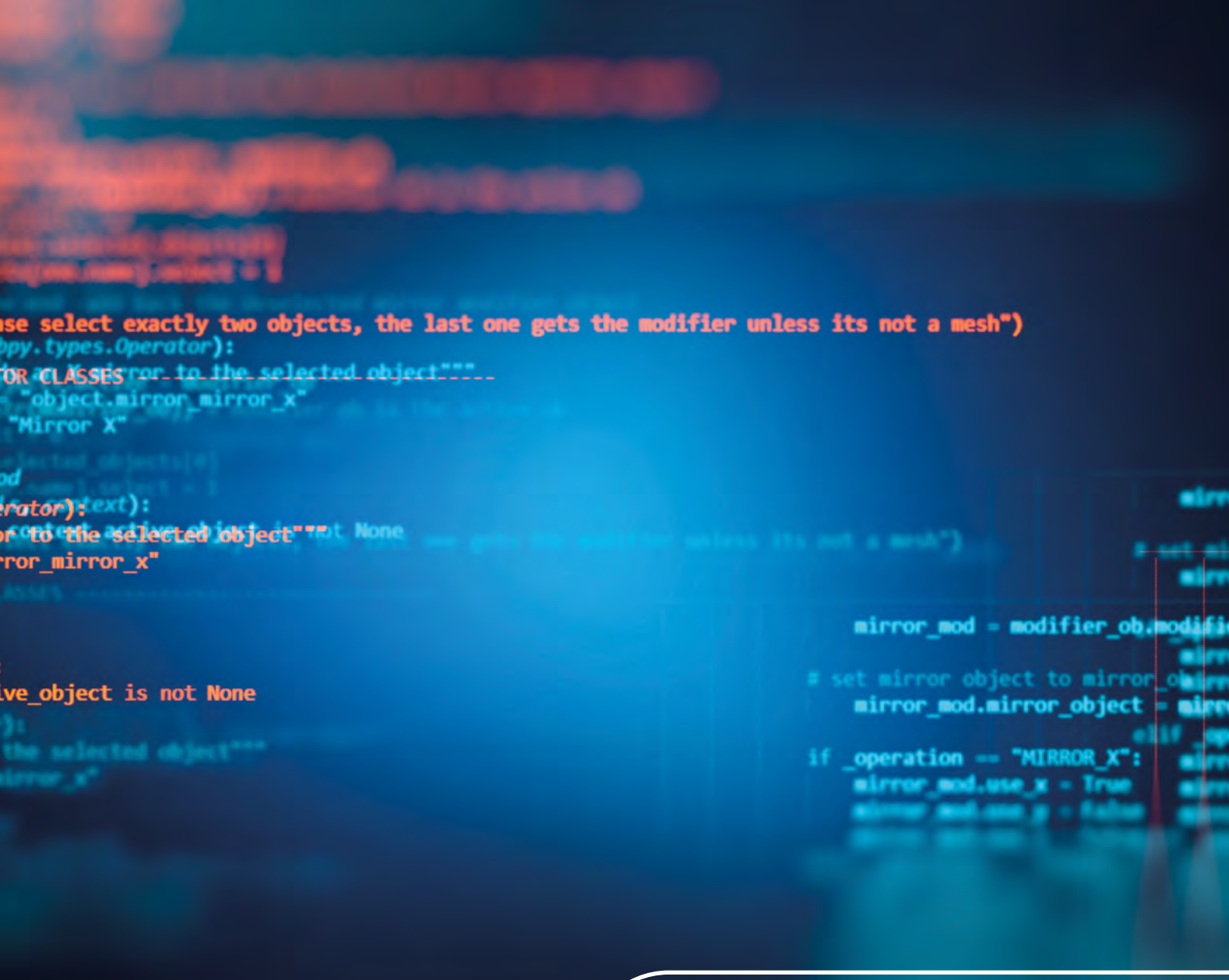

Mundo Tech

PROGRAMAÇÃO COM PYTHON

SENAI <LAB365>

SUMÁRIO

PROGRAMAÇÃO COM PYTHON

Caro estudante! Neste material, serão apresentados alguns fundamentos de Python que irão lhe auxiliar no andamento do curso de *Data Science*.

O Python é a linguagem de programação mais utilizada quando se trata de *Data Science*.

O QUE É O PYTHON?

As linguagens de programação podem ser entendidas como tradutoras entre programadores e computadores. Através delas, são passadas instruções para as máquinas, isso é, ocorre a programação. Essas instruções são passadas por meio de códigos, rotinas, scripts ou outros arquivos. Para utilizar uma linguagem de programação, precisamos obedecer às regras semânticas, bem como pontuações adequadas, palavras-chaves e até mesmo indentação (principalmente em Python). Essas regras são conhecidas como sintaxe, que varia dependendo da linguagem.

Python é uma linguagem de programação interpretada, de código-fonte aberto e disponível para vários sistemas operacionais. Diz-se que uma linguagem é interpretada se ela não precisa ser compilada (traduzida para uma linguagem da máquina), mas sim “lida” por um outro programa (chamado de interpretador), que traduz para a máquina o que seu programa quer dizer.

O Python é uma linguagem muito produtiva, de fácil aprendizado, concisa e que vem crescendo cada vez mais no ramo de criação de softwares. Ele é uma linguagem para uso geral, mas está cada vez mais popular no campo de Data Science.

Além disso, o Python é uma linguagem muito poderosa, sendo usada por várias grandes empresas, como Facebook, Google, NASA, Netflix, Dropbox e Instagram.

CONCEITOS BÁSICOS

Tipos

Um valor, como um número ou um texto, é algo comum em um programa. Estes valores são de diferentes tipos e, no Python, eles podem ser classificados em três tipos básicos:

- int - Um número inteiro;
- float - Um ponto flutuante;
- *string* - Uma sequência de caracteres.

Por exemplo: 1 e 2 são números inteiros e 'Hello World!' é um texto, também chamado de *string. Strings* são delimitadas por aspas (simples ou duplas) – e é exatamente dessa maneira que o interpretador Python também identifica uma *string*.

Ao contrário da maioria das outras linguagens, em Python, não precisamos declarar uma variável, ou seja, definir qual o tipo dela.

Por exemplo, caso se deseje atribuir o valor 3 à variável A, basta digitar A=9. O Python saberá que A é um inteiro (tipo "int"). Por outro lado, se o valor a ser armazenado for 37.8, que é um dado do tipo "ponto flutuante", este deveria ser expresso como A=37.8. Observe que, para o Python, A=9 e B=37.8 são variáveis de tipos diferentes e isto deve ser levado em conta ao se realizar certos tipos de manipulações de dados.

Variáveis

Podemos pedir para o Python lembrar de um valor para utilizá-lo em outro momento do programa. Para isso, ele irá guardar este valor em uma **variável**.

Variável é um nome que faz referência a um valor. É como se fosse uma etiqueta que colocamos naquele valor e, quando precisarmos usá-lo, o chamamos pelo nome atribuído na etiqueta.

Um comando de atribuição (o sinal de igualdade =) cria uma nova variável e atribui um valor a ela:

>>> mensagem = ‘ oi, python’
‘oi, python’
 	
>>> numero = 5
5
 	
>>> pi = 3.14
3.14
	Variável mensagem recebeu o valor “oi Python”. Por estar entre aspas duplas, significa que é uma *string*.
	
	Variável Número recebeu o valor 5. Isso significa que é um valor inteiro.
	
	Variável pi recebeu o valor 3.14. Isso significa que é um valor *float*.

Para recuperar esses valores, basta chamá-los pelos nomes das variáveis definidas anteriormente. E, pode-se utilizar a função type() para verificar os tipos de variáveis.

>>> mensagem
‘oi, python’
 	
>>> numero
5
 	
>>> pi
3.14
	>>> type(mensagem)
	<class ‘str’ >
	
	>>> type(numero)
	<class ‘int’>
	
	>>> type(pi)
	<class ‘float’>

Dica

Devemos escolher nomes de variáveis que forneçam algum significado ao código. Esses nomes podem ser bem longos ou conter letras e números. É uma convenção entre os programadores Python começar a variável com letras minúsculas e utilizar o underscore (_) para separar palavras como: meu_nome, numero_de_cadastro, telefone_residencial. Esse padrão é chamado de *snake case*.

Também não podemos iniciar o nome de uma variável com número.

Atribuição

Os operadores de atribuição realizam a ação de alocar valores para variáveis. Podem ser usados para realizar operações aritméticas e alocar o resultado simultaneamente.

No exemplo, a seguir, a variável X recebe o valor 4 e a variável y recebe o valor 2. Na segunda linha, significa que x recebe x + y.

```
>>> x = 4; y = 2
>>> x += y #equivale a x = x + y
>>> x
6
```

Operador	Descrição
=	Atribuição
+=	Atribuição com soma
-=	Atribuição com subtração
*=	Atribuição com multiplicação
/=	Atribuição com divisão

Comandos e Funções

Os comandos são palavras-chaves que executam certas ações, e as funções são procedimentos que devolvem dados, dependendo de seus argumentos.

As funções são similares a funções matemáticas, isto é, dependendo do argumento, o resultado é diferente.

Quando uma função é chamada, geralmente recebe argumentos, que são passados como parâmetros, entre parênteses. As respostas da função, quando há, são chamadas de retornos. Por exemplo, ao usar a função de raiz quadrada sqrt com argumento 25, é retornado o valor 5, ou seja, para:

```
x  =  sqrt(25)  A variável x receberá o valor 5.
```

O próprio usuário pode criar funções para serem utilizadas no código. Esse processo é denominado definição de funções.

Operadores Aritméticos

Operadores são símbolos especiais que representam cálculos, como adições e multiplicações. Para fazer cálculos com números, utilizamos os operadores +, -, *, / e **, que representam, respectivamente, adição, subtração, multiplicação, divisão e potenciação. Uma expressão é uma combinação de valores, variáveis e operadores, como x + 17, 1 + 1 etc. Quando digitamos uma expressão no modo interativo, o interpretador vai calcular e imprimir o resultado.

```
>>> 2  *  3
6
```

Também podemos utilizar variáveis.

```
>>> x = 1
>>> y = 3
>>> x + y
4
```

```
>>> x  -  y
-2

>>> x  *  y
3
```

```
>>> x  /  y
0.3333333333333333

>>> x  **  y
1
```

Operadores Lógicos

Esses operadores avaliam expressões lógicas e retornam verdadeiro ou falso. Por falso, entende-se o valor nulo, zero, e por verdadeiro, qualquer valor diferente de zero, por padrão, o valor um. Os operadores lógicos são muito utilizados em estruturas condicionais e estruturas de repetição.

Operador	Descrição
NOT	Não
AND	E
OR	OU

Operadores Relacionais

Os operadores relacionais avaliam expressões e retornam verdadeiro ou falso, assim como os lógicos. No entanto, os relacionais são responsáveis por comparações entre elementos.

Operador	Descrição	Exemplo	Resultado
==	Igual a	7 == 7	Verdadeiro
!=	Diferente de	7 != 8	Verdadeiro
<>	Diferente de	10 <> 10	Falso
>	Maior que	10 > 20	Falso
>=	Maior ou igual a	1234 >= 1234	Verdadeiro
<	Menor que	100 < 1000	Verdadeiro
<=	Menor ou igual a	1.25 <= 2.50	Verdadeiro

Importação

A linguagem Python é conhecida pelas diversas áreas de aplicações. Por vezes, tais áreas necessitam de comandos ou funcionalidades específicas que não estão, por padrão, no Python. Desta forma, elas acabam vindo de bibliotecas externas que precisam ser importadas, caso o usuário deseje utilizá-las. Para isso, é utilizado o comando *import*.

Indentação

A indentação é extremamente importante em Python, já que ela define um bloco. Na linguagem C, por exemplo, um bloco é definido por { e }. Em Python, normalmente são utilizados 4 espaços para indentação, e eles são preferidos no lugar de tabulações.

Exemplo de bloco na linguagem C	Exemplo de bloco na linguagem Python

`if (true) {`
`  int variavel_local = 1 ;`
`  dentroDoBloco( ) ;`
`}`
`foraDoBloco( ) ;`
	`if True:`
	`  variavel_local = 1`
	`  dentro_do_bloco( )`
	`fora_do_bloco( )`

Comentários

Comentário são informações que podem ser incluídas em um código e servem para documentação. Os comentários não são interpretados ou executados.

```
# isso é um comentário
def func( ):
    pass
```

Entrada de usuário

Podemos criar interatividade e pedir para o usuário entrar com um valor digitado do teclado. O Python possui uma função que captura a entrada de valores: a função input(). Quando essa função é chamada, o programa para e espera o usuário digitar alguma coisa. Quando o usuário aperta a tecla ENTER, o programa processa e imprime o valor digitado em forma de *string*.

```
>>> entrada  =  input( )
'oi pyhton'

>>> print(entrada)
'oi python'
```

Controle de Fluxo

O controle de fluxo é o controle da sequência de comandos executados pela rotina. Um código é sempre executado linha por linha, na ordem em que as encontram. Desta forma, se existem somente procedimentos simples, o funcionamento da rotina é linear.

Já com o controle de fluxo, podemos fazer com que o código tome decisões ou execute alguns procedimentos repetidas vezes. Tais ações equivalem a dizer para o compilador instruções como: "se isso, faça aquilo" ou "enquanto isso, execute aquilo".

Estrutura Condicional

As estruturas condicionais são responsáveis por tomadas de decisão, que indicarão determinados procedimentos ou outros, dependendo da condição avaliada. As mais populares das estruturas condicionais são if-else. A estrutura if avalia uma expressão: caso a expressão seja verdadeira, os procedimentos subordinados são executados. Não é obrigatório, mas, em seguida, pode-se adicionar apenas o else, com outros procedimentos subordinados, que serão executados caso a condição do if seja falsa.

```
x = int(input('digite um valor: '))
if x<10:
    print('x<10')
else:
    print('x>=10')
```

Neste exemplo, se o usuário informar um valor menor que 10, será mostrado que x<10. Caso contrário, ele irá mostrar que x>=10

Estrutura de Repetição

Já o segundo conjunto de estruturas, as estruturas de repetição, são responsáveis por criar laços que repetem os procedimentos. Cada repetição realizada é chamada de iteração.

Exemplo	Descrição	Explicação
1 `i = 0` 2 `while i<10:` 3 `    print(i)` 4 `    i +=1`	**While** O While avalia a condição a ele exposta a cada iteração. E, enquanto ela for verdadeira, o laço é mantido. Quando a condição é falsa, o laço é terminado.	Neste exemplo, o i começa valendo 0. O while avalia a condição, que é, inicialmente, verdadeira. Então a primeira iteração é realizada, que imprimirá 0 e incrementará o i. Isto acontecerá até que i seja igual a nove, pois assim será impresso 9, o i será incrementado para 10 e a condição para o while será falsa. Assim, o laço irá terminar.
1 `for i in range(10):` 2 `print(i)`	**for** O for, ao invés de avaliar uma condição, como o while, no Python, atribui valores a uma variável até que a lista deles acabe. Esses valores podem ser de tipos diferentes e são atribuídos segundo a ordem em que aparecem. Quando a lista de valores acaba, o *loop* termina. Para a atribuição, é necessário o operador de filiação in.	Neste caso, serão impressos os valores: 0, 1, 2, 3, 4, 5, 6, 7, 8 e 9.

O objetivo deste material é servir de guia para a linguagem de programação Python. Foram abordados aqui os conceitos básicos e a sintaxe Python. Divirta-se treinando!

SENAI <LAB365>