Ficha da Unidade

Mestrado Integrado em Engenharia Informática e Computação
Fundamentos da Programação
Ocorrência: 2018/2019
—

Institutional page

Informação geral

Unidade: Fundamentos da Programação
Código: EIC0005
Sigla: FPRO
Curso: MIEIC
Anos Curriculares: 1º
Página oficial: Sigarra
Créditos ECTS: 6
Horas de Contacto: 70
Horas Totais: 162
Ocorrência: 2018/2019
Semestre: 1S
Teóricas: 2x1,5h: João Correia Lopes (JCL)
Teórico-Práticas: 7x2h: João Correia Lopes (JCL) & Rui Camacho (RCS)
Atendimento: 1x1h: João Damas

Língua de trabalho

Português.

Objetivos

1. INTRODUÇÃO

A fluência no processo de desenvolvimento de software é um pré-requisito básico para o trabalho dos engenheiros de informática. Para usar computadores na resolução de problemas de maneira eficaz, os estudantes devem ser competentes na leitura e escrita de programas usando linguagens de programação de Alto Nível.

2. OBJETIVOS ESPECÍFICOS

O objetivo global desta Unidade é dar ao aluno a capacidade de criar algoritmos e de usar uma linguagem de programação para implementar, testar e depurar algoritmos para resolver problemas simples.

O aluno será capaz de entender e usar os conceitos fundamentais de programação e a abordagem Funcional da programação, especificamente a programação livre de efeitos (*effect-free*), onde as chamadas de função não têm efeitos colaterais e as variáveis são imutáveis, e de contrastar esta abordagem com a abordagem Imperativa.

3. DISTRIBUIÇÃO PERCENTUAL

Componente científica: 40%
Componente tecnológica: 60%

Resultados de aprendizagem e competências

No final do curso, espera-se que o estudante consiga resolver problemas através do desenvolvimento de programas de complexidade média, usando as abordagens ou paradigmas de programação imperativa e funcional.

Mais especificamente, o estudante deverá ser capaz de::

1. Projetar, implementar, testar e depurar um programa que use as construções fundamentais de programação, cálculos básicos, estruturas condicionais e iterativas padrão, Entrada/Saída simples, persistência (ficheiros) e exceções.
2. Entender a Abstração de dados e usar tipos de dados simples ou agregados.
3. Entender a Abstração procedimental e usar a definição de funções, passagem de parâmetros, recursão.
4. Escrever funções úteis que aceitem e devolvam outras funções.
5. Implementar algoritmos básicos que evitem atribuir a um estado mutável ou considerar igualdade de referência.
6. Entender variáveis e escopo léxico num programa (*closures*).
7. Definir e usar iteradores e outras operações em agregações, incluindo operações que tomam funções como argumentos (*map*, *reduce/fold*, e *filter*).
8. Usar as ferramentas de programação que ajudam a escrever, testar e documentar programas em computador de acordo com as melhores práticas de programação.

Modo de trabalho

Presencial.

Pré-requisitos (conhecimentos prévios) e co-requisitos (conhecimentos simultâneos)

Não são necessários conhecimentos prévios.

Programa

- Introdução à resolução de problemas, a algoritmos e à programação com Python.
- Conceitos fundamentais de programação: tipos de dados simples; variáveis, expressões e declarações; fluxo de programa, condicionais, iteração; persistência.
- Tipos de dados agregados: strings, tuplos, listas, dicionários.
- Funções, passagem de parâmetros, recursão.
- Funções como cidadãos de primeira classe e *closures*.
- Funções de ordem superior úteis: *map*, *reduce/fold*, e *filter*.
- Programação livre de efeitos usando chamadas de função sem efeitos colaterais e variáveis imutáveis
- Ferramentas de programação, depuração, exceções e estratégias de resolução de problemas.

Bibliografia Obrigatória

- Peter Wentworth, Jeffrey Elkner, Allen B. Downey, and Chris Meyers, *How to Think Like a Computer Scientist — Learning with Python 3*, 2012 [PDF 2018] [HTML] [Biblioteca]

Bibliografia Complementar

- Brad Miller and David Ranum, Learning with Python: Interactive Edition. Based on material by Jeffrey Elkner, Allen B. Downey, and Chris Meyers HTML
- Steven F. Lott, Building Skills in Python - A Programmer's Introduction to Python, FreeTechBooks, 2010 HTML
- Allen Downey, Think Python — How to Think Like a Computer Scientist, 2nd Edition, Version 2.2.23, Green Tea Press, 2015 HTML PDF code
- David Mertz, Functional Programming in Python, O'Reilly Media, 2015 PDF
- Guttag, John. Introduction to Computation and Programming Using Python: With Application to Understanding Data, Second Edition. MIT Press, 2016. ISBN: 9780262529624.
- João Pavão Martins, Programação em Python - Introdução à Programação Utilizando Múltiplos Paradigmas, Instituto Superior Técnico, 2ª edição, 2017, ISBN: 9789898481474

Métodos de ensino e atividades de aprendizagem

O envolvimento contínuo do estudante com a Unidade é promovido, tanto com o estudo e discussão dos tópicos em aulas teóricas e teórico-práticas como através de trabalhos de programação dentro e fora das aulas.

São usadas ferramentas de correção automática de código para aumentar a eficiência do feedback dado aos estudantes.

O estudante é encorajado a encontrar as melhores ideias para resolver problemas específicos, executá-las e implementar as soluções de programação, de forma elegante, legível e eficiente (em tempo e em espaço).

A linguagem de programação Python é usada na implementação.

As aulas teóricas são usadas para apresentar e discutir os tópicos do programa, usando um computador ligado a um projetor multimédia.

As aulas teórico-práticas são usadas para ajudar os estudantes a entender os tópicos do programa e a resolver as tarefas de programação semanais.

As tarefas de programação em aula teórica e fora das aulas são dadas, semanalmente, para melhorar o desenvolvimento regular e eficaz dos processos de aprendizagem autónomos.

Tarefas em sala de aula são usadas para avaliação somativa no final de cada aula teórica (questionários Moodle).

Além disso, uma plataforma baseada em jogos (Kahoot) é usada para fazer uma avaliação formativa, a fim de avaliar o quão eficazes são as aulas teóricas e ajudar a ajustar o ritmo e a ênfase nos tópicos de aula.

Software

- Python 3.6 (https://www.python.org/)
- Spyder3 (https://docs.spyder-ide.org/)

Palavras-chave

Ciências Físicas > Ciência de computadores > Programação

Tipo de avaliação

Avaliação distribuída sem exame final.

Avaliação e componentes de Ocupação

Descrição	Tipo	Tempo (horas)	Data de Conclusão
Participação presencial (estimativa)	Aulas	70	
PE1	Teste/Exame	0	26/10/2018
PE2	Teste/Exame	0	23/11/2018
PE3	Teste/Exame	0	21/12/2018
PE	Teste/Exame	0	08/01/2019
TE	Teste/Exame	0	11/01/2019
Estudo	Estudo	92	
	Total:	162	

Obtenção de frequência

Os estudantes são admitidos ao exame teórico (TE), se não excederem o limite de faltas (25% do número total de aulas teórico-práticas estimadas) e se obtiverem um mínimo de 40% na componente de avaliação associada à avaliação prática em computador (PE) .

Fórmula de cálculo da classificação final

A avaliação será baseada nas seguintes componentes::

LE = Exercícios em sala de aula teórica
RE = Exercícios de programação semanais para casa e aulas teórico-práticas
PEn = Avaliação prática em computador (n-ésimo)
PE = (PE1 + PE2 + PE3) / 3
TE = Avaliação teórica com consulta de um livro

Classificação final = 10% LE + 10% RE + 40% PE + 40% TE

Observações:

1. É exigida classificação mínima de 40% na componente TE.
2. Terá lugar um outro TE para estudantes admitidos a exame e que faltem ao exame com justificação válida, ou de recuperação para os estudantes que não obtiveram a nota mínima. No último caso, a nota é limitada a 50%.
3. Se a classificação final for maior que 17, o estudante pode ser submetido a um exame oral e a nota final é a média das duas.
4. Os estudantes que atingirem pelo menos 85% nas tarefas de programação presencial e semanal (LE + RE), a meio da Unidade podem optar por manter a nota e fazer um pequeno projeto para atingir a nota máxima (100%).

Provas e trabalhos especiais

A avaliação deste curso usa três tipos de provas::

1. Exercícios em aula teórica e fora de aula relacionadas com os tópicos atuais, a serem submetidos no Moodle e classificados automaticamente.
2. Avaliação prática em computador de um conjunto de pequenos programas de computador utilizando o Moodle e classificados automaticamente, e com consulta de textos de referência.
3. Avaliação teórica através de um questionário de múltipla escolha no Moodle, com consulta de um livro, para avaliar a capacidade dos estudantes de usar e discutir os melhores algoritmos e estruturas de dados para problemas específicos.

Avaliação especial (TE, DA, ...)

A classificação final será a média da classificação de duas componentes de avaliação: Avaliação prática em computacional (PE) e Avaliação teórica (TE).

Classificação final = 50% PE + 50% TE

Observações:

1. É exigida classificação mínima de 40% em cada componente (PE e TE)
2. Se, por decisão do corpo docente, um dos componentes de avaliação não tiver lugar, será substituído por um exame oral.
3. Se a classificação final for maior que 17, o estudante pode ser submetido a um exame oral e a nota final é a média das duas.

Melhoria de classificação

O estudante pode solicitar um exame único, com consulta de apenas um livro, com nota numa escala de 0 a 20.

Se a classificação final for maior que 17, o estudante pode ser submetido a um exame oral e a nota final é a média das duas.

Observações

Nenhuma classificação de componente individual (LE[1)], RE, PE, TE[2)]) pode ser reutilizada noutra inscrição no curso.

— FPRO, 2018/19

1)
LE = Ler e entender + ver bons exemplos de programação
2)
REPETE = Experimentar com os programas e exercícios indicados

From:
https://web.fe.up.pt/~jlopes/ - **JCL**

Permanent link:
https://web.fe.up.pt/~jlopes/doku.php/teach/fpro/201819/ficha

Last update: **26/07/2019 09:45**