], donde ya se puede contar con un bloque dedicado a la programación[12] en cada uno de los cursos de la ESO y Bachillerato.

4.1.1. Comunidad Valenciana

Pasamos a analizar la parte de Programación introducida en el currículo de la ESO y Bachillerato, tal como se ha desarrollado en la Comunitat Valenciana.

8 http://www.dogv.gva.es/datos/1995/07/05/pdf/1995_835276.pdf
9 http://www.dogv.gva.es/datos/1995/06/19/pdf/1995_835093.pdf
10 http://www.dogv.gva.es/datos/2004/06/23/pdf/2004_X6485.pdf
11 http://www.dogv.gva.es/datos/2015/06/10/pdf/2015_5410.pdf
12 Con la excepción de 4º ESO, que viene determinado por el currículo a nivel nacional dictado por el Ministerio de Educación.

- 1º ESO

  "Introducción a la programación en entornos de aprendizaje. Elaboración guiada de programas sencillos a través de aplicaciones de escritorio, móviles o de portales web de aprendizaje y promoción de la programación en entornos educativos. Introducción a los conceptos de la programación por bloques: composición básica de las estructuras y encaje de bloques. Programación de gráficos, animaciones y juegos sencillos".

  El planteamiento para 1º ESO corresponde claramente a la iniciación básica de la programación mediante tutoriales interactivos en portales como Code.org o MIT Scratch. Cuando se compara con los contenidos de 2º ESO (más abajo), llama la atención que no se introduce ningún concepto como variables, estructuras de control, condiciones o descomposición de problemas. Sin embargo, se presupone que el estudiante será capaz –al final del bloque– de realizar animaciones y juegos sencillos.

- 2º ESO

  "Entornos para el aprendizaje de la programación. Familiarización con el entorno de trabajo. Objetos. Gestión de la apariencia y los sonidos asociados a los objetos. Integración de imágenes creadas o retocadas mediante *software* de tratamiento de la imagen digital. Ejecución simultánea de varios objetos. Comunicación entre objetos. Uso de eventos. Bloques de movimiento. Estructuras de control del flujo del programa. Condiciones y operadores. Bucles. Creación de gráficos combinando bucles y herramientas de dibujo. Definición y uso de variables. Descomposición de problemas de mayor complejidad en conjuntos más sencillos de bloques. Realización de proyectos sencillos y compartido en línea. Evaluación de proyectos de otros compañeros".

  El planteamiento de 2º es una clara continuación de 1º. Teniendo en cuenta que, al ser una asignatura optativa, no podemos considerar que el alumno haya cursado Informática el año anterior (esto puede ocurrir en todos los cursos). Aunque 1º se presentaba como una mera exposición al entorno de programación visual por bloques, puede ser necesario volver a introducir a los alumnos que no la cursaron. Dependiendo del tiempo dedicado al bloque de programación en 1º, podrá ser más o menos necesaria la repetición de los principales conceptos para una parte del grupo.

- 3º ESO

  "Lenguajes de programación. Concepto, funcionalidad y tipos de lenguajes. Derechos de autor en las aplicaciones. Tipo de *software*: el *software* libre y el *software* propietario. Licencias de *software*. Programación de aplicaciones para dispositivos móviles como videojuegos, de comunicación, de

captura y edición de fotos, de integración de elementos multimedia, etc. Familiarización con el entorno de trabajo. Diseño de la interfaz de usuario.

Inserción, configuración y distribución en pantalla de los componentes de la interfaz de usuario de la aplicación. Estructuras de control del flujo de la aplicación: condicional, bucles y funciones. Definición y uso de variables. Uso de componentes multimedia. Integración de imágenes, audio y vídeo propios, creados o modificados por medio del *software* de edición correspondiente. Gestión de la comunicación: llamadas, mensajes, GPS, etc. Operaciones matemáticas y de cadenas de texto. Descomposición de problemas de mayor complejidad en módulos más sencillos. Funciones. Gestión de interfaces de la aplicación. Realización de proyectos de diferentes niveles de dificultad de forma individual o cooperativamente. Ejecución de la aplicación en dispositivos móviles o en emuladores. Descarga e instalación de la aplicación en el dispositivo. La distribución de aplicaciones para dispositivos móviles.

Evaluación de proyectos de otros compañeros".

Nos encontramos el mismo caso que en 2º ESO, pero con un aumento en la complejidad del tipo de aplicaciones a elaborar y una semejanza clara con otro entorno reconocible: MIT App Inventor. En este caso se hace un mayor hincapié en la estética y el diseño de la interfaz y la interacción tanto con el usuario como con diferentes subsistemas propios de dispositivos móviles.

- 1º Bachillerato

"Representación del problema o proyecto mediante el modelado. Análisis de requisitos de una aplicación. Entrada y salida de los datos. Restricciones del programa. Diseño de diagramas sencillos de casos de uso o de diagramas de contexto. Aplicación de algoritmos y de diagramas de flujo en la resolución de problemas sencillos. Resolución de un problema dividiéndolo en subproblemas de menor complejidad que facilite la elaboración de algoritmos para su resolución, y combinando las soluciones para resolver el problema original. Resolución de un problema a través de la generalización de ejemplos particulares. Lenguajes de programación. Definición. Tipos de lenguajes de programación. Análisis del código fuente de un pequeño programa informático. Obtención de resultados a partir de unas condiciones iniciales predeterminadas y realizando las trazas de ejecución. Programación de pequeñas aplicaciones mediante un lenguaje de programación determinado: para la programación de aplicaciones de escritorio, para el desarrollo web, para el diseño de aplicaciones de dispositivos móviles o para la creación de programas de control robótico y su ejecución en plataformas de hardware. Sintaxis y semántica de un lenguaje de programación determinado. Estructura de un programa. Variables y constantes. Tipos de datos sencillos. Entrada y salida. Operadores. Estructuras de control: bifurcaciones y bucles. Funciones y procedimientos".

En este curso tenemos la opción de elegir el tipo de aplicación: escritorio, web, móvil o IoT (Internet de las Cosas). Cabe destacar que hay un bloque de contenidos de este mismo curso, se contempla el modelo relacional y la creación de bases de datos, por lo que se podría aprovechar dicha circunstancia para poder elaborar artefactos *software* más complejos. Como actualmente, la Informática no forma parte de las pruebas de acceso a la Universidad, cabe esperar una influencia negativa a la hora de ser escogida (tanto en 1º como 2º de Bachillerato). El alumno que escoja esta optativa debe estar motivado ante los retos planteados en el temario y no obtener una compensación en el actual baremo empleado en el acceso a la Universidad. A todo esto, hay que añadir la posibilidad –comentada anteriormente– de no haber cursado previamente Informática en ninguno de los cursos anteriores.

- 2º Bachillerato

  "Representación del problema o proyecto mediante el modelado. Análisis de requisitos de una aplicación. Entrada y salida de los datos. Restricciones del programa. Diseño de Diagramas de Flujos de Datos o de casos de uso, de clases y de secuencias. El paradigma de la programación orientada a objetos (POO). Objetos y clases. Aplicación de algoritmos y de diagramas de flujo en la resolución de problemas de mediana complejidad. Resolución de un problema dividiéndolo en subproblemas de menor complejidad que facilite la elaboración de algoritmos para su resolución, y combinando las soluciones para resolver el problema original. Resolución de un problema a través de la generalización de ejemplos particulares. Técnicas simples de diseño de algoritmos. Programación de aplicaciones de mediana complejidad mediante un lenguaje de programación determinado: para la programación de aplicaciones de escritorio, para el desarrollo web, para el diseño de aplicaciones de dispositivos móviles o para la creación de programas de control robótico y su ejecución en plataformas de hardware. Sintaxis y semántica de un lenguaje de programación determinado. Aplicación de los conceptos básicos de la POO. Definición de clases.

  Instanciación de objetos. Herencia. Tipos de datos estructurados. Módulos. Acceso a bases de datos. Uso de entornos de desarrollo de *software*. Análisis del código fuente de un programa informático. Obtención de resultados a partir de unas condiciones iniciales predeterminadas y realizando las trazas de ejecución. Depuración y optimización de programas".

  Siguiendo con el planteamiento de 1º de Bachillerato (con respecto a la motivación intrínseca necesaria para cursar una asignatura que no puntúa en el baremo de acceso a la Universidad) y con un aumento de la complejidad (modularización, programación orientada a objetos, etc.), el número de estudiantes que escojan esta optativa se verá reducido drásticamente.

Cuando analizamos el currículo actual de la ESO, vemos que el diseño se realizó pensando en unas

herramientas muy concretas (Scratch y App Inventor). De esta manera caemos en la tradicional enseñanza del manejo de aplicaciones (p. e. ofimática) y centrándonos más en el uso en lugar de plantear la resolución de problemas más abiertos que conduzcan al aprendizaje del pensamiento computacional.

5. Propuesta metodológica multidisciplinar

En los cursos de introducción a la programación, tradicionalmente se ha planteado al alumno una serie de retos de mayor o menor dificultad una vez explicados los constructos elementales del lenguaje de programación que debe emplear (Guzdial, 2015b). Aunque, más bien al contrario, dedicar mayor tiempo a la lectura de código antes de comenzar a programar ha resultado mucho más efectivo. La legibilidad de Python nos facilita pues esta tarea, pudiendo ser más efectivos en la lectura comparado con otros lenguajes.

En general, hay claros resultados positivos cuando se aplican algunos enfoques activos (Freeman, 2014) al proceso de enseñanza+aprendizaje en las ciencias de la computación. Algunos de ellos son: el PBL (Nuutila, 2005), la introducción de juegos (Papastergiou, 2009) o los presentados por Porter, Guzdial y otros (Porter, 2013): *media computation* (Guzdial, 2015a), *pair programming (*Braught, 2011) y *peer instruction* (Stoilescu, 2010; Zingaro, 2014).

El aprendizaje basado en proyectos (PBL) permite al alumno enfatizar en la propia actividad aprendiendo sobre los problemas, fijando sus propias metas de aprendizaje además de realizar una búsqueda activa y análisis de la información. Un ejemplo claro y motivador es el desarrollo de juegos (para el que Python tiene varias librerías y recursos educativos) que puede servir incluso de integrador de otros bloques de contenidos (como los de multimedia).

Precisamente, el empleo de elementos multimedia (audio, imágenes y vídeo) en la introducción a la programación, ha jugado un papel determinante tanto en la motivación y resultados como para reducir la brecha de género antes mencionada (Guzdial, 2015a). Existen gran cantidad de recursos (módulos, códigos de ejemplo, secciones de libros con licencia libre, etc.) que permiten la manipulación de elementos multimedia con Python.

La programación por parejas se ha confirmado como efectiva en el aprendizaje de la programación desde hace años. Curiosamente tendemos a buscar lo contrario en las aulas de Informática ("Un alumno, un ordenador" ha sido más de una vez el *slogan*). Tanto la programación por parejas como la Instrucción entre pares puede resultar más efectiva todavía por el empleo de un lenguaje como Python, que permite la interacción inmediata (por ser interpretado) y por lo tanto una barrera menor que facilita la ejecución de estas metodologías.

6. Conclusiones

Algunas características del lenguaje Python (facilidad de aprendizaje, interpretado, librería amplia y módulos de terceros[13], *software* libre, multitud de recursos educativos abiertos) le dan una versatilidad clara para adaptarse a cualquiera de los planteamientos anteriores (y aplicables a cualquier otro currículo autonómico actual). En definitiva, permite crear un ecosistema idóneo para la educación del pensamiento computacional.

Es necesario un replanteamiento de los objetivos que ha de perseguir una asignatura de Ciencias de la Computación. El profundo debate que ha existido en otros países no parece haber afectado a las autoridades educativas en España. Deberíamos ser capaces de inspirarnos en el trabajo que presentó el creador de Python: *Computer Programming for Everybody* en el que se planteó la pregunta: "What will happen if users can program their own computer?" y en los Grandes Principios de la Computación (Denning, 2003). Otra fuente de inspiración deberían ser los casos de éxito en otros países, en especial Israel por su trayectoria.

Como propuestas de investigación futura cabría proponer una metodología para la elaboración de recursos educativos abiertos (bien partiendo de cero o mediante traducciones a nuestra lengua de otros recursos de gran aceptación). Otro ámbito de interés podría ser la creación de herramientas de ayuda para el aprendizaje de la programación, como por ejemplo el análisis del código generado por el alumno. Un ejemplo de este tipo de herramienta para MIT Scratch es DrScratch[14] (casualmente programada en… Python).

7. Referencias

Armoni, M., Meerbaum-Salant, O., & Ben-Ari, M. (2015). From scratch to “real” programming. *ACM Transactions on Computing Education (TOCE), 14(*4), 25. doi: https://doi.org/10.1145/2677087

Ateeq, M., Habib, H., Umer, A., & Rehman, M. U. (2014, April). C++ or Python? Which One to Begin with: A Learner’s Perspective. In *Proceedings of the 2014 International Conference on Teaching and Learning in Computing and Engineering* (LaTiCE) (pp. 64-69). EEUU: IEEE. doi: https://doi.org/10.1109/LaTiCE.2014.20

Balanskat, A., & Engelhardt, K. (2015). *Computing our future. Computer programming and coding. Priorities, school curricula and initiatives across Europe*. European Schoolnet, Brussels.

13 El índice de paquetes de terceros (PyPI) sobrepasa las 100.000 entradas.

14 http://www.drscratch.org/

Bart, A. C., Tibau, J., Tilevich, E., Shaffer, C. A., & Kafura, D. (2016, June). Implementing an Open-access, Data Science Programming Environment for Learners. In *Proceedings of the 2016 IEEE 40th Annual Computer Software and Applications Conference (COMPSAC)* (Vol. 1, pp. 728-737). EEUU: IEEE. doi: https://doi.org/10.1109/COMPSAC.2016.132

Braught, G., Wahls, T., & Eby, L. M. (2011). The case for pair programming in the computer science classroom. *ACM Transactions on Computing Education (TOCE), 11*(1), Article 2. doi: https://doi.org/10.1145/1921607.1921609

Brown, N. C., Altadmri, A., & Kölling, M. (2016, March). Frame-Based Editing: Combining the Best of Blocks and Text Programming. In *Proceedings of the 2016 International Conference on Learning and Teaching in Computing and Engineering (LaTICE)* (pp. 47-53). EEUU: IEEE. doi: https://doi.org/10.1109/LaTiCE.2016.16

Compañ-Rosique, P., Satorre-Cuerda, R., Llorens-Largo, F., & Molina-Carmona, R. (2015). Enseñando a programar: un camino directo para desarrollar el pensamiento computacional. *Revista de Educación a Distancia, 46*. doi: https://doi.org/10.6018/red/46/11

Denning, P. J. (2003). Great principles of computing. *Communications of the ACM, 46*(11), 15-20. doi: https://doi.org/10.1145/948383.948400

Dorling, M., & White, D. (2015, February). Scratch: A way to logo and python. In *Proceedings of the 46th ACM Technical Symposium on Computer Science Education* (pp. 191-196). New York, EEUU: ACM. doi: https://doi.org/10.1145/2676723.2677256

Elkner, J. (2000, January). Using Python in a high school computer science program. In *Proceedings of the 8th International Python Conference* (pp. 2000-2001).

Freeman, S., Eddy, S. L., McDonough, M., Smith, M. K., Okoroafor, N., Jordt, H., & Wenderoth, M. P. (2014). Active learning increases student performance in science, engineering, and mathematics. *Proceedings of the National Academy of Sciences, 111*(23), 8410-8415. doi: https://doi.org/10.1073/pnas.1319030111

Gal-Ezer, J., & Stephenson, C. (2014). A tale of two countries: Successes and challenges in K-12 computer science education in Israel and the United States. *ACM Transactions on Computing Education (TOCE), 14*(2), Article 8. doi: https://doi.org/10.1145/2602483

García-Peñalvo, F. J. (2016a). A brief introduction to TACCLE 3 – Coding European Project. In F. J. García-Peñalvo & J. A. Mendes (Eds.), *2016 International Symposium on Computers in Education (SIIE 16)*. USA: IEEE. doi: https://doi.org/10.1109/SIIE.2016.7751876

García-Peñalvo, F. J. (2016b). Proyecto TACCLE3 – Coding. In F. J. García-Peñalvo & J. A. Mendes (Eds.), *XVIII Simposio Internacional de Informática Educativa, SIIE 2016* (pp. 187-189). Salamanca, España: Ediciones Universidad de Salamanca.

García-Peñalvo, F. J. (2016c). What Computational Thinking Is. *Journal of Information Technology Research, 9*(3), v-viii.

García-Peñalvo, F. J., Reimann, D., Tuul, M., Rees, A., & Jormanainen, I. (2016). *An overview of the most relevant literature on coding and computational thinking with emphasis on the relevant issues for teachers.* Belgium. doi: https://doi.org/10.5281/zenodo.165123

Grandell, L., Peltomäki, M., Back, R. J., & Salakoski, T. (2006, January). Why complicate things?: introducing programming in high school using Python. In *Proceedings of the 8th Australasian Conference on Computing Education-Volume 52* (pp. 71-80). Australian Computer Society, Inc.

Guo, P. J. (2013, March). Online python tutor: embeddable web-based program visualization for cs education. In Proceeding of the 44th ACM technical symposium on Computer science education (pp. 579-584). New York, EEUU: ACM. doi: https://doi.org/10.1145/2445196.2445368

Guzdial, M. (2015a). Learner-centered design of computing education: Research on computing for everyone. *Synthesis Lectures on Human-Centered Informatics, 8*(6), 1-165. doi: https://doi.org/10.2200/S00684ED1V01Y201511HCI033

Guzdial, M. (2015b). What's the best way to teach computer science to beginners? *Communications of the ACM, 58(*2), 12-13. doi: https://doi.org/10.1145/2714488

Hazzan, O., Gal-Ezer, J., & Blum, L. (2008, March). A model for high school computer science education: The four key elements that make it! In *ACM SIGCSE Bulletin, 40*(1), 281-285. doi: https://doi.org/10.1145/1352322.1352233

Heintz, F., Mannila, L., & Färnqvist, T. (2016, October). A review of models for introducing computational thinking, computer science and computing in K-12 education. In *Proceedings of 2016 IEEE Frontiers in Education Conference* (FIE), (pp. 1-9). EEUU: IEEE. doi: https://doi.org/10.1109/FIE.2016.7757410

Hubwieser, P., Giannakos, M. N., Berges, M., Brinda, T., Diethelm, I., Magenheim, J., ... & Jasute, E. (2015, July). A global snapshot of computer science education in K-12 schools. In *Proceedings of the 2015 ITiCSE on Working Group Reports* (pp. 65-83). New York, EEUU: ACM. doi: https://doi.org/10.1145/2858796.2858799

Ihantola, P., Helminen, J., & Karavirta, V. (2013, November). How to study programming on mobile touch devices: interactive Python code exercises. In *Proceedings of the 13th Koli Calling International*

Conference on Computing Education Research (pp. 51-58). New York, EEUU: ACM. doi: https://doi.org/10.1145/2526968.2526974

Kirschner, P. A. (2017). Stop propagating the learning styles myth. *Computers & Education, 106*, 166-171. doi: https://doi.org/10.1016/j.compedu.2016.12.006

Koulouri, T., Lauria, S., & Macredie, R. D. (2015). Teaching introductory programming: a quantitative evaluation of different approaches. *ACM Transactions on Computing Education (TOCE), 14*(4), Article 26. doi: https://doi.org/10.1145/2662412

Mészárosová, E. (2015). Is Python an Appropriate Programming Language for Teaching Programming in Secondary Schools? *International Journal of Information and Communication Technologies in Education, 4(*2), 5-14. doi: https://doi.org/10.1515/ijicte-2015-0005

Murphy, E., Crick, T., & Davenport, J. H. (2017). An Analysis of Introductory Programming Courses at UK Universities. *The Art, Science, and Engineering of Programming, 1*(2), Article 18. doi: https://doi.org/10.22152/programming-journal.org/2017/1/18

Nuutila, E., Törmä, S., & Malmi, L. (2005). PBL and computer programming—The seven steps method with adaptations. *Computer Science Education, 15*(2), 123-142. doi: https://doi.org/10.1080/08993400500150788

Papastergiou, M. (2009). Digital game-based learning in high school computer science education: Impact on educational effectiveness and student motivation. *Computers & Education, 52*(1), 1-12. doi: https://doi.org/10.1016/j.compedu.2008.06.004

Parker, K. R., Chao, J. T., Ottaway, T. A., & Chang, J. (2006). A formal language selection process for introductory programming courses. *Journal of Information Technology Education, 5*, 133-151.

Pears, A., Seidman, S., Malmi, L., Mannila, L., Adams, E., Bennedsen, J., ... & Paterson, J. (2007). A survey of literature on the teaching of introductory programming. *ACM SIGCSE Bulletin, 39*(4), 204-223. doi: https://doi.org/10.1145/1345375.1345441

Peña Ros, R. (2015a). Paseo por la programación estructurada y modular con Python. *ReVisión, 8*(1), 17-17.

Peña Ros, R. (2015b). Python como primera aproximación a la programación. *ReVisión, 8(*2), 17-29.

Porter, L., Guzdial, M., McDowell, C., & Simon, B. (2013). Success in introductory programming: What works? *Communications of the ACM, 56*(8), 34-36. doi: https://doi.org/10.1145/2492007.2492020

Ragonis, N., Hazzan, O., & Gal-Ezer, J. (2010, March). A survey of computer science teacher preparation

programs in Israel tells us: computer science deserves a designated high school teacher preparation! In *Proceedings of the 41st ACM technical symposium on Computer science education* (pp. 401-405). New York, EEUU: ACM. doi: https://doi.org/10.1145/1734263.1734402

Riesco Albizu, M., Díaz Fondón, M., Álvarez Gutiérrez, D., López Pérez, B., Cernuda del Río, A., & Juan Fuente, A. (2014). Informática: materia esencial en la educación obligatoria del siglo XXI. *ReVisión, 7(*3), 53-60.

Stefik, A., Daleiden, P., Franklin, D., Hanenberg, S., & Tichy, W. (2015). *Programming Languages and Learning*.

Stoilescu, D., & Egodawatte, G. (2010). Gender differences in the use of computers, programming, and peer interactions in computer science classrooms. *Computer Science Education, 20*(4), 283-300. doi: https://doi.org/10.1080/08993408.2010.527691

Tabet, N., Gedawy, H., Alshikhabobakr, H., & Razak, S. (2016, July). From Alice to Python. Introducing Text-based Programming in Middle Schools. In *Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education* (pp. 124-129). New York, EEUU: ACM.

Tang, T., Rixner, S., & Warren, J. (2014, March). An environment for learning interactive programming. In *Proceedings of the 45th ACM technical symposium on Computer science education* (pp. 671-676). New York, EEUU: ACM. doi: https://doi.org/10.1145/2538862.2538908

Tedre, M., & Denning, P. J. (2016, November). The long quest for computational thinking. In *Proceedings of the 16th Koli Calling Conference on Computing Education Research* (pp. 24-27). doi: https://doi.org/10.1145/2999541.2999542

Tollervey, N. H. (2015). *Python in Education*. Python in Education.

Van Rossum, G. (1999). Computer programming for everybody. *Proposal to the Corporation for National Research Initiatives.*

Weintrop, D. (2015, February). Minding the gap between blocks-based and text-based programming. In *Proceedings of the 46th ACM Technical Symposium on Computer Science Education* (pp. 720-720). New York, EEUU: ACM.

Zingaro, D., & Porter, L. (2014). Peer Instruction in computing: The value of instructor intervention. *Computers & Education, 71,* 87-96. doi: https://doi.org/10.1016/j.compedu.2013.09.015