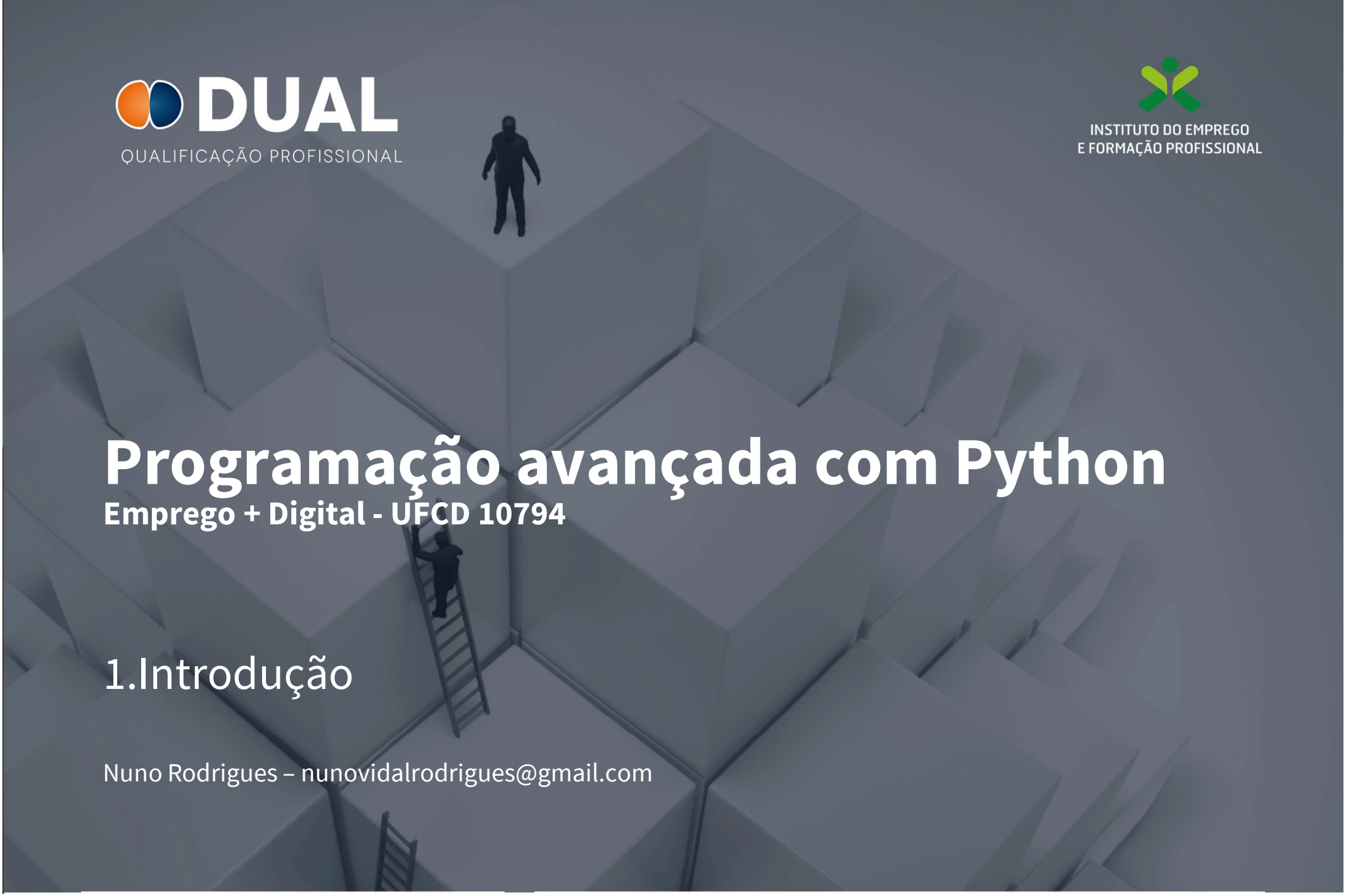


DUAL
QUALIFICAÇÃO PROFISSIONAL

INSTITUTO DO EMPREGO
E FORMAÇÃO PROFISSIONAL

Programação avançada com Python

Emprego + Digital - UFCD 10794

1.Introdução

Nuno Rodrigues – nunovidalrodrigues@gmail.com

UM SERVIÇO
AHK Deutsch-Portugiesische Industrie- und Handelskammer
Câmara de Comércio e Indústria Luso-Alemã

ENTIDADE FORMADORA CERTIFICADA

PRR Plano de Recuperação e Resiliência

REPÚBLICA PORTUGUESA

Financiado pela União Europeia
NextGenerationEU

40 ANOS 1983 – 2023
HISTÓRIAS DE QUALIFICAÇÃO

DUAL
QUALIFICAÇÃO PROFISSIONAL

Objetivos da ação de formação

- ✓ Aplicar as boas práticas de escrita de código.
- ✓ Criar classes e utilizar objetos de forma efetiva.
- ✓ Utilizar as propriedades dos objetos para criar “código dinâmico”.
- ✓ Efetuar a depuração e log.
- ✓ Programar para a web em Python.

DUAL
QUALIFICAÇÃO PROFISSIONAL

Avaliação da ação de formação

- ✓ Avaliação contínua de participação em aula e realização de 1 teste escrito e/ou prático intermédio e 1 teste escrito e/ou prático final.
- ✓ Para a conclusão com sucesso, é necessário um resultado positivo nos momentos de avaliação escrita e/ ou prática e uma assiduidade mínima de 90% da duração total do curso.

DUAL
QUALIFICAÇÃO PROFISSIONAL

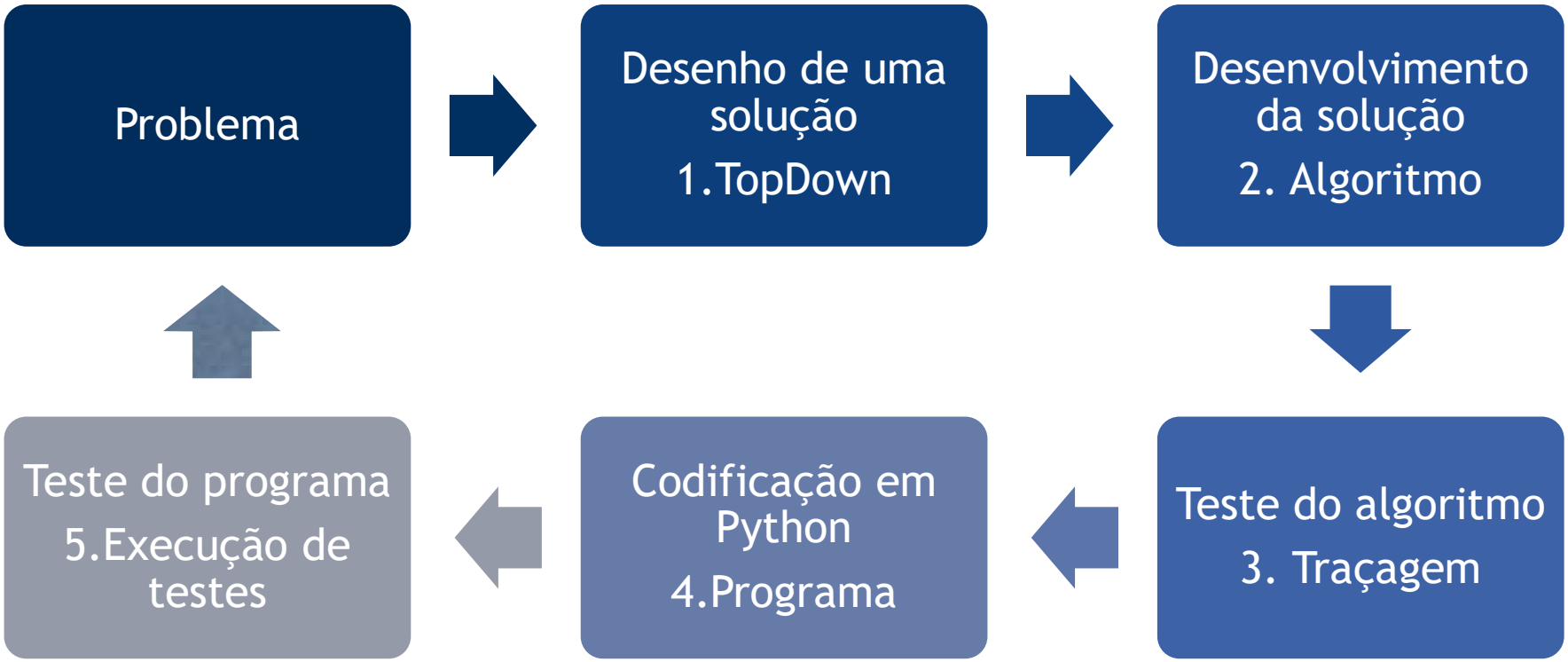
Introdução à linguagem Python

A linguagem de programação Python foi desenvolvida por Guido van Rossum e lançada em 1991. Atualmente é uma linguagem em código-fonte aberto (*open source*) e supervisionada pela Python Software Foundation. Trata-se de uma linguagem de programação imperativa, de alto nível e interpretada que serve, adequadamente para realizar programação estruturada e programação orientada por objetos. O fato de se ter tornado amplamente utilizada, deve-se precisamente ao fato de poder ser utilizada enquanto linguagem de programação nestes dois tipos de programação.

Na linguagem Python não é apenas possível a implementação dos princípios e das figuras algorítmicas e das estruturas de dados principais, mas também a indentação dos blocos de instruções, simplificando a sua sintaxe e tornando os programas mais legíveis. Em Python a indentação é obrigatória para delimitar os blocos de instrução e impor a ordem de execução dos programas.

DUAL
QUALIFICAÇÃO PROFISSIONAL

Introdução à linguagem Python



DUAL
QUALIFICAÇÃO PROFISSIONAL

Ambientes de desenvolvimento

Existem diferentes ambientes de desenvolvimento para a linguagem Python. Nesta ação de formação serão focados dois IDE's (Integrated Development Environment) Spyder e Visual Studio Code.

VS Code

https://learn.microsoft.com/pt-br/training/modules/python-install-vscode/1-introduction

https://code.visualstudio.com/docs/python/python-tutorial

Spyder

https://www.spyder-ide.org/

DUAL
QUALIFICAÇÃO PROFISSIONAL

Tratamento idiomático de dados

Comparação de operadores em cadeia

Existem dois grandes tipos de operadores em *Python* os operadores Booleanos e os operadores de comparação.

No que concerne aos operadores Booleanos, estes apenas operam em **True** e **False** (Verdadeiro e Falso), e respeitam essencialmente estados lógicos. Existem três operações com objetos do tipo Booleano.

Operadores	Descrição
X *and* Y	O **e** lógico
X *or* Y	O **ou** lógico
not X	A negação lógica

DUAL
QUALIFICAÇÃO PROFISSIONAL

Tratamento idiomático de dados

Comparação de operadores em cadeia

No que concerne aos operadores de comparação , estes apenas operam com base em comparações matemáticas onde permitem realizar verificações como indica a tabela seguinte.

Operadores	Descrição
>	Maior
>=	Maior ou igual
<	Menor
<=	Menor ou igual
==	Igual valor
!=	Diferente(valor)
is	Igual identidade
Is not	Diferente identidade

DUAL
QUALIFICAÇÃO PROFISSIONAL

Tratamento idiomático de dados

Comparação de operadores em cadeia

Exemplo:

```
>>>True and False
False
>>>True and True
True
>>>True or False
True
>>> False or False
False
>>> not True
False
```

DUAL
QUALIFICAÇÃO PROFISSIONAL

Tratamento idiomático de dados

Comparação de operadores em cadeia

Exemplo:

```
>>>4>=5
False
>>>3<=4
True
>>>4 !=5
True
>>> 3==3
True
>>> not True
False
```

DUAL
QUALIFICAÇÃO PROFISSIONAL

Tratamento idiomático de dados

Identação

Python

```
if __name__=='__main__':
    print("Quantas moedas tem o Bernardo?\n")
    bernardo=int(input())
    print("Quantas moedas ofereceu o Ricardo?\n")
    ricardo=int(input())
    moedas=bernardo+ricardo
    print("O Bernardo fica com", moedas, "moedas.")
```

C/C++

```
#include <iostream>

using namespace std;
int main() {
cout << "Quantas moedas tem o Bernardo?\n";
int bernardo;
cin >> bernardo;
cout << "Quantas moedas ofereceu o Ricardo?\n";
int ricardo;
cin >> ricardo;
int moedas = bernardo + ricardo;    cout << "O Bernardo fica com " << moedas << " moedas.";
return 0;
}
```

DUAL
QUALIFICAÇÃO PROFISSIONAL

Tratamento idiomático de dados

Falsy Truthy

Expressões com operadores permitem validar se uma situação é verdadeira ou falsa e podem ser facilmente aplicadas em estruturas de decisão do tipo if ou ciclos de repetição como condição de execução.

```
# Expressão a verificar 5 < 3
>>> if 5 < 3:
        print("True")
    else:
        print("False")
# Saída
False
```

DUAL
QUALIFICAÇÃO PROFISSIONAL

Tratamento idiomático de dados

Falsy Truthy

No exemplo anterior, tudo ocorre da forma pretendida uma vez que efetivamente 5<3 é falso. Mas se retirarmos a expressão e passarmos para algo como:

```
>>> a = 5
>>> if a:
        print(a)
```

Verifica-se que a resposta é 5 como expectável mas se **a** assume o valor de zero não temos saída.

```
>>> a = 0
>>> if a:
        print(a)
```

DUAL
QUALIFICAÇÃO PROFISSIONAL

Tratamento idiomático de dados

Falsy Truthy

Uma vez que a variável **a** não um expressão típica, não possui operadores e por este motivo não valida como verdadeiro ou falso dependendo do valor?

É nesta situação que entra o conceito Falsy and Truthy para os valores que não sendo valores verdeiros por si mesmos podem ser validados como True ou False.

Em Python, cada valor individualmente pode ser validado de True ou False embora não tenham de necessariamente pertencer a uma expressão para validar a sua veracidade ou falsidade.

DUAL
QUALIFICAÇÃO PROFISSIONAL

Tratamento idiomático de dados

Falsy Truthy

As regras aplicadas em Python para estas situações são:

- ✓ Valores validados como False são considerados Falsy.
- ✓ Valores validados como True são considerados Truthy.

Segundo a documentação de Python qualquer objeto pode ser testado quanto ao valor de verdade, para uso em uma condição if ou while ou como operando das operações booleanas (and, or, not).

DUAL
QUALIFICAÇÃO PROFISSIONAL

Tratamento idiomático de dados

Falsy Truthy

Existem diferentes utilidades para escrever código conciso e organizado com estes elementos, embora, exista o risco de obter dados errados caso não sejam seguidas as seguintes regras.

Valores Falsy:

- ✓ Um número zero (int, real ou complexo).
- ✓ Uma string, lista, tupla, dicionário ou conjunto de comprimento zero.
- ✓ O objeto Falso.

DUAL
QUALIFICAÇÃO PROFISSIONAL

Tratamento idiomático de dados

Falsy Truthy

Valores verdadeiros:

- ✓ Um número diferente de zero (int, real ou complexo).
- ✓ Uma string, lista, tupla, dicionário ou conjunto de comprimento diferente de zero.
- ✓ Quase todos os outros objetos que não estão na lista falsa.

DUAL
QUALIFICAÇÃO PROFISSIONAL

Tratamento idiomático de dados

Formatação de strings

A manipulação correta de strings e a sua formatação é essencial em Python. Existem diferentes métodos para a manipulação de strings e a sua aplicação nos diferentes programas que realizamos.

Marcador de Posição

Embora seja uma forma antiga de manipulação de strings continua a ser bastante útil para a realização de diferentes apresentações de dados. Como marcador, utilizamos %s para colocar uma string no interior de outra string.

DUAL
QUALIFICAÇÃO PROFISSIONAL

Tratamento idiomático de dados

Formatação de strings

Marcador de Posição

Exemplo:

```
print('Nesta string será colocada outra %s' % 'string')
```

Após execução obtemos

```
Nesta string será colocada outra string
```

DUAL
QUALIFICAÇÃO PROFISSIONAL

Tratamento idiomático de dados

Formatação de strings

Marcador de Posição

Em alguns casos é necessário passar uma string que possui significado para o interpretador da linguagem, no entanto pretendemos que o interpretador trate como uma string normal.

A estes casos chamamos *raw strings*, literalmente strings “cruas”.

Por exemplo, \t em Python é a representação de espaço da tecla TAB.

Caso queiramos que seja exibido literalmente \t ao invés do espaço, indicamos com %r, onde o r é para indicar raw string:

DUAL
QUALIFICAÇÃO PROFISSIONAL

Tratamento idiomático de dados

Formatação de strings

Marcador de Posição

Exemplo:

```
print('Segue o espaço do TAB: \t. Agora a representação crua: %r.' % '\t’)
```

Após execução obtemos

```
Segue o espaço do TAB:        . Agora a representação crua: '\t'.
```

DUAL
QUALIFICAÇÃO PROFISSIONAL

Tratamento idiomático de dados

Formatação de strings

Utilização com números

Sempre que se pretende uma representação fiel do número anteriormente utilizado ou obtido, utiliza-se %s para que o mesmo seja convertido para string. Podem ser realizadas manipulações, como truncaturas para inteiro e mostrar o sinal.

DUAL
QUALIFICAÇÃO PROFISSIONAL

Tratamento idiomático de dados

Formatação de strings

Utilização com números

Exemplo:

```
num = 13.744
print('Números como %s são convertidos para string' % num)
print('Também podem ser representados como inteiros: %d; repare que sem arredondamento' % num)
print('Mostrando o sinal em inteiros: %+d' % num)
print('Mostrando o sinal em floats: %+f' % num)
```

Após execução obtemos

```
Números como 13.744 são convertidos para string
Também podem ser representados como inteiros: 13; repare que sem arredondamento
Mostrando o sinal em inteiros: +13
Mostrando o sinal em floats: +13.744000
```

DUAL
QUALIFICAÇÃO PROFISSIONAL

Tratamento idiomático de dados

Funções Lambda

As funções lambda são conhecidas também por funções anónimas e tratam-se de funções que o utilizador não precisa de definir, ou seja, não precisa de escrever a função e posteriormente invocar a mesma dentro do código.

A sintaxe da função lambda em Python pode variar conforme cada caso na sua utilização e aplicação ao longo do programa.

A grande vantagem de utilizar lambda é de criar um parâmetro para outra função.

Uma função lambda é definida através da seguinte expressão

{função anónima}::=lambda{parâmetros formais}:{expressão}

Tratamento idiomático de dados

Funções Lambda

Uma função lambda é definida através da seguinte expressão:

{função anónima}::=lambda{parâmetros formais}:{expressão}

A **{expressão}** corresponde a uma expressão em Python.

Ao encontrar uma **{função anónima}**, o Python cria uma função cujos parâmetros formais correspondem a **{parâmetros formais}** e cujo corpo corresponde a **{expressão}**. Quando esta função anónima é executada, os parâmetros concretos são associados aos parâmetros formais, e o valor da função é o valor da {expressão}. Note-se que numa função anónima não existe uma instrução **return** (estando esta implicitamente associada a {expressão }, cujo valor é devolvido).

DUAL
QUALIFICAÇÃO PROFISSIONAL

Tratamento idiomático de dados

Funções Lambda

Exemplos:

Adicionar 10 ao argumento a e obter o resultado

```
x=lambda a : a+10
print(x(5))
```

Multiplicar o argumento a com o argumento b e obter o resultado

```
x=lambda a,b : a*b
print(x(5,6))
```

Tratamento idiomático de dados

Funções Lambda

Qual a utilidade das funções Lambda?

Para melhor explicar esta questão, devemos analisar a utilização das funções definidas.

```
def dobro(n):
        return n*2
print(dobro(10))
```

Na situação anterior foi criada a função dobro e as tarefas que a mesma realiza no seu interior. Utilizando uma função lambda, poderíamos escrever a mesma execução da seguinte forma

```
dobro = lambda n : n*2
print(dobro(10))
```

DUAL
QUALIFICAÇÃO PROFISSIONAL

Tratamento idiomático de dados

Funções Lambda

Qual a utilidade das funções Lambda?

A utilização das funções lambda ajuda a tornar o código mais simplificado e com uma aplicação mais simplista.

Exemplo

```
def minhafunc(n):
        return lambda a : a*n
dobro=minhafunc(2)
print(dobro(11))
```

DUAL
QUALIFICAÇÃO PROFISSIONAL

Tratamento idiomático de dados

Funções Lambda

Qual a utilidade das funções Lambda?

Com o exemplo anterior, é possível verificar que caso se pretenda alterar para o calculo do triplo de um número ou outro valor qualquer, as alterações necessárias são mínimas mantendo a mesma estrutura.

Exemplo

```
def minhafunc(n):
        return lambda a : a*n
triplo=minhafunc(3)
print(triplo(11))
```

DUAL
QUALIFICAÇÃO PROFISSIONAL

Tratamento idiomático de dados

Funções Lambda

Qual a utilidade das funções Lambda?

É possível também introduzir estruturas de decisão anexas às funções lambda.

```
maior=lambda a,b: a if a>b else b
print(maior(10,45))
```

DUAL
QUALIFICAÇÃO PROFISSIONAL

Tratamento idiomático de dados

Funções Lambda

Utilização de funções lambda com listas

A função Filter() é uma função embutida na libraria Python que retorna só os valores que se encontram num determinado critério. A sintaxe desta função é filter(função, iterador). O iterador pode ser qualquer sequencia como uma lista, uma serie de objetos ou outro elemento sequenciado.

Exemplo: Pretendemos obter os elementos pares de uma lista de valores.

```
lista_valores = [1,2,3,4,5,6,7,8,9]
list(filter(lambda x: x%2==0, lista_valores))
[2, 4, 6, 8]
```

DUAL
QUALIFICAÇÃO PROFISSIONAL

Tratamento idiomático de dados

Funções Lambda

Utilização de funções lambda com listas

A função map() é outra função incluída na libraria Python que permite modificar uma lista de valores quando cada valor na lista original foi alterado com base na função.

Exemplo: Pretende-se elevar ao cubo os valores contidos numa lista utilizando a função map()

```
lista_valores = [1,2,3,4,5,6,7,8,9]
cubo = map(lambda x: pow(x,3), lista_valores)
list(cubo)
[1, 8, 27, 64, 125, 216, 343, 512, 729]
```

DUAL
QUALIFICAÇÃO PROFISSIONAL

INSTITUTO DO EMPREGO
E FORMAÇÃO PROFISSIONAL

Centro DUAL Lisboa
Avenida Infante D. Henrique,
Lote 320, Entreposto 2,
Piso 2, Fração 2
1800-220 Lisboa
Tel. +351 213 474 415
duallisboa@dual.pt

Centro DUAL Porto
Avenida Sidónio Pais, 379
4100-468 Porto
Tel. +351 226 061 561
dualporto@dual.pt

Centro DUAL Portimão
Rua Jaime Palhinha
Edifício Portimar, 2º Andar
8500-833 Portimão
Tel. +351 282 484 703
dualportimao@dual.pt

- **Qualificação Inicial**
- **Qualificação Contínua**
- **Qualificação Intraempresa**
- **Qualificação REFA**
- **Estágio nas Empresas**
- **Serviço às Empresas**
- **Centro de Exames TELC**

www.dual.pt

- DUAL.QualificacaoProfissional
- @dual_qp
- company/dual-ccila
- @dual_qp
- DUALQualificaçãoProfissional

40 ANOS 1983 - 2023
HISTÓRIAS DE QUALIFICAÇÃO

UM SERVIÇO
AHK Deutsch-Portugiesische Industrie- und Handelskammer
Câmara de Comércio e Indústria Luso-Alemã

ENTIDADE FORMADORA CERTIFICADA

PRR Plano de Recuperação e Resiliência

REPÚBLICA PORTUGUESA

Financiado pela União Europeia
NextGenerationEU