Guía didáctica

Metodología de Programación

Título: Grado en Ingeniería Informática

Módulo: Fundamentos de Informática
Créditos: 6
ECTS Código: 08GIIN
Curso: 2018-2019

viu Universidad Internacional de Valencia

Título: Metodología de Programación

Descripción:

En esta asignatura profundizaremos los conocimientos vistos en Fundamentos de programación (o conocimientos básicos previos en programación). El alumno seguirá mejorando sus competencias en la programación mediante la aplicación de las mismas al desarrollo de un pequeño proyecto.
En paralelo se introducirán nuevos conceptos de programación para afrontar cada vez problemas más complejos.

Carácter: Obligatoria

Créditos ECTS: 6

Contextualización:

Lo que distingue un ingeniero informático a la hora de programar es su conocimiento más amplio de una serie de aspectos que le permiten desarrollar programas que además de realizar la tarea esperada lo hacen con eficacia, eficiencia y de manera consistente.
Veremos la relación entre la representación a alto nivel de los diferentes tipos de datos a bajo nivel. El uso de herramientas de depuración para detectar y corregir los errores. La necesidad de realizar pruebas para la validación de los programas. Como contemplar los casos imprevistos que pueden generar errores en tiempo de ejecución, haciendo un buen manejo de errores y capturando las excepciones. Podremos manejar grandes cantidades de información y/o obtener persistencia de la misma usando ficheros para E/S/. Todo ello aplicando metodologías de programación para la resolución de problemas complejos, como son la descomposición de los mismos usando el diseño modular y reutilizando código con las bibliotecas.

Modalidad de impartición*:* Online

Equipo docente:

Profesor: Roger Clotet Martínez

Correo electrónico: roger.clotet@campusviu.es

Temario:

Tipos de datos del lenguaje de alto nivel y su representación interna

Referencias de memoria y memoria dinámica

Encapsulamiento y ocultamiento de la información

Diseño modular y creación de bibliotecas

Herramientas de depuración, pruebas y validación

Gestión de errores

Mantenimiento del software

E/S (Input/Output), ficheros

Competencias*:*

CG.1.- Que los estudiantes hayan desarrollado aquellas habilidades de aprendizaje necesarias para emprender estudios posteriores con un alto grado de autonomía.

CG.2.- Conocimiento de las materias básicas y tecnologías, que capaciten para el aprendizaje y desarrollo de nuevos métodos y tecnologías, así como las que les doten de una gran versatilidad para adaptarse a nuevas situaciones.

CG.3.- Que los estudiantes sepan aplicar sus conocimientos a su trabajo o vocación de una forma profesional y posean las competencias que suelen demostrarse por medio de la elaboración y defensa de argumentos y la resolución de problemas dentro de su área de estudio.

CG.4.- Que los estudiantes puedan transmitir información, ideas, problemas y soluciones a un público tanto especializado como no especializado.

CE.1.- Conocimientos básicos sobre el uso y programación de los ordenadores, sistemas operativos, bases de datos y programas informáticos con aplicación en ingeniería.

CE.2.- Conocimiento de la estructura, organización, funcionamiento e interconexión de los sistemas informáticos, los fundamentos de su programación, y su aplicación para la resolución de problemas propios de la ingeniería.

RESULTADOS DE APRENDIZAJE

Al finalizar esta asignatura se espera que el estudiante sea capaz de describir:

RA.1.□ Explicar la relación de tipos de lenguaje de alto nivel y la representación a bajo nivel de dicha información.

RA.2.□ Distinguir correctamente las referencias y los objetos referenciados.

RA.3.□ Desarrollar nuevos tipos de datos, realizando una correcta separación entre interfaz e implementación.

RA.4.□ Manejar correctamente herramientas de depuración, pruebas y validación.

RA.5. Realizar una correcta gestión de la E/S, especialmente motivada por la necesidad de manejar grandes cantidades de información almacenada en ficheros

RA.6. Usar Python para la programación de ordenadores.

RA.7. Aplicar metodologías de programación para la resolución de problemas.

Actividades Formativas:

Actividad Formativa	Horas	Presencialidad
Clases expositivas	60	60
Resolución de ejercicios prácticos	80	30
Prácticas de laboratorios virtuales	100	20
Tutorías	60	0
Trabajo Autónomo	300	0

Metodologías docentes:

Clases teóricas impartidas como lecciones magistrales o exposiciones, en las que además de presentar el contenido de la asignatura, se explican los conceptos fundamentales y se desarrolla el contenido teórico.
Colección de tareas que el alumnado llevará a cabo a lo largo de toda la asignatura, entre las que podemos encontrar: análisis de casos, resolución de problemas, prácticas de laboratorios, comentarios críticos de textos, análisis de lecturas, etc.
Sesiones periódicas entre el profesorado y el alumnado para la resolución de dudas, orientación, supervisión, etc.
Trabajo tanto individual como grupal para la lectura crítica de la bibliografía, estudio sistemático de los temas, reflexión sobre problemas planteados, resolución de actividades propuestas, búsqueda, análisis y elaboración de información, investigación e indagación, así como trabajo colaborativo basado en principios constructivistas.

Sistema de Evaluación:

Sistemas de evaluación	Ponderación mínima	Ponderación máxima
Entrega de informes de problemas y ejercicios	10	25
Planteamiento, estudio, análisis y resolución de casos	0	10
Informes o memorias de prácticas de laboratorio	0	20
Trabajos o proyectos desarrollados en grupo o de forma individual	0	15
Participación activa en los debates, foros y otros medios	5	10
Evaluación final: se podrán realizar exámenes finales o parciales (que incluyan ítems de alternativas, de asociación, multi-ítems, interpretativos, preguntas de desarrollo breve o extenso), supuestos prácticos y/o análisis de casos, sobre el desarrollo y los resultados de las actividades propuestas	60	60

Bibliografía:

Zuras, D., Cowlishaw, M., Aiken, A., Applegate, M., Bailey, D., Bass, S., ... & Canon, S. (2008). IEEE standard for floating-point arithmetic. IEEE Std 754-2008, 1-70.

Pigoski, T. M. (1996). Practical software maintenance: best practices for managing your software investment. Wiley Publishing.

Eick, S. G., Graves, T. L., Karr, A. F., Marron, J. S., & Mockus, A. (2001). Does code decay? assessing the evidence from change management data. IEEE Transactions on Software Engineering, 27(1), 1-12.

Python Software Foundation. Python 3 documentation. Recuperado el (17-09-2018) de https://docs.python.org/3/

Python Software Foundation. Package Index. Recuperado el (17-09-2018) de https://pypi.org/

Pedro Gomis Román (2017). Manual de la asignatura 04GIIN Fundamentos de Programación. Grado de Ingeniería Informática Universidad Internacional de Valencia.

Python Software Foundation. Scientific PYthon Development EnviRonment (Spyder). Recuperado el (17-09-2018) de https://pypi.python.org/pypi/spyder/

Pressman, R. (2010). Ingeniería del Software-Enfoque Práctico Mc Graw Hill 7ª. Edición.

Lehman, Meir M. (1980). «Programs, Life Cycles, and Laws of Software Evolution». Proc. IEEE 68 (9): 1060-1076.

Lehman, M. M., Ramil, J. F., Wernick, P. D., Perry, D. E., & Turski, W. M. (1997, November). Metrics and laws of software evolution-the nineties view. In Software metrics symposium, 1997. proceedings., fourth international (pp. 20-32). IEEE.

Boehm, B. W. (1988). A spiral model of software development and enhancement. Computer, 21(5), 61-72.

Almeida, M. A. R. (1993). Metodología de la programación: a través de Pseudocódigo. MacGraw-Hill.

Balcázar, J. L. (1993). Programación metódica (pp. I-XVI). McGraw-Hill.

Raúl González Duque. Tutorial de Python 'Python para todos'. Recuperado el (17-09-2018) de http://mundogeek.net/tutorial-python/