Entrevista En Diferido 8: Jose Carlos.

1 de Mayo de 2019 a las 23:41

Hoy comenzamos una entrevista a Jose Carlos, profesor de secundaria, usuario de OpenBSD y Python.

Antes de todo, gracias por participar y apoyar este proyecto. La primera pregunta es una presentación por parte del entrevistado, para que los lectores te puedan conocer un poco.

Entrevista En Diferido: **¿Te podrías presentar en unas líneas?**

Jose Carlos: Hola a todos los suscriptores del canal. En primer lugar, agradezco a José la invitación. Me sorprendió porque tengo un perfil diferente a los anteriores entrevistados. No tengo un blog activo, tampoco uso mucho Twitter o Mastodon, un poco en Telegram... En fin, mejor vamos con la presentación.

Parafraseando a Mark Twain: "No tuve tiempo de escribir una presentación corta, así que escribí una larga en su lugar." Si me tuviera que definir, diría que soy una persona muy curiosa que espera no dejar nunca de serlo... pero eso no tiene nada de particular (imagino que todas las personas que lean esto pensarán lo mismo de ellas). Así que tendré que contar algo diferente de mí con la esperanza de que pueda resultar interesante a alguien.

Nací en un pueblo de Valencia al final del 'baby boom' y creo que lo que más me influyó (además de mi familia y amigos) fue la serie documental 'Cosmos', del gran Carl Sagan. Siempre me atrajo la ciencia ficción (series y libros). Ojeaba a escondidas los libros de electrónica de mi hermano mayor. Por eso hice FP de Electrónica Industrial. Mis padres compraron un Amstrad CPC-6128 y comencé a jugar primero con BASIC y luego con Pascal. Al acabar FP estudié la carrera de Informática de la que -con los años- sólo me arrepiento de no haber profundizado más. En esos años conocí a mi mujer, compañera y apoyo fundamental en mi vida, que más tarde me regaló un hijo e hija de los que me siento muy orgulloso.

Comencé mi carrera como informático de mantenimiento en un Instituto Tecnológico pero en cuestión de meses me tuve que reconvertir en jefe del nuevo departamento de Proyectos Informáticos para el Instituto y sus empresas asociadas. Fue entonces cuando 'vi la luz' del software libre (segunda mitad de los 90), dirigiendo proyectos web con LAMP cuando todo el mundo usaba Microsoft. De ahí di el salto a una pequeña consultora y luego a otra más grande (multinacional). Una parte importante de mi trabajo consistía en evaluar herramientas que nos dieran una ventaja competitiva, y el software libre nunca defraudó. Conocí Python y OpenBSD a finales de los 90, lenguaje y sistema operativo que no he dejado de seguir y que he propuesto en proyectos siempre que he podido. Desgraciadamente el perfil que desempeñaba no me permitía

'meterme en harina' aunque sí que hice mucha revisión de código y diseño funcional.

Finalmente, hace quince años, di el salto a la docencia. Desde entonces soy Profesor de Enseñanza Secundaria por la especialidad de Informática. Eso significa que puedo dar clase a todos los niveles de la ESO, Bachillerato y Ciclos Formativos de la Familia de Informática y Comunicaciones. La mitad de estos años los he pasado en la Conselleria de Educación de la Generalitat Valenciana, en dos Servicios de Informática: uno dedicado a los Centros Educativos (donde participé en el proyecto LliureX) y el otro dedicado a la Gestión Académica.

Este curso es el primero después de ocho años apartado de las aulas, lo cual me permite retomar la docencia con energías renovadas y con ganas de experimentar un montón con los alumnos. Mi objetivo: transmitirles mi pasión por la Informática y que aprendan a ser autónomos y utilizar con sentido crítico la Informática. Además, el pasado noviembre fui elegido presidente de la Asociación de Profesores de Informática de la Comunidad Valenciana (@APICV). Desde 2013 coordino las redes sociales de la asociación (básicamente Twitter y Telegram).

Como @JoseAJimenez soy también muy fan de Telegram (estoy suscrito a todos sus grupos y canales... creo) y de 'podcasts' (muchos más de los que puedo escuchar). También soy aficionado a la Filosofía (aunque en FP no existía la asignatura) y, junto con la Historia, creo que cada vez son más necesarias para la sociedad.

EED: Para empezar , unas cuantas preguntas cortas para conocerte mejor y tú entorno tecnológico.

¿Ordenador principal?
¿Sistema operativo principal?
¿Lenguaje de programación?
¿Libro o ebook?
¿Recurso de programación(web,curso,libro,aplicación...) imprescindible?
Para aprender a programar¿Leer libro,ver un video o hacer un curso?

JC:

Ordenador principal:

Desde hace más de 10 años compro equipos de segunda mano pero de 'marca'. Mi ordenador principal es un portátil Lenovo Thinkpad x220. Me encanta la serie X de Thinkpad. No tengo grandes necesidades así que escogí un equipo ligero y duradero que estuviera bien soportado en OpenBSD. Aunque siendo estrictos, el ordenador que más horas uso al día es el del aula: APD con un i7-4790 3.5GHz, 16GB de RAM DDR3 y dos discos SATA III de 2TB, dos tarjetas de red... Actúa como servidor del aula: servidor de ficheros, router, DHCP, web local, proxy, control de aula, actualización de equipos cliente, etc.

Sistema operativo principal:

Tengo multiarranque en el portátil: Ubuntu y OpenBSD. A veces tengo que compartirlo con la familia y no se aclaran mucho con el gestor de ventanas que uso en OpenBSD (i3). Aun así utilizo OpenBSD menos de lo que me gustaría 😔. En el servidor del aula tengo instalado LliureX Aula 16 (que es la distribución educativa de la Generalitat Valenciana, derivada de Ubuntu) que viene con un montón de utilidades para la gestión del aula.

Lenguaje de programación:

Soy un gran defensor de Python como primer lenguaje de programación, mejor que LOGO, BASIC, Pascal o incluso Scratch. Una sintaxis simple, con una buena curva de aprendizaje, con el techo muy alto y que fomenta buenas prácticas. Aunque sirve para muchos ámbitos, no pienso que haya que hacerlo todo con Python. Para tareas más intensivas o cercanas al hardware siempre nos quedará C. También me parecen interesantes Go, Rust (sobre todo desde la aparición de WebAssembly) y Forth (un viejo conocido, ideal para pequeños sistemas empotrados).

Libro en papel o ebook:

Aunque lo prefiero, admito que soy fetichista con el libro de papel pero ¡ya no me caben en casa! Tengo el -mal llamado- síndrome de Diógenes con los libros electrónicos. Cada vez hay más listados con libros de licencia libre, ofertas en Humble Bundle, Packt Pub, etc. Al final tengo mucho más de lo que podré leer. El lector de ebooks tiene claras ventajas: no ocupa espacio, lo llevas a cualquier parte, se puede actualizar, te traduce palabras, puedes leer a oscuras... No obstante, el papel te permite ojear rápidamente, anotar más fácilmente e invita a dedicar más tiempo concentrado en la lectura.

Recurso de programación (web, curso, libro, aplicación...) imprescindible:

Si tengo que elegir un solo recurso, claramente sería un editor de código que ayude en la tarea de programar: leer, escribir, ejecutar, depurar, etc. que incluya ayuda (contextual y de referencia) pero que no sea recargado a la vista como un IDE moderno. Buscar en StackOverflow debe hacerse por algo muy específico y sabiendo lo que se hace, nunca para cosas triviales o que se puedan aprender en un curso o libro. Y éstos deberían utilizarse previamente a la programación, no durante ella (excepto los libros de referencia).

Para aprender a programar, ¿leer libros, ver vídeos o hacer un curso?

100% libros, aunque sean 'online'. Me parecen interesantes las herramientas que permiten crear libros interactivos, que incrustan trozos de código que puedan ser ejecutados directamente en contexto (p.e. usando repl.it, trinket.io o jupyter notebooks). La gran desventaja que le veo a los vídeos es la dificultad de ojear, buscar o saltarnos lo que ya sabemos. También conlleva mayor trabajo la actualización de su contenido. Sé que este es un tema especial para @JoseAJimenez porque se lo he oído más de una vez en sus podcasts. Aprovecho para aportar un estudio que habla del tema: What Is the Best Way For Developers to Learn New Software Tools? An Empirical Comparison Between a Text and a Video Tutorial https://doi.org/10.22152/programming-journal.org/2017/1/17

Con respecto a los cursos caben muchos matices porque los hay desde los que son sólo texto (sin interacción) hasta los que están completamente basados en vídeo. Los hay abiertos y cerrados, masivos y particulares, 'online' y 'offline'... Hace 7 años hice un MOOC completo (empecé otros pero sólo me dediqué a curiosear los materiales). En aquel momento le vi potencial pero sigo pensando que, como concluye el estudio que he referenciado antes, habría acabado antes con una versión en texto.

EED: Como has indicado en tu presentación, has vuelto a la docencia de informática y en relación a la enseñanza de la informática a jovenes.

¿Qué mejorarías?
¿Qué añadirías?
¿Qué eliminarías?

JC:

¿Qué mejoraría?

Ay, me has tocado la fibra... Actualmente hay mucho ruido con respecto a la enseñanza de la Informática. En la mayoría de países están apostando por una asignatura obligatoria incluso desde Primaria. Pero se encuentran con un problema: faltan docentes preparados y con los cursos que reciben no alcanzan suficiente seguridad para enseñar la materia. En España se está empezando a plantear una alternativa similar: cursos para reciclar a profesores de otras especialidades. Como curiosidad, en todas estas iniciativas, la palabra Informática brilla por su ausencia. Es sustituida por Programación, Robótica y Pensamiento Computacional.

En la Comunidad Valenciana tenemos la experiencia de ser una asignatura optativa de oferta obligada en la ESO y Bachillerato, impartida por profesores de la especialidad de Informática. Aunque es el único caso en España, tampoco es la situación ideal: muchos alumnos -y sobre todo alumnas- no la llegan a cursar, cuando desde Europa se quiere que todos los ciudadanos tengan una competencial digital básica. No nos engañemos, llevan décadas llamando 'nuevas tecnologías' a todas las diferentes aplicaciones de la Informática que sacan en cada vez más ámbitos. En verdad, 'el software se está comiendo al mundo' como explicó Marc Andreessen en 2011.

Hay un informe de la Sociedad Científica Informática de España http://www.scie.es/informatica-ensenanza-no-universitaria/ (en el que participó @APICV) donde se aborda la cuestión de una manera más profunda. Cada vez se oyen más propuestas de asignaturas de todo tipo en la educación, todas interesantes. El problema con Informática es que ya llegamos tarde. Desde hace años vivimos rodeados de aplicaciones de la Informática pero sigue siendo 'magia' para la mayoría de los ciudadanos. A nadie se le ocurre decir que Biología, Física o Química no son necesarias pero con la Informática se suele argumentar que 'los niños son nativos

digitales', lo cual es absolutamente falso.

Ahora mismo, fuera de la Comunidad Valenciana, con suerte sólo tienes la opción de cursar la optativa de Informática en 4º ESO y Bachillerato. Sin embargo, en la mayor parte de los casos es impartida por profesores de la especialidad de Tecnología (de formación muy diversa, como su temario de oposiciones). De estos, sólo algunos valientes o aquellos que sí tienen una formación previa en Informática se atreven a impartirla. Así que mi propuesta sería un plan de implantación a varios años para ofrecerla transversalmente en Primaria (haría falta adecuar Magisterio para ello) y una asignatura troncal en la mitad de cursos de la ESO y Bachillerato (optativa en el resto de cursos) impartida por docentes de la especialidad de Informática (y aquellos que se certificaran adecuadamente para poder impartirla). Para conseguir que toda la ciudadanía tuviera una competencia digital básica y efectiva. Esto beneficiaría a los estudiantes tanto si se incorporaran al mundo laboral como si siguieran estudiando, y por extensión al conjunto de la sociedad.

El peligro es acabar copiando la estrategia que no ha funcionado en otros países. Hacer una implantación con prisas, dirigida por el rédito político que pueda obtener el gobierno de turno, no es deseable. En Israel hace décadas crearon un Instituto de investigación para la enseñanza de la Informática, igualando otras materias que llevaban muchos más años con ello. Allí la Informática está consolidada desde hace años en Bachillerato como una asignatura igualada en profundidad y rigor con Física, Química, Biología, etc. Veremos qué acaba ocurriendo.

¿Qué añadiría?

El ritmo al que se incorporan o adquieren relevancia algunos conceptos en Informática obliga a renovar periódicamente el currículo. Temas como la ciberseguridad, la Inteligencia Artificial, la Ciencia de Datos, la Internet de las Cosas (IoT) y su impacto en la sociedad deberían tener en mi opinión mayor peso. El trabajo que han hecho en otros países, con larga tradición y peso en la Informática como Reino Unido o Estados Unidos, con respecto al currículo se trabajan estos aspectos.

A nivel metodológico añadiría el Aprendizaje Basado en Proyectos que viene como un guante a una asignatura como Informática. Pero para ello es necesaria una apuesta fuerte en formación y recursos. Igualmente hay muchas estrategias efectivas (comprobadas científicamente) que se desconocen y debería formar parte de nuestra formación como docentes. Como ejemplo, el libro (con licencia libre) de "Teaching Tech Together" de Greg Wilson del cual soy un fan absoluto: http://teachtogether.tech

Del mismo autor me parece muy interesante todo el estudio que está haciendo alrededor de los recursos educativos abiertos (OER). En su blog http://third-bit.com/ tiene entradas muy interesantes como “Ten Simple Rules for Creating an Effective Lesson”, “Ten Simple Rules for Collaborative Lesson Development” o su reciente "Leading Questions for Creating a Learning Commons". La situación ideal a alcanzar es la de crear Comunidades de Aprendizaje tanto entre docentes de la misma especialidad como otras comunidades transversales. Parece bastante utópico pero es realizable si se consigue la masa crítica.

¿Qué eliminaría?

Con los años me he desencantado de Scratch (y de la programación visual de bloques, en general). Este curso he empezado a usar Python desde 1º ESO y no he tenido ningún problema. Es cierto que los lenguajes visuales eliminan los problemas iniciales de errores de sintaxis (¡y también la oportunidad de aprender de esos errores!) pero más adelante te limitan (sobre todo si pensamos en la cantidad de librerías interesantes que tiene Python). Me parecen más interesantes para Primaria. Algo similar me ocurre con la robótica. Realmente consigue captar la atención de los alumnos y colocar un objetivo muy claro ante ellos, es divertido para ellos (también puede ser frustrante) pero llega un momento en que, si quieres trabajar otro tipo de algoritmos, ya no puedes emplear robots.

Por otro lado, más que eliminar, matizaría que tampoco hay que sustituir Informática por Programación. Me asusta el interés de grandes multinacionales porque los jóvenes aprendan a programar a toda costa. Como si quisieran devaluar la mano de obra de los programadores. De la misma manera que estudiar Matemáticas no pretende hacer que todos los ciudadanos sean matemáticos, la Informática tampoco debe aspirar a lo mismo. Con el mantra de 'Pensamiento Computacional' se están justificando muchas iniciativas cuando no existe evidencia de exista dicho pensamiento como otras ideas como las inteligencias múltiples, etc.

EED: Eres usuario de OpenBSD, que no es un sistema muy conocido y utilizado. Como usuario de este sistema.

¿Porqué utilizas OpenBSD?
¿Que te aporta este sistema, características, uso y ventajas?

JC:

¿Por qué utilizo OpenBSD?

Después de haber probado varias distribuciones de GNU/Linux a mediados de los 90 -cuando eran un montón de diskettes-, (RedHat, Mandrake, Debian y alguna más) me quedé con la que más me facilitaba el trabajo diario... Mandrake (derivada de RedHat). Sin embargo, se me quedó esa sensación de haber optado por el camino fácil. La oportunidad que te da un sistema operativo GNU/Linux es poder entrar a ver las tripas y aprender un montón. Pero la agilidad que demandaba mi trabajo no me permitía dedicar ese tiempo al autoaprendizaje (aunque pudiera ser potencialmente beneficioso para futuros proyectos). En casa estuve probando un poco Debian pero un día descubrí OpenBSD en un artículo de la revista Wired en 1997. Me capturó su filosofía como sistema operativo, como un amor a primera vista. De repente todo empezaba a cuadrar; todas las decisiones que tomaban me parecían las más sensatas. El único problema: no era muy usable como SO de escritorio, al menos para mí en aquel

momento.

Desde entonces fui siguiendo el proyecto, conociendo su evolución, probándolo esporádicamente... Lo llegamos a utilizar como cortafuegos en 2001 en la empresa y con algún cliente, pero poco más. Fue más tarde, al instalarlo en un equipo con multiarranque (GRUB), cuando empecé a disfrutarlo más. OpenBSD es amigo tuyo si te gusta la línea de comandos y leer la ayuda (man). Como decía el autor del primer podcast al que me suscribí (BSDTalk en 2005), "requires reading skills instead of clicking skills" (aunque no se refería a OpenBSD sino a la línea de comandos). Will Backman ya no sigue con su podcast pero fue un gran divulgador de la desconocida familia BSD (en especial OpenBSD).

Para las personas interesadas en conocer las diferencias entre Linux y BSD les recomiendo https://www.over-yonder.net/~fullermd/rants/bsd4linux/ que, aunque habla principalmente de FreeBSD permite coger las ideas principales. Otra página que puede resumir rápidamente el proyecto es http://www.openbsdjumpstart.org/ pero lo mejor es ir a la fuente: https://www.openbsd.org/

Como conceptos llamativos del proyecto citaría los siguientes:

- primer proyecto de software libre que puso en abierto su repositorio de código (CVS)
- primer proyecto de software libre en realizar de manera regular hackathons
- primer proyecto en realizar un ciclo bianual en sacar versiones (a principios de mayo y noviembre)
- desarrollan OpenSSH que es el SSH empleado en la inmensa mayoría de plataformas
- desarrollan LibreSSL (fork de OpenSSL) desde el escándalo de HeartBleed
- documentación excelente (tanto interna del código como de la ayuda man)
- código de alta calidad y muy legible (drivers incluidos, no se aceptan BLOBs)

y muchas más innovaciones en: http://www.openbsd.org/innovations.html

¿Que me aporta este sistema, caracteristicas, uso y ventajas?

OpenBSD me aporta estabilidad, tranquilidad por la seguridad, documentación formidable y un entorno de aprendizaje de buenas prácticas no sólo como SO sino como filosofía de desarrollo de software. El proyecto destaca en su web que sus esfuerzos van dirigidos a la portabilidad, estandarización, corrección, seguridad proactiva y criptografía integrada. Creo que se quedan cortos pero mi opinión está condicionada por mi amor declarado... 😍

Como he dicho antes, el principal uso de OpenBSD sería para cortafuegos, servidores bastión, etc. pero, con lo hostil que se volvió Internet hace ya tiempo, mi principal uso es como escritorio. Aunque actualmente tengo i3 como gestor de ventanas, me estoy planteando eliminar el multiarranque y dejar únicamente OpenBSD en el portátil pero tendré que volver a Gnome (para cuando lo use mi mujer o mis hijos). Mis necesidades son simples: línea de comandos para seguir aprendiendo y un navegador (cada vez hago más cosas en la nube por coordinarme con otras personas). Pero insisto, OpenBSD no es para el público general, acostumbrado a Windows o al que Ubuntu ya

le supone un poco de esfuerzo.

Por otro lado, si en algún momento vuelvo a dar ciclos formativos, me gustaría explotar sus posibilidades en FP y preparar un módulo con OpenBSD como SO base para enseñar servicios de red sanos (la sintaxis de los archivos de configuración suele ser muy sencilla).

Por último, si hay alguna persona interesada en participar en la comunidad de OpenBSD España la invito a entrar en @OpenBSD_es □□ Otra cosa que se me olvidaba es la mina de sabiduría que es la lista de correo de discusión genérica: https://marc.info/?l=openbsd-misc (muy recomendable si os gustan las tripas de los SO)

EED: Has comentado que eres defensor de Python, yo te conocí en el grupo de Telegram de Python. Respecto a Python...

¿Cuál es el uso principal que le das?
¿Porqué lo utilizas?

JC:

¿Cuál es el uso principal que le das?

El uso que he dado a Python ha ido variando en el tiempo. Ahora mismo únicamente lo utilizo para enseñar a programar. Por aportar algo más, explico mi relación con el lenguaje desde que lo conocí:

Me hablaron de Python unos meses antes de que sacaran la versión 1.5.2 (abril de 1999) y decidí imprimir el tutorial oficial para leermelo en casa. Cuando llevaba la mitad leído me decidí a hacer mi primer 'script'. En un cuarto de hora había conseguido hacer -con poco más de 20 líneas- un programa que se descargaba la viñeta de mi autor favorito de la web de un periódico. Por defecto descargaba la viñeta del día pero lo convertí en una función para pedir la de cualquier día. Fue trivial hacer un bucle para descargar todas las publicadas. Diez minutos más y conseguí que inspeccionara la cabecera del archivo gráfico para determinar el formato (a veces, aunque tenía una extensión GIF o JPG, tenía realmente el otro formato) y escribirlo correctamente en disco.

Esa experiencia fue determinante para mi enganche al lenguaje. No conocía en aquel momento otro que pudiera hacer lo mismo con tan pocas líneas de código (y además con la apariencia de pseudocódigo). No obstante no pude darle uso en el trabajo hasta poco más de un año después (cuando di el salto a la empresa privada). En ese momento descubrí un servidor de aplicaciones basado en Python llamado ZOPE (2000). Realizamos algún proyecto con él pero realmente lo que consiguió cambiar la imagen de 'lenguaje de juguete' que tenía Python en la empresa fue la llegada de Plone unos meses después: todo un CMS programable y completamente configurable construido encima de ZOPE.

Más tarde, al final de mi primer curso como profesor interino, descubrí el 'web framework' Django (2005). De nuevo me enganchó porque veía la ventaja de un sistema desacoplado, en el que podías usar las partes que te interesaran y en el que todo quedaba más visible y controlado que con ZOPE o Plone. Como ya no estaba en la empresa, sólo me dediqué a seguir un par de tutoriales para hacer un par de pequeños proyectos. Pero ahí quedó la cosa.

No fue hasta comenzar el lapso de 8 años en la Conselleria de Educación cuando retomé mi contacto con la programación. Sobre todo para ETL (Extract, Transform and Load) donde descubrí Pandas, que me permitía hacer un manejo de datos con poquísimas líneas. Durante los últimos años allí fui afianzando la idea de usar Python como primer lenguaje de programación (cuando volviera a la docencia) y fui recopilando recursos en una lista pública en GitHub: https://github.com/quobit/awesome-python-in-education (son todos recursos en inglés).

En la última evaluación de este curso estoy probando diferentes recursos y métodos para enseñar programación a los diferentes niveles de ESO y Bachillerato, lo cual me permite aprender un montón de los alumnos. Mi objetivo es elaborar una nueva clasificación que permita a cualquier docente tener acceso a recursos orientados a diferentes rutas de aprendizaje. Para ello creé un grupo en Telegram (que de momento está durmiente) orientado a debatir estrategias y recursos de Python en la enseñanza: @PythonEsp_Edu

¿Por qué lo utilizas?

Aunque ya comenté los motivos por los que me parecía un lenguaje ideal para enseñar a programar, me gustaría añadir algún motivo más:

- el ecosistema tan rico que posee (y que no para de crecer): hay librerías de terceros de cualquier cosa imaginable (no hay más que darse un paseo por PyPI). Principalmente estoy pensando en IA y Ciencia de Datos pero también webscraping, acceso a APIs de cualquier servicio que podamos pensar, etc.
- el alto nivel de expresividad que aporta tanto el lenguaje como el ecosistema permite desarrollos rápidos de aplicaciones (con muchas menos líneas de código que otros lenguajes)
- pero sobre todo, el elemento principal que hace que Python sea tan especial, es la comunidad tan inclusiva (algo fundamental en el ámbito de educación). Con el paso de los años cada vez tengo más claro que lo más importante de un proyecto de software libre es la comunidad que tiene, más que el código que produce...

EED: En tu presentación has comentado que eres aficionado de la Filosofía e Historia ,me gustaría saber en su experiencia como docente de informática.

¿Crees que existe relación entre las humanidades y la tecnología?
¿Has aplicado o piensas aplicar temáticas propias de las humanidades en tus clases de informática?

JC:

¿Crees que existe relación entre las humanidades y la tecnología?

Por supuesto que hay relación entre las humanidades y la tecnología. Para empezar, la Filosofía es la madre de la Ciencia. La Lógica es una parte fundamental de la Filosofía, de las Matemáticas y de la Informática. Otro ejemplo directo que muestra la relación entre ambas es la disciplina de Interacción Humano-Máquina. O el caso de la Inteligencia Artificial, Computabilidad, Impacto en la Sociedad (aspectos éticos), etc.

Tanto es así que hace pocos años se creó un doble grado en la Universidad de Oxford llamado "Computer Science and Philosophy" y cada vez en más universidades, por ejemplo en España la UOC tiene el Internet Interdisciplinary Institute (IN3) "especializado en el estudio de Internet y de los efectos de la interacción de las tecnologías digitales con la actividad humana".

Para ampliar información sobre la relación entre Filosofía y Computación podéis ir a http://www.philocomp.net/ o también la relación entre Filosofía e Información http://www.philosophyofinformation.net/ (principalmente en los aspectos éticos - recordemos las noticias de Cambridge Analytica, etc.)

Con respecto a la Historia: la primera enseñanza de la Historia es que se repite. La segunda, que se nos olvida que se repite. Y así pasa también con la Informática. Recomiendo encarecidamente un vídeo de Bret Victor donde se puede comprobar de una manera muy efectiva: The Future of Programming https://www.youtube.com/watch?v=8pTEmbeENF4

¿Has aplicado o piensas aplicar temáticas propias de las humanidades en tus clases de informática?

Me encanta la definición de la Informática que dan en uno de mis libros favoritos "Foundations of Computer Science" (Aho, Ullman): La Informática es la mecanización de la abstracción, en una frase que tiene mucha 'miga' en mi opinión. Partiendo de ahí, diría que intento dar un toque filosófico en cualquier aspecto posible de mis clases de Informática. Como he indicado antes son muchas las conexiones. Aunque veamos con los más mayores algún episodio de "Black Mirror" (si no conocéis la serie, es muy recomendable) y analicemos los dilemas éticos, o los experimentos que se hacen para entrenar los vehículos autónomos en caso de potenciales accidentes, o los efectos de las redes sociales en sus vidas, siempre se puede dar un enfoque reflexivo y crítico a practicamente cualquier contenido de la asignatura (de hecho, en el temario hay aspectos de comportamiento en la red -netiqueta- e impacto en la sociedad).

Tampoco es algo que haga de manera estructurada. Antes no lo hacía y ahora veo en sus caras como se paran a pensar (y eso es muy positivo). Empatizo todo lo que puedo con ellos y reflexionamos juntos. Desde luego es un enfoque transversal que surge de manera espontánea muchas veces (otras está planificado). Un caso concreto que puede servir de ejemplo: cuando aprendemos a usar diccionarios en Python, hacen un programa que analiza un texto (cantidad de palabras diferentes y frecuencias). A continuación, escogemos diferentes textos del Proyecto Gutenberg por ejemplo del s. XIX y nos vamos acercando a la actualidad. Todos llegan rápidamente a la conclusión

de que el vocabulario que se emplean en los libros es cada vez más reducido.

Para finalizar, otro vídeo que les hace reflexionar: ¿Así que tú crees que la tecnología es neutra? - Jesus M González Barahona https://www.youtube.com/watch?v=uRXWLLVWVMY

EED: Seguimos con el tema de Python, en la pregunta anterior has comentado porque lo utilizas y has enumerado un conjunto de características . Pero me gustaría profundizar un poco mas, desde el punto de vista mas técnico...

¿Qué funcionalidad destacarías de Python?
¿Qué herramienta de desarrollo de Python crees que todo desarrollador en Python debería conocer?

JC:

¿Qué funcionalidad destacaría de Python?

Aunque en un primer momento la sintaxis sea muy importante (baja muchísimo la barrera de entrada) diría que el hecho de ser interpretado le da varias ventajas.

Probar directamente cualquier porción de código para comprobar si funciona como esperamos y así incorporarlo a nuestro programa. O el poder de introspección que te da: interrogar tipos, métodos, atributos, identificadores, ayuda, etc. por no mencionar módulos como 'inspect' o 'dis'.

Digamos que esa característica potencia el valor para prototipar en Python y, lo bueno, es que para muchos casos resulta suficiente. Le 'regala' tiempo al programador y no se interpone en su tarea de encontrar una solución.

Evidentemente, no todo son ventajas pero yo diría que compensa. Muchas veces los recién incorporados al lenguaje (provenientes de otros compilados) se quejan precisamente de su lentitud. Para ellos les pediría que reflexionaran honestamente después de haber leído este par de artículos: https://www.paypal-engineering.com/2014/12/10/10-myths-of-enterprise-python/ y https://medium.com/pyslackers/yes-python-is-slow-and-i-dont-care-13763980b5a1

¿Qué herramienta de desarrollo de Python crees que todo desarrollador en Python debería conocer?

Sin duda los entornos virtuales. Es una asignatura pendiente del lenguaje que últimamente se está intentando solucionar desde varias inciativas.

En cuanto una persona comienza a programar y a utilizar módulos de terceros, es muy común que acabe liándose con el Python del sistema, el comando 'pip', etc.

A este respecto me gustaría recomendar los recursos de Real Python y en este caso

concreto https://realpython.com/python-virtual-environments-a-primer/

EED: Volvemos a la docencia, eres profesor de informática en la Enseñanza Secundaria. Te propongo una situación ideal, tienes todo tipo de recursos de forma ilimitada para impartir informática y conseguir un curso perfecto.

¿Cómo impartirías ese curso? ¿Qué materia, recursos y metodología utilizarías?

JC: Siento como que durante los 8 años que no he estado en las aulas ha pasado algo que ha cambiado las reglas del juego: el smartphone. A diferencia de nosotros, los adolescentes reciben una enorme cantidad de estímulos audiovisuales (Youtube, Instagram, memes...). Viven en un mundo que reclama constantemente su atención y se convierten en consumidores multitarea. Los materiales que son más efectivos con ellos no pueden incluir una gran cantidad de texto sobre el que haya que reflexionar más de 15 minutos. Su pomodoro tiene menos tiempo. Por otro lado, cambiar de tarea les cuesta menos y eso puede ser una ventaja en algunos contextos.

Así pues, los recursos a emplear deberían contener elementos que no les resulten ajenos. Esa evolución ya se podía observar en los libros de texto, con muchas imágenes, esquemas, texto resaltado y cajas con información relacionada (cada vez más con enlaces a recursos 'on-line'). Esto no significa que el texto desaparece pero desde luego tiende a acortarse, a ser más sintético y directo.

Un recurso interactivo es más adecuado pero también requiere más esfuerzo por parte del docente. Fijaos en este recurso: Computer Science Field Guide An online interactive resource for high school students learning about computer science http://csfieldguide.org.nz/en/ Lo primero que llama la atención es la cantidad de elementos interactivos y vídeos cada pocos párrafos. Si miramos el repositorio correspondiente: https://github.com/uccser/cs-field-guide podemos observar el trabajo que hay detrás. Si os fijáis en la página de contributors la mayoría de los commits son de 2-4 personas. Es fundamental la creación de una comunidad alrededor de un recurso educativo abierto (OER) como este para asegurar su mantenimiento y mejora continua. Recuerdo de nuevo a Greg Wilson con su "Ten simple rules for collaborative lesson development" https://doi.org/10.1371/journal.pcbi.1005963

Volviendo al planteamiento que me hacías, "situación ideal, recursos ilimitados, curso perfecto...": no creo en el curso perfecto ni en el método perfecto. Pero, siguiendo el hilo de lo comentado anteriormente, el objetivo inalcanzable a perseguir sería crear el material adecuado alrededor del cual habría una comunidad de usuarios (otros docentes) que como comunidad de prácticas/aprendizaje realizaría una mejora continua del material.

Las características del material serían:

- Inclusión de vídeo: siendo realistas, lo más práctico sería traducir material ya existente. Por ejemplo, Crash Course Computer Science https://www.youtube.com/playlist?list=PL8dPuuaLjXtNlUrzyH5r6jN9ulIgZBpdo (los primeros de la lista ya tienen traducciones y, por cierto, se da bastante espacio a la Historia en estos vídeos). O algunas playlists del canal de Youtube de Code.org.
- Introducir temas de reflexión sobre el impacto en la sociedad con algún capítulo de Black Mirror. El hecho de que sea material en inglés se tiene que ver como una ventaja: cuanto mayor exposición haya al inglés, más útil para la Informática.
- Exponer los conceptos de Hardware, Sistemas Operativos, Redes y Multimedia basándose en los smartphone que todos usan constantemente.
- Introducir conceptos de programación a lo largo de todo el curso. Dependiendo de la cantidad de horas semanales, una hora o media hora todas las semanas a explicar un concepto.
- Integrar todo el material anterior en Jupyter Notebooks por ser una herramienta asequible y con facilidad para actualizar contenido.
- Emplear metodología ABP (aprendizaje basado en proyectos) y crear al menos uno o dos proyectos por evaluación.

En cuanto a la materia que impartiría, dependería del nivel/curso del alumnado y de sus conocimientos previos. Ese es otro de los problemas con los que nos encontramos al ser una materia optativa: no podemos presuponer ningún conocimiento previo al alumno, sea cual sea el curso. Lo cual nos obliga a nivelar conocimientos y, como consecuencia, bajar el nivel general que nos habíamos planteado. En el Informe de la SCIE que comenté anteriormente "La Informática en la Enseñanza no Universitaria" se plantea contenidos para cada uno de los niveles. Pero claro, presuponiendo una troncalidad (no optativa) de la materia.

No se puede esperar que un estudiante pueda aprender toda la Informática básica con dos o tres horas semanales durante un solo curso. Como tampoco se puede esperar que aprendan a programar en unas pocas horas, como algunos libros ponen en sus portadas. Lo cual me recuerda un texto de Peter Norvig que deberían leer todos los que esperan ese milagro de programar con cuatro ideas básicas y apenas esfuerzo: "Teach Yourself Programming in Ten Years" http://norvig.com/21-days.html

EED: Has comentado que durante un tiempo participastes en el proyecto LliureX.

¿Qué es LliureX?¿En que estado se encuentra el proyecto actualmente?

JC:

¿Qué es LliureX?

LliureX http://lliurex.net es una distribución GNU/Linux basada en Ubuntu orientada al sistema educativo valenciano. Se desarrollan varias adaptaciones (localizadas en las dos lenguas cooficiales): una genérica llamada escriptori de la que derivan el resto, infantil para los más pequeños, música con un kernel de baja latencia, pyme pensada para

pequeñas empresas y como estrella el modelo de aula y de centro . El modelo de aula consiste en un servidor y el resto de equipos cliente pueden tener a su vez varias adaptaciones (en función del hardware disponible, clientes ligeros, semiligeros o pesados).

No sólo se hace una recopilación y adaptación de aplicaciones educativas (tanto de contenidos como de gestión). Lo cual no deja de ser complicado. También se incluyen muchos desarrollos propios de herramientas para facilitar la tarea docente; los casos más claros son los modelos de aula y de centro. Así como la constante lucha contra el envejecimiento del parque informático de los centros, que obliga a vigilar el rendimiento tanto en hardware viejo como más reciente (lo cual implica una versión para equipos más viejos con menos recursos o incluso para Raspberry Pi).

Con relativa frecuencia se escucha en algunos foros que eso de que cada Comunidad Autónoma tenga una distribución es tirar el dinero, lo cual es completamente falso: la Generalitat Valenciana ahorra mucho dinero en licencias (recordemos que se trata no sólo de un sistema operativo de escritorio sino de también de suite ofimática, edición multimedia, bases de datos, servidores de red y un largo etcétera). Por otro lado, me hace gracia que, cuando jamás ha habido un solo convenio entre Comunidades Autónomas en cuarenta años de Constitución, se diga que habría que hacer una distribución (más grande y libre ;-). Lo mismo se podría aplicar a las decenas o centenares de distribuciones que hay. Como ocurre en Biología, la diversidad es riqueza y la evolución se encargará del resto.

¿En qué estado se encuentra el proyecto actualmente?

El proyecto comenzó su andadura en 2005 por iniciativa del Conseller de Educación en aquel momento, Esteban González Pons (por entonces todo un adalid del software libre dentro del PP). Con el cambio de gobierno autonómico, hace 4 años, se ha redoblado la apuesta con una iniciativa de implantación en parte de la administración autonómica y en algunos ayuntamientos (tienen usuarios hasta en latinoamérica). Desde sus comienzos hasta ahora, la evolución ha sido constante. Siempre se ha ido mejorando en la medida de las posibilidades y podríamos decir que está muy bien consolidado. Por ejemplo, el equipamiento de dotación informática (tanto dotación nueva como renovación de equipos) de los centros viene con LliureX preinstalado desde hace unos años. Y existe una comunidad de usuarios cada vez mayor que comparte sus experiencias, tienen foros de preguntas y respuestas, etc.

No obstante, me gustaría ver mejoras en dos aspectos principales:

- Modelo de participación (en incluso gobernanza), abriéndose a la comunidad para que participe incluso en el código propio de LliureX (aunque siempre ha tenido su código fuente disponible en un repositorio de Subversion, hace no mucho se han publicado ya 244 repositorios de las distintas partes del proyecto en GitHub, facilitando la posibilidad de aportar parches).
- Documentación técnica que facilite las aportaciones de código por parte de docentes y alumnos de Informática. Es un esfuerzo muy grande pero necesario para conseguir que su comunidad crezca y madure como tal.

EED:

¿Los ordenadores pueden pensar?

Mucho ha llovido desde que Alan Turing se lo preguntó. Su respuesta fue que sí y que en algún momento podrían pasar el 'juego de imitación' que propuso (ahora conocido como el Test de Turing). Según la definición común de 'pensar' y de 'ordenador', podríamos decir que en contextos acotados ya se ha superado la prueba. Todavía estamos bajo el shock de la presentación de Google Dúplex: https://www.xataka.com/robotica-e-ia/asi-funciona-google-duplex-el-sistema-que-se-pone-al-telefono-por-ti-y-que-da-un-poquito-de-miedo y llega el gigante de comercio electrónico asiático, Alibaba y redobla la apuesta: https://www.technologyreview.es/s/10794/alibaba-reta-google-con-una-ia-que-conversa-como-los-humanos

Aún así, muchos nos quedamos con sabor a poco. 'Eso no es realmente pensar', nos decimos, 'se trata de una simulación'. El debate da para muchos libros (como existen de hecho, y más que se escribirán). ¿Pueden tener voluntad? ¿Pueden sentir emociones? ¿Ser creativos? ¿Autoconscientes? O, dándole la vuelta a la pregunta, ¿es el cerebro un ordenador? Podríamos decir que sí, si pensamos que el funcionamiento del cerebro es computable (una vez tengamos el suficiente nivel de detalle para simular su comportamiento). Entramos en terreno metafísico fácilmente, y ahí podría no acabar de escribir...

Por zanjar un poco el asunto, desconfío de la 'nueva IA' en el sentido de que no es Inteligencia General. Para ello, creo que es necesario combinar las técnicas actuales, que se han visto efectivas para contextos muy reducidos (aunque no exentos de posibles problemas de sesgos) con muchas otras técnicas tradicionales en la IA 'clásica' y todas las recombinaciones que puedan surgir de dicha hibridación. Un efecto de toda esta investigación es que también nos lleva a conocernos mejor. Y en esa búsqueda, es posible que se alcance finalmente ese objetivo. ¿Llegará la singularidad con ello? No lo sabemos, pero hay cada vez más películas que juegan con esa idea. Mi favorita: 'Ex Machina' sin dejar tampoco de recomendar 'West World'.

¿Es necesario que los ordenadores piensen?

Pues en principio no lo veo necesario. Es suficiente con que se dediquen a lo que hacen mejor (aún hay recorrido para que lo hagan mejor todavía). Pero el comienzo de la pregunta ('Es necesario?') me da que pensar... ¿Acaso se dirige la investigación siempre por la necesidad o simplemente porque puede ser hecho (sin importar las consecuencias)? Para estar preparado o incluso evitar un cambio 'desfavorable' para la humanidad surgen sociedades como OpenAI https://openai.com/

EED: Como última pregunta de la entrevista , una pregunta un poco diferente

¿Qué te hubiera gustado que te preguntase? Evidentemente la respuesta

será tuya.

JC: Pues esta pregunta es la más difícil porque me tientan muchos temas. Pero voy a profundizar en el que ha suscitado una pregunta del público:

¿Por qué interesa la filosofía a un informático?

La verdad es que no sé qué me interesó antes. Creo que me compré mi primer libro de filosofía (Fundamentos de Filosofía de Bertrand Russell) con 16-17 años. Pura curiosidad. No me lo quería perder por estar haciendo FP en lugar de BUP y COU. Luego leí otros ensayos cortos de Russell (y me confirmaron en mi ateísmo). Más tarde encontré un nexo de unión con la Lógica. Fue gracioso ayudar a una de mis hermanas con la asignatura. En electrónica digital, con las puertas lógicas, trabajamos muchísimo. Al acabar de ayudarla me preguntó: "¿y tú cómo sabes todo esto? (si estás en FP)". Se lo expliqué y me contestó: "¡Pues yo pensaba que esto no servía para nada!"

Años más tarde, en la universidad tuve mi asignatura favorita: "Informática Teórica". Aglutinaba Teoría de Lenguajes Formales, Autómatas y Computabilidad. Ahí conocí el trabajo de Chomsky (del cual luego me atrajeron sus ideas políticas). Mi mujer estudiaba Filología y al hojear el libro de la asignatura me preguntó: "Pero ¿este Chomsky es *mi* Chomsky?". Y yo le contesté: "No, es *nuestro* Chomsky". La segunda vez que alguien se sorprendía de una aplicación directa. Creo que fue a partir de esa época cuando me empecé a interesar de manera más seria por la Filosofía.

Al fin y al cabo, si la Informática consiste en "mecanizar abstracciones", son precisamente las abstracciones más brillantes y elegantes las que permiten avanzar en nuestro campo. Estaría bien hacer un Museo de Informática pero, en vez de con hardware (fijaos, renegando de mi pasado de electrónica y sistemas), con los conceptos abstractos que han permitido llegar tan lejos. Se le podría llamar "Parque de Abstracciones" ;-)

Y a estas alturas de la vida me pregunto yo, ¿acaso no hacemos en el fondo lo mismo que la filosofía? Creamos abstracciones y operamos/razonamos con ellas, creamos sistemas y metasistemas, hacemos modelos del mundo... Aunque estamos en un punto en el que los modelos los empieza a hacer la máquina y tenemos dificultad en ver las consecuencias. Y aquí entra el aspecto ético (¡Filosofía de nuevo!). Y eso sin movernos al terreno metafísico donde nos podríamos replantear la estructura de la realidad en última instancia representada como bits (hay varios guiños de eso en la saga de Matrix).

Por último, no quiero dejar de decir que la Filosofía también nos permite analizar y criticar conceptos que consideramos tan consolidados que están mitificados. Y eso nos permite avanzar. Así que, cuando alguna vez oigo a algún compañero decir que deberían eliminar la Filosofía de las aulas, me enervo. Es renegar del valor del pensamiento riguroso, capaz de ponerlo todo patas arriba.

La Filosofía nos puede dar respuestas que no nos gustan, nos puede exigir un esfuerzo al que no estábamos dispuestos pero ahí está la elección: escoger como Cifra la felicidad ignorante o la pastilla roja como hace Neo sin titubear...

EED: El último día de la entrevista, no hay pregunta. Me gustaría darte las gracias por participar y espero que la entrevista haya sido una buena experiencia y entretenida.

Si puedes indicarnos tus métodos de contacto y si quieres promocionar algún proyecto tuyo como podcast, web, canal y grupos de Telegram o cualquier otra cosa, tienes este espacio para tí.

Por último, me gustaría que me recomendaras a alguien que creas que estaría dispuesto a participar en una futura entrevista.

Muchas gracias y hasta la próxima.

JC: En realidad te tengo que dar yo las gracias. Gracias por haber pensado en mí. Gracias por el formato. Ha sido un ejercicio de introspección muy interesante.

También me ha servido para arrinconar un poco más el síndrome del impostor (ver imagen más abajo).

La forma de contacto más directa es por Telegram a @quobit o por cualquier otro medio disponible en keybase.io/quobit. No consigo mantener una web, no he cogido el hábito de escribir con regularidad (debería).

Proyectos:

Lista de recursos Python para educación https://github.com/quobit/awesome-python-in-education

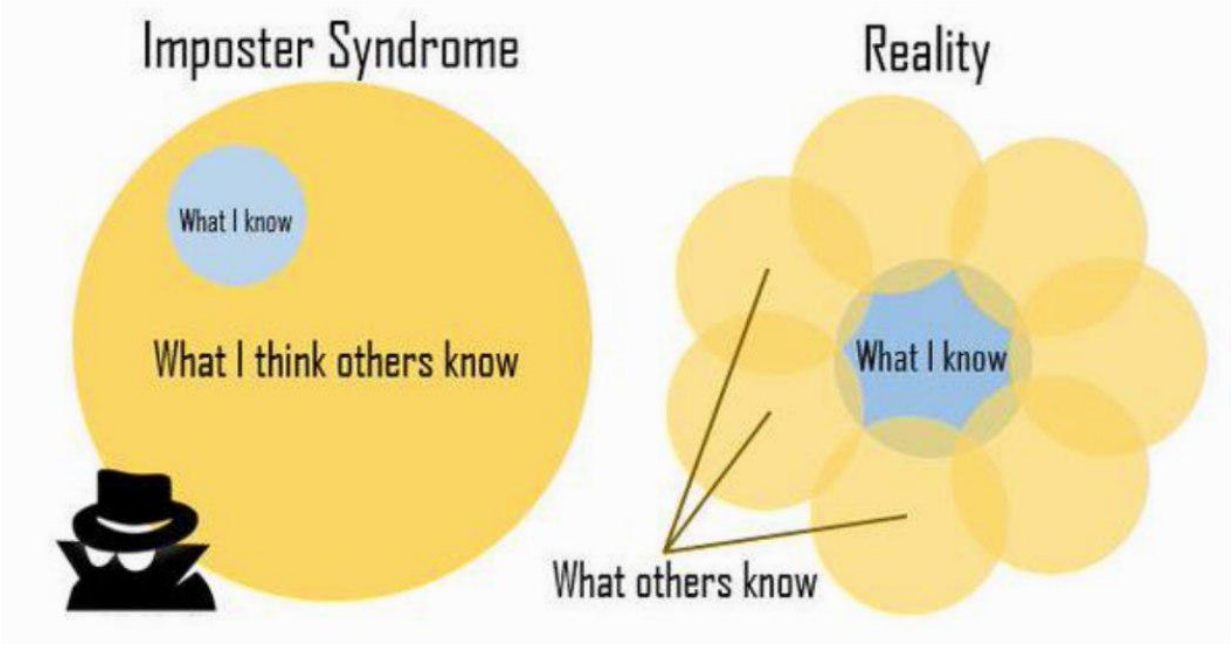
Grupo Telegram para educadores Python de todo tipo: @PythonEsp_edu

Grupo Telegram OpenBSD España: @OpenBSD_es

Canal de recursos para profesores de Informática: @apicv

Persona que propongo, sin haberle preguntado (ojalá acceda □□), con un perfil técnico y humano envidiable: @albertocurro

Lo conozco de los comienzos del grupo @PythonEsp aunque ahora se prodiga más en @DevOpsHispano y @FlaskEsp



EED: Doy por finalizada está entrevista.

La próxima entrevista será el 6 de Mayo, a la espera de confirmación del entrevistado.