English version at the end of this document

Ano Letivo	2019-20
Unidade Curricular	COMPLEMENTOS DE PROGRAMAÇÃO
Cursos	SISTEMAS E TECNOLOGIAS DE INFORMAÇÃO
Unidade Orgânica	Instituto Superior de Engenharia
Código da Unidade Curricular	18121006
Área Científica	CIÊNCIAS INFORMÁTICAS,FORMAÇÃO TÉCNICA
Sigla	FT
Línguas de Aprendizagem	Português
Modalidade de ensino	Presencial
Docente Responsável	Pedro Jorge Sequeira Cardoso

DOCENTE	TIPO DE AULA	TURMAS	TOTAL HORAS DE CONTACTO (*)
Pedro Jorge Sequeira Cardoso	PL; TP	TP1; PL1	15TP; 45PL

* Para turmas lecionadas conjuntamente, apenas é contabilizada a carga horária de uma delas.

ANO	PERÍODO DE FUNCIONAMENTO*	HORAS DE CONTACTO	HORAS TOTAIS DE TRABALHO	ECTS
1º	S2	15TP; 45PL	140	5

* A-Anual;S-Semestral;Q-Quadrimestral;T-Trimestral

Precedências

Sem precedências

Conhecimentos Prévios recomendados

Conceitos de programação procedimental. Frequência da UC de Programação.

Objetivos de aprendizagem (conhecimentos, aptidões e competências)

Proporcionar uma visão geral sobre os conceitos definidos em programação orientada a objetos (POO). Capacitar o aluno a oferecer soluções algorítmicas para problemas computacionais.

Utilizar uma linguagem de POO para implementação das soluções.

Conteúdos programáticos

Introdução ao paradigma da Programação Orientada a Objetos
1. Perspetiva histórica
2. Características da programação orientada a objetos (encapsulamento, partilha de comportamento, evolução)
3. Objetos (noção de Objeto e identidade; protocolo de utilização de um Objeto; mensagens e métodos; noção de estado e comportamento.
4. Linguagens baseadas em classes; instanciação de Objetos.
5. Hierarquias de Objetos: Mecanismos de herança; Reutilização de código; Sistemas reflexivos; Polimorfismo.
6. Interfaces: Separação entre interface e implementação de interfaces.
7. Relações entre Objetos: Extensão, generalização e especialização; Associação, agregação e composição; Coesão e acoplamento.
8. Aplicações (GUI, Ligações a Bases de Dados, Desenvolvimento de aplicações móveis)

Demonstração da coerência dos conteúdos programáticos com os objetivos de aprendizagem da unidade curricular

O encadeamento e a sequência dos conteúdos programáticos permitirão desenvolver uma aprendizagem gradual, geradora das seguintes competências:

1 - Capacidade de resolver problemas recorrendo as técnicas de POO;

2 - Capacidade de saber codificar e implementar usando POO

3 - Capacidade de aplicar os conceitos ao desenvolvimento de aplicações, nomeadamente com interfaces gráficas e coneções a bases de dados.

Metodologias de ensino (avaliação incluída)

Metodologias de ensino

Aulas Teórico-Práticas: Método expositivo, orientado de acordo com um ensino baseado em problemas, com projeção e explicação dos objetivos e conteúdos correspondentes a cada semana.

Aulas Práticas: Prática laboratorial em computador. Resolução e codificação de problemas tipificados, selecionados em conformidade com o conteúdo teórico semanal.

Avaliação

Componentes da avaliação classificadas de 0-20 valores:

- Trabalho(s) prático(s) (TP)
- Prova escrita (PE): Teste / Exame

Classificação mínima em cada uma das componentes 7 valores.

Nota final: 0.5 PE + 0.5 TP

De acordo com o n.º 3 do artigo 6.º do despacho reitoral RT 59/2015, de 28 de julho, nos cursos técnicos superiores profissionais, a inclusão do cumprimento do dever de assiduidade nos métodos de avaliação é obrigatória, nos seguintes termos:
a) Considera-se que um estudante cumpre o dever de assiduidade a uma UC, quando não exceda o número limite de faltas correspondente a 25% das horas de contacto previstas

Demonstração da coerência das metodologias de ensino com os objetivos de aprendizagem da unidade curricular

O objetivo desta UC é dar ao aluno um conjunto de conhecimentos em Programação Orientada aos Objetos. Estes conhecimentos não se cingirão à sintaxe de uma linguagem de programação particular, mas incluirão aspetos relativos à modelação orientada aos objetos.

A representação dos conceitos associados à modelação OOP será feita utilizando um conjunto estrito de diagramas UML (*Unified Modeling Language*), sempre que se justifique. O objetivo não é ensinar exaustivamente UML, mas possibilitar a visualização dos conceitos associados à POO de uma forma simples (e.g. classes, relações entre classes, herança, etc.).

Bibliografia principal

- Borges, L (2010). Python para desenvolvedores. Edição do autor
- Goldwasser, M. H., & Letscher, D. (2008). *Object-oriented programming in Python* . Pearson Prentice Hall.
- Lukaszewski A. (2010). MySQL for Python. Packt Publishing.
- Lutz, M. (2013). *Learning python: Powerful object-oriented programming* . " O'Reilly Media, Inc.".
- Reges, S., Stepp, M., & Obourn, A. (2018). *Building Python Programs* . Pearson.
- Romano, F. (2015). Learning Python. Packt Publishing.
- Summerfield, M. (2008), Programming in Python 3: A Complete Introduction to the Python Language. Addison-Wesley Professional.

Academic Year	2019-20
Course unit	PROGRAMMING II
Courses	SISTEMAS E TECNOLOGIAS DE INFORMAÇÃO
Faculty / School	INSTITUTE OF ENGINEERING
Main Scientific Area	CIÊNCIAS INFORMÁTICAS,FORMAÇÃO TÉCNICA
Acronym	FT
Language of instruction	Portuguese
Teaching/Learning modality	Presential
Coordinating teacher	Pedro Jorge Sequeira Cardoso

Teaching staff	Type	Classes	Hours (*)
Pedro Jorge Sequeira Cardoso	PL; TP	TP1; PL1	15TP; 45PL

* For classes taught jointly, it is only accounted the workload of one.

Contact hours

T	TP	PL	TC	S	E	OT	O	Total
0	15	45	0	0	0	0	0	140

T - Theoretical; TP - Theoretical and practical ; PL - Practical and laboratorial; TC - Field Work; S - Seminar; E - Training; OT - Tutorial; O - Other

Pre-requisites

no pre-requisites

Prior knowledge and skills

Concepts of procedural programming. Frequency of Programming CU.

The students intended learning outcomes (knowledge, skills and competences)

Provide an overview of concepts defined in object-oriented programming (OOP). Enable student to offer algorithmic solutions to computational problems.

Use an OOP language to implement programming solutions.

Syllabus

Introduction to the Object Oriented Programming Paradigm
1. Historical perspective
2. Characteristics of object-oriented programming (encapsulation, behavior sharing, evolution)
3. Objects (notion of object and identity, protocols of object's usage, messages and methods, notion of state and behavior).
4. Class-based languages; instantiation of objects.
5. Object Hierarchies: Inheritance mechanisms; Reuse of code; Reflection; Polymorphism.
6. Interfaces.
7. Relations between Objects: Extension, generalization and specialization; Association, aggregation and composition; Cohesion and coupling.
8. Applications (GUI, database connections)

Demonstration of the syllabus coherence with the curricular unit's learning objectives

The sequencing of the syllabus contents will allow for a gradual learning process, generating the following competences:

1 - Ability to solve problems using techniques of OOP;

2 - Ability to code and implement using OOP

3 - Ability to apply the concepts to the development of applications, namely with graphical interfaces and database connections.

Teaching methodologies (including evaluation)

Teaching methodologies

Theoretical-Practical classes: Expositive method, oriented according to a problem-based teaching, with projection and explanation of the objectives and contents corresponding to each week.

Practical classes: Laboratory practice in computer. Resolution and codification of typified problems, selected according to the theoretical weekly content.

Evaluation

Valuation components classified as 0-20 values:

- Practical assignments (TP)
- Written exam (PE): Test / Exam

Minimum classification in each of the components: 7 values.

Final grade: 0.5 PE + 0.5 TP

Pursuant to no. 3 of Article no. 6 of RT 59/2015, of July 28, in the professional higher technical courses, the inclusion of the fulfillment of the duty of assiduity in the methods of evaluation is obligatory, in the following terms:
a) It is considered that a student fulfills the duty of assiduity to a CU, when it does not exceed the limit of absences corresponding to 25% of the foreseen contact hours

Demonstration of the coherence between the teaching methodologies and the learning outcomes

The purpose of this UC is to give the student a set of knowledge in Object Oriented Programming. This knowledge will not be bound to the syntax of a particular programming language, but will include aspects related to object-oriented modeling.

The representation of the concepts associated with OOP modeling will be done using a strict set of Unified Modeling Language (UML) diagrams whenever possible. The goal is not to teach UML comprehensively, but to enable the visualization of concepts associated with OOP in a simple way (eg classes, relationships between classes, inheritance, etc.).

Main Bibliography

- Borges, L (2010). Python para desenvolvedores. Edição do autor
- Goldwasser, M. H., & Letscher, D. (2008). *Object-oriented programming in Python* . Pearson Prentice Hall.
- Lukaszewski A. (2010). MySQL for Python. Packt Publishing.
- Lutz, M. (2013). *Learning python: Powerful object-oriented programming* . " O'Reilly Media, Inc.".
- Reges, S., Stepp, M., & Obourn, A. (2018). *Building Python Programs* . Pearson.
- Romano, F. (2015). Learning Python. Packt Publishing.
- Summerfield, M. (2008), Programming in Python 3: A Complete Introduction to the Python Language. Addison-Wesley Professional.