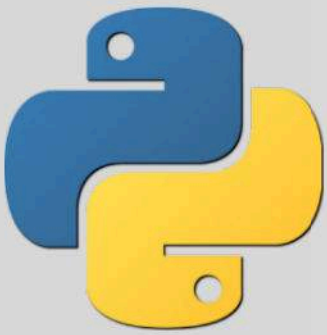
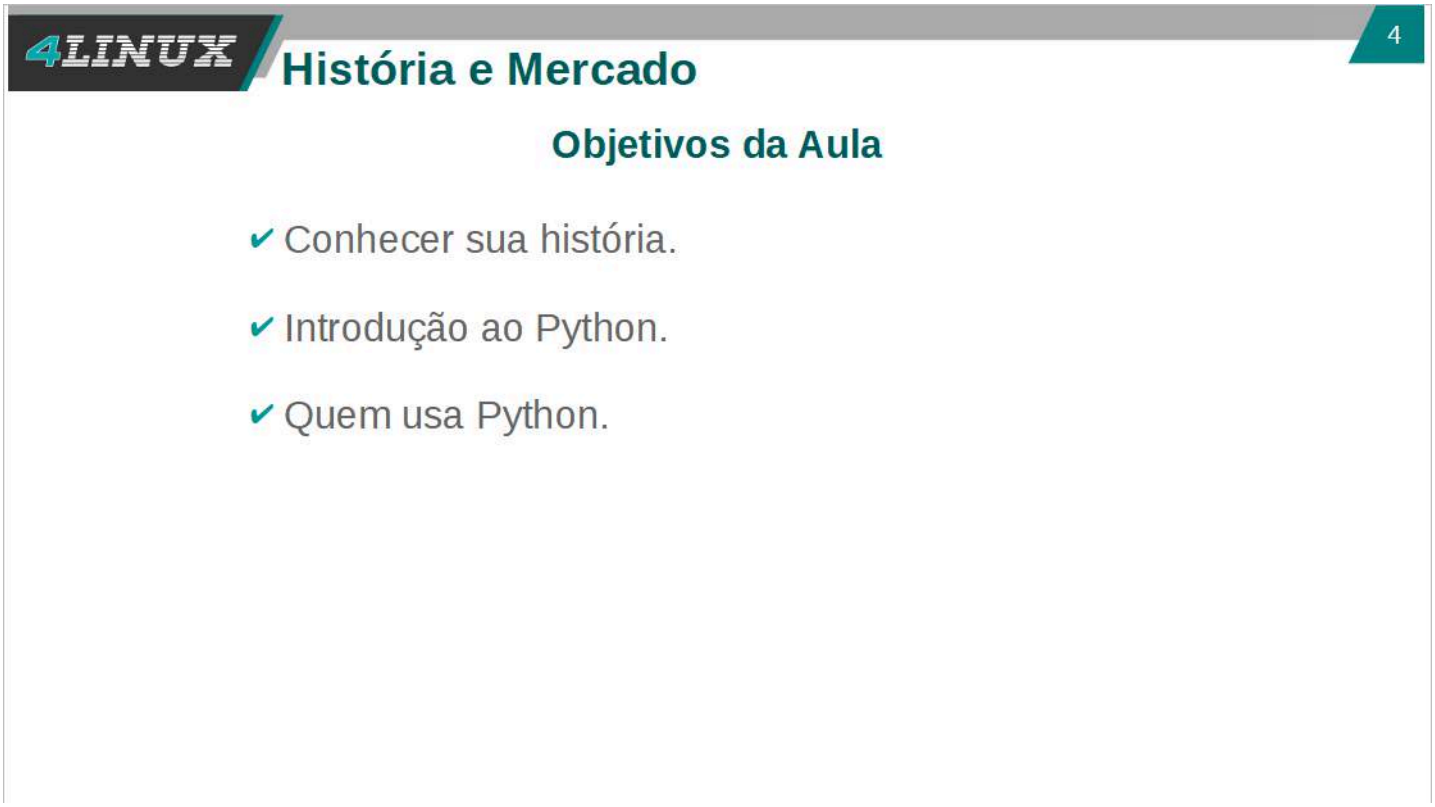
Python Fundamentals



História e Mercado

1.1

Anotações



História e Mercado

Nesta aula apresentaremos a história do Python e, empresas que o utilizam.

Anotações

Conhecer História

python™

1989	Criação do Python – Guido Van Rossum
1991	Classes e Herança
1994	Programação Funcional: lambda, map, filter, reduce
2001	Surgimento da Python Software Foundation
2008	Lançamento do Python 3
2012	Raspberry PI – Influência por Python

A linguagem Python surgiu em meados de 1989, criada por Guido Van Rossum, programador holandês. Com base na linguagem ABC cujo propósito era ensinar programação e prototipagem de programas, seguiu a mesma linha com o Python.

O nome Python, foi temporariamente dado pelo autor, por ser fã da série Monty Python, acabou ficando definitivo.

Hoje é uma linguagem onipresente em sistemas Linux, sendo usado pela RedHat, HP, Google, RackSpace, IBM, entre outras. Também possui variações para trabalhar com Java (Jython), DotNet (IronPython) e outras.

Na wiki da Python Software Foundation há uma lista dos usuários:
https://wiki.python.org/moin/OrganizationsUsingPython

Em 2015 foi considerada a 4ª linguagem mais popular pelo iee.org, perdendo somente para Java, C e C++.: http://spectrum.ieee.org/computing/software/the-2015-top-ten-programming-languages

Ficou também em 4º lugar no ranking da RedMonk:
http://redmonk.com/sogrady/2015/07/01/language-rankings-6-15/

Python é uma linguagem que serve a qualquer propósito e, se encaixa muito bem no contexto DevOps, que utilizaremos no decorrer deste curso.

Introdução ao Python

Conhecendo o Python

Alto Nível: podemos dizer que a linguagem é de alto nível por ser mais próxima à linguagem do ser
Humano, não diretamente relacionada ao computador, ou seja, não existe a necessidade do programador conhecer instruções de processador e afins.

Interpretada: significa uma linguagem de programação na qual não há necessidade de gerar um arquivo binário, você apenas terá o arquivo e, a partir disto é necessário um interpretador para conseguir executar o seu script ou aplicação.

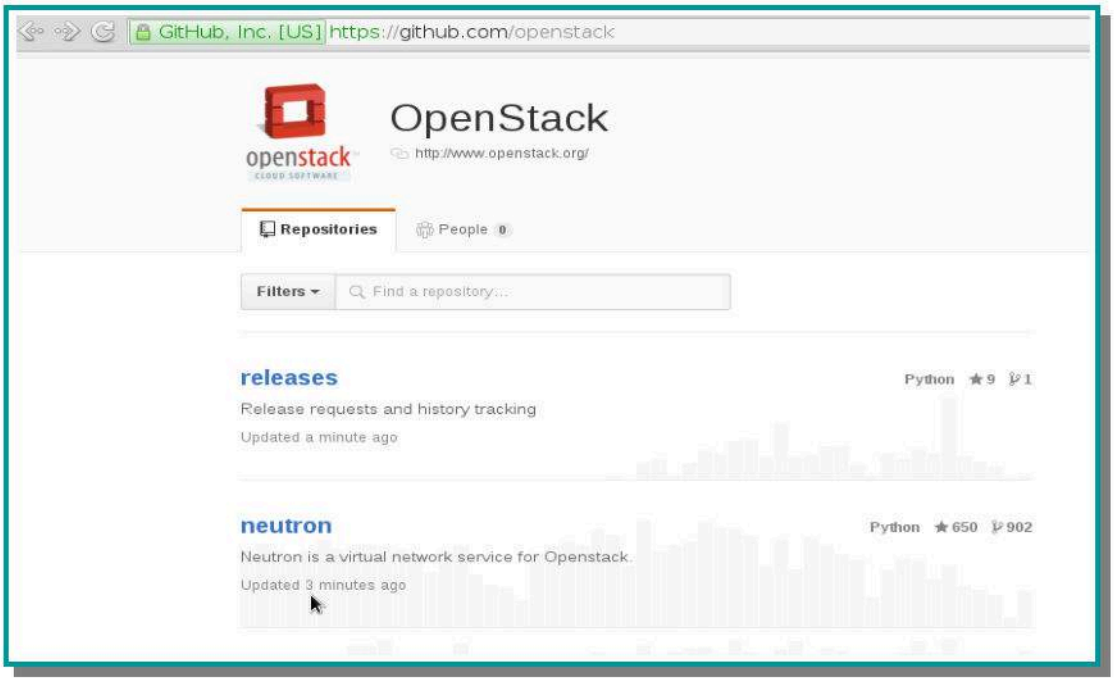
Tipagem Dinâmica: não é necessário definir qual o tipo de valor em uma variável, para criar, basta atribuir um valor a esta. Ex: idade = 30, mesmo eu não tendo definindo o tipo de valor, será assumido o tipo "int" (inteiro).

Suporta Módulos: podemos fazer a instalação de módulos externos, por exemplo: Paramiko para SSH ou o Flask para servidor web.

Permite utilizar Operadores: Conseguimos utilizá-los para adição, subtração, multiplicação, entre outros.

Legível: permite criar seu Code Style e sua identação, também possibilita definir blocos de código.

Linguagem Orientada a Objetos: Python tem recursos que dão suporte à Programação Orientada a
Objetos (POO).



OpenStack

OpenStack é a ferramenta de Virtualização OpenSource mais popular da atualidade, foi escrita utilizando a linguagem python.

OpenStack teve influência muito forte da RackSpace, empresa concorrente da Amazon WebServices e da Nasa (agência espacial americana), por disponibilizarem o código fonte no GitHub.

OpenStack: https://github.com/openstack

Anotações

Ansible

O Ansible é uma ferramenta para automatização de infraestrutura de T.I., criada com o propósito de automatizar a instalação de pacotes no Linux, Deploy de Aplicações, Gerenciamento de Configurações, Automatização de Tarefas e Monitoração.

Também escrito em python seu código fonte está disponível no github no seguinte link: https://github.com/ansible/ansible

Anotações

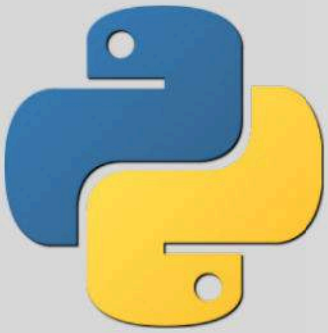
Quem usa Python

Globo

A Globo.com, também usa fortemente a linguagem de programação. É possível comprovar isso verificando a quantidade de repositórios na linguagem python que existentes no github: https://github.com/globocom

Anotações

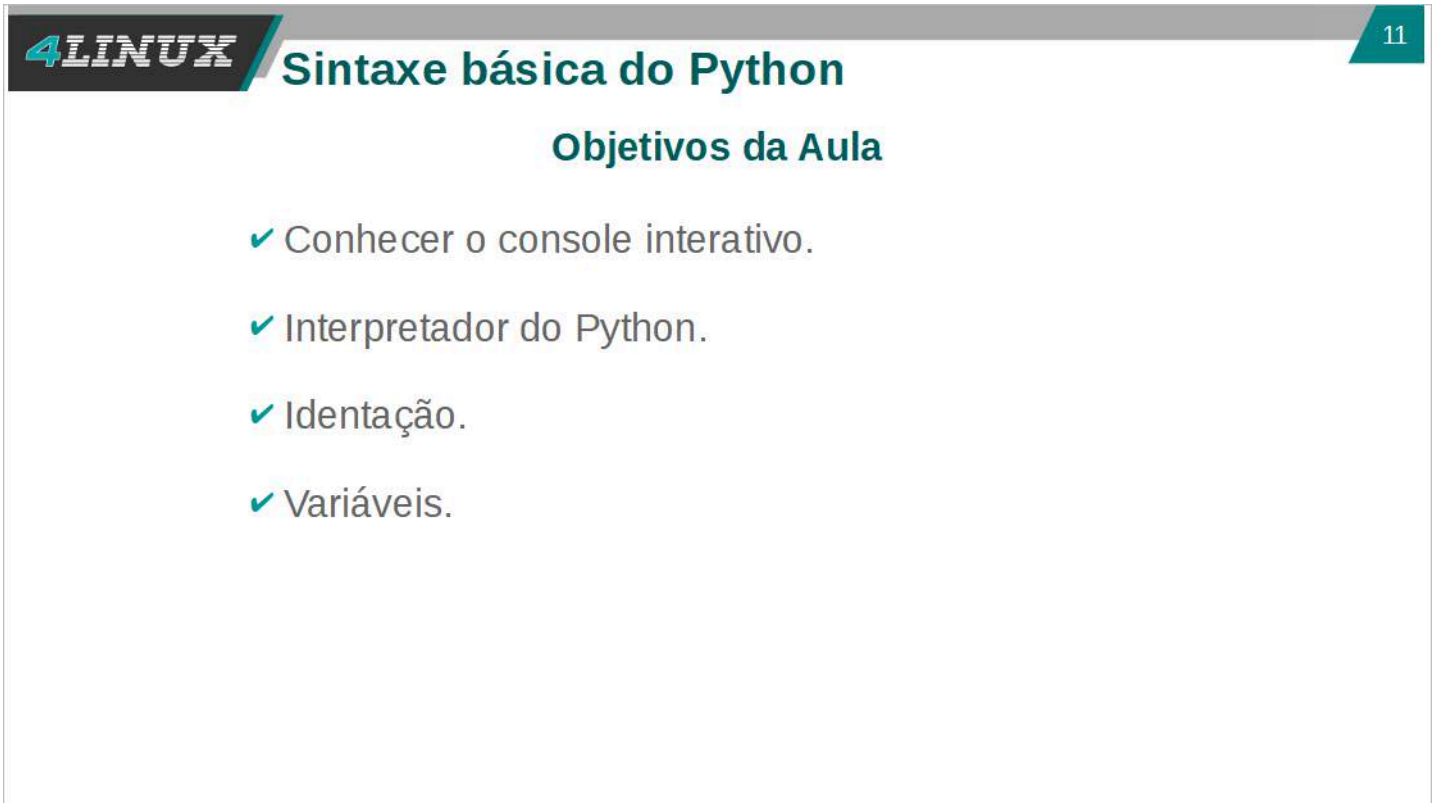
Python Fundamentals



Sintaxe Básica do Python

1.2

Anotações



Sintaxe básica do Python

Esta aula aborda o console interativo que a linguagem nos proporciona, entender sobre o interpretador do Python, como funciona a identação e como utilizar variáveis.

Anotações

Console Interativo

Python possui um modo interativo que permite “conversar” diretamente com o interpretador para definir variáveis, funções e classes.

```
root@developer:/# python3
>>> print "Agora você está no modo interativo :)"
Agora você está no modo interativo :)
>>> exit()
```

Console Interativo

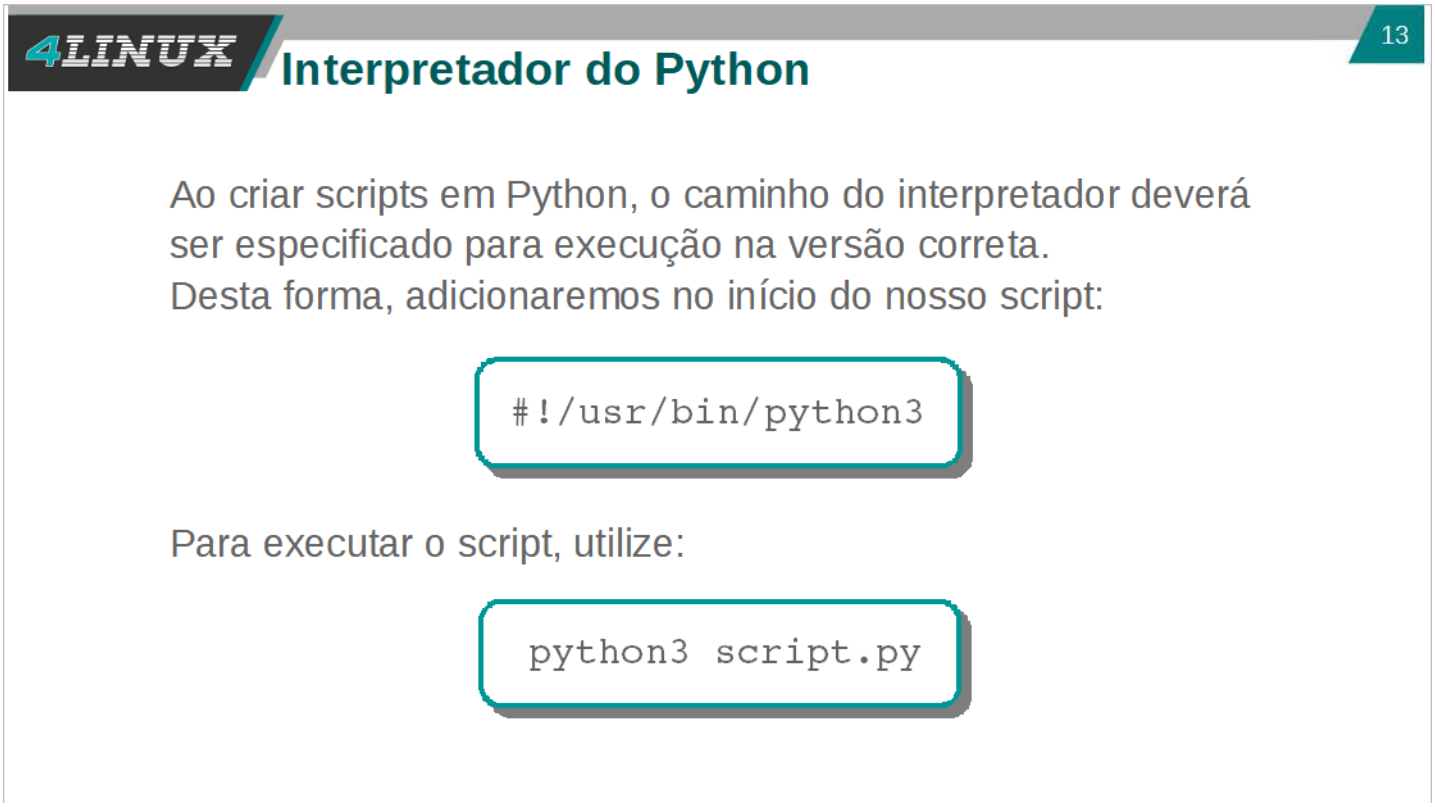
Ao acessar o Console Interativo, o interpretador imprimirá uma mensagem de boas vindas, o número da versão, uma nota copyright e o prompt, que pode ser identificado por três sinais de maior (“>>>”), caso apareça “...”, Salveo interpretador está aguardando completar o comando.

O console é utilizado para a realização testes, antes de colocarmos o código em nossa aplicação. Desta forma, conseguimos verificar se determinada função ou classe, funciona tendo o retorno esperado.

O Console Interativo, também é interessante para aprender, por exemplo: ao executar o comando “help()” ele trará a documentação de como utilizar um módulo.

Para saber mais sobre o que é programar em Python, digite “import this”. Estes princípios também podem ser encontrados em: https://www.python.org/dev/peps/pep-0020/

Anotações



Interpretador do Python

Para executar os scripts em Python, além da forma descrita, podemos utilizar ./script.py. Neste Caso, é obrigatório ter o interpretador especificado para que não ocorram erros.

Quando executamos com “python3 script.py” não existe esta necessidade, pois ao chamar o script já, estamos falando quem irá interpretar, neste caso o python3.

Para utilizar um script com Python na versão 2, basta especificar o seguinte interpretador: #!/usr/bin/python2 e pode ser executado com “python2 script.sh”.

Anotações

Identação

Python separa os blocos de código por identação:

```
#!/usr/bin/python3
print("Hello World!")
nome = input("Qual é a melhor linguagem de
   programação: ")
if nome == "python":
    print("Você acertou!")
else:
    print("Errou! =(")
```

Identação

A sintaxe do Python é diferente de outras linguagens, não possui chaves para separar os blocos de código.

Os códigos são separados por identação, conforme o exemplo no slide, isso facilita a organização do código e a sua leitura.

Em Python facilmente sabemos quando a programação está errada. Caso o código esteja muito para a direita, identificamos a ocorrência de muitas funções aninhadas que, provavelmente podem ser separadas facilitando a manutenção e evitando a duplicação de código.

Anotações

Comentários

Podemos adicionar comentários em nossos códigos. Serão ignorados pelo Python, mas, tornam o código organizado, facilitando sua compreensão.

```
#Comentário de uma linha
```

Desta forma podemos inserir comentários em quantas linhas forem necessárias!

Anotações

Variáveis

No Python existem os seguintes tipos de variáveis:

```
CONSTANTE = 3.14
numero = 1
decimal = 0.003
texto = "4Linux Devops"
dicionario = {"nome":"Guido","idade":59}
lista = ["item1","item2",3,"quatro",3.14]
tupla = (1,2,3,"Python")
```

Variável e Constante

Programas de computador utilizam os recursos de hardware mais básicos para executar algoritmos. Enquanto o processador executa os cálculos, a memória é responsável por armazenar dados e servi-los ao processador. O recurso utilizado nos programas para escrever e ler dados da memória do computador é conhecido como variável. Consiste simplesmente um espaço na memória no qual guardamos uma informação que reservamos a qual atribuiremos um nome. Por exemplo: podemos criar uma variável chamada "idade" para armazenar a idade de uma pessoa.

Chamamos variável, este espaço alocado na memória, porque o valor armazenado pode ser alterado ao longo do tempo, ou seja, o valor ali alocado é "variável" ao longo do tempo. Diferente das constantes, cujo espaço reservado, na memória, para armazenar um valor, não muda com o tempo. Por exemplo: o valor PI (3.14159265359...), nunca vai mudar!

Anotações

Python Code Style

1º	A Identação deve ser de 4 espaços (Sem Tabs).
2º	Limite de 79 caracteres por linhas (84 no máximo).
3º	Linhas muito longas são quebradas por \
4º	Alinhar os parênteses em caso de quebra de linha.
5º	As funções devem estar sempre 2 linhas abaixo a de cima.
6º	Não usar espaço depois de abrir ou fechar um parênteses.

Exemplos

Python não apresenta muitas regras, quanto à sua forma de programar. Esse code style, foi baseado nas práticas dos desenvolvedores do Flask.

Exemplos de boas práticas

Nome de funções:

```
funcao_com_nome_muito_grande(param1,param2) \
.cascateando_o_retorno()
```

Usando Parênteses:

```
quebrando_por_parenteses(param1,
                         Param2)
```

Em caso de listas:

```
alunos = [
    “se”, “forem”,”poucos”,”items”,
    “forem”,
    “muitos”
]
```

Em caso de funções:

```
def funcao1(param):
    print “4linux”

def funcao2(param):
    print “devops”
```

No caso de métodos dentro de uma classe, separa-se somente por uma linha em branco, em caso de funções, 2 linhas em branco.

Em caso de expressões:

```
valor = (num1 / num2) * num3 / num4
If var == “python”:
```

No caso de comparadores, a recomendação é usar a variável, sempre, antes da string ou número a ser comparado.

Caso sejam valores booleanos, faça o if da seguinte maneira:

```
If valor is True:
If  valor is False
```

Em caso de valores nulos:

```
If var is None:
```

Essas comparações também podem ser feitas utilizando o sinal de igual ==.

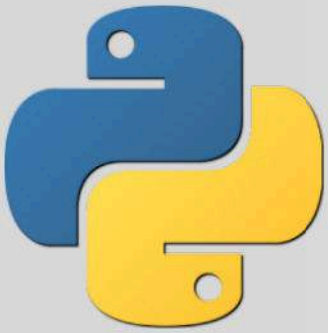
Exemplo:

```
If var == True
```

Porém, o comparador is, foi criado com o propósito de comparar esses valores, logo, deve ser usado.

Anotações

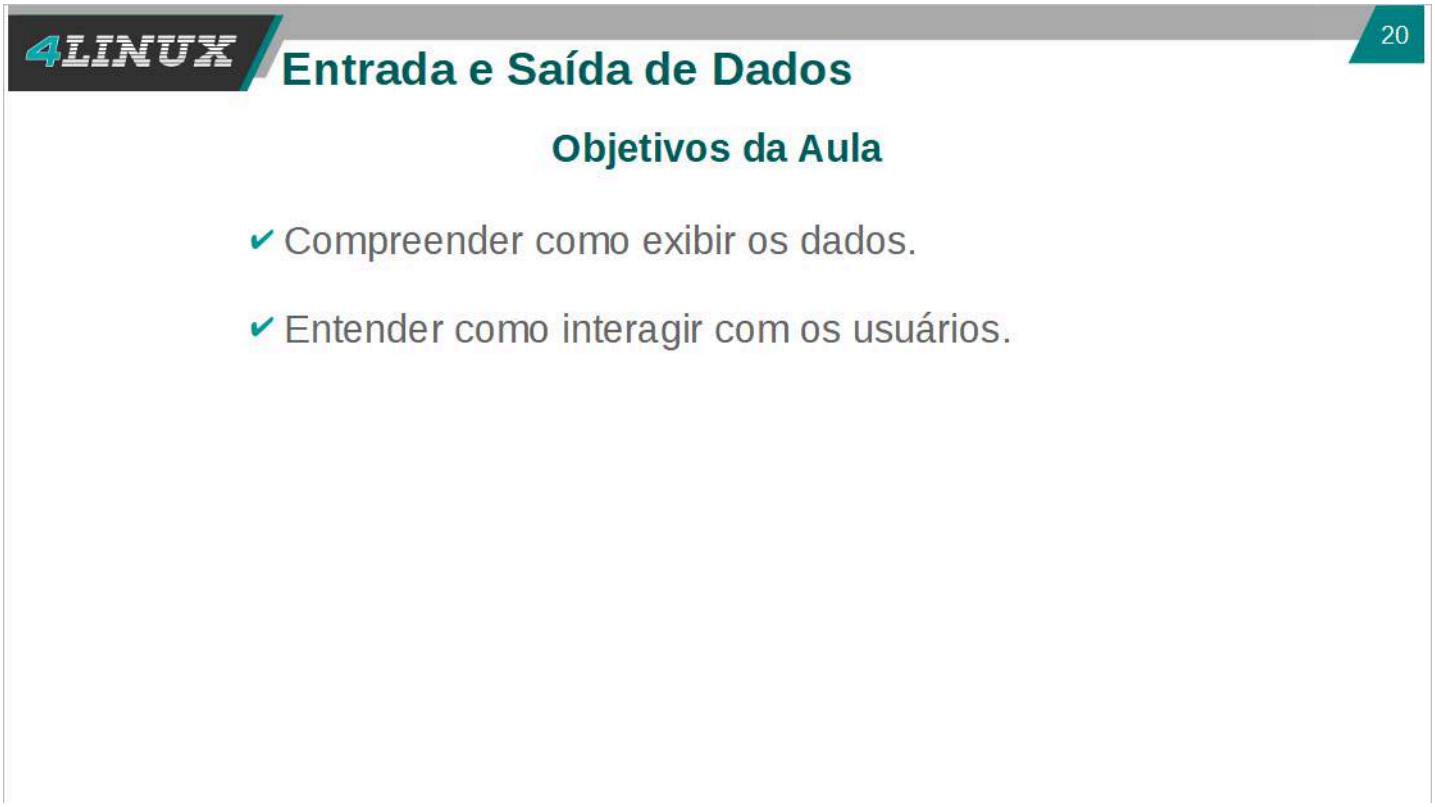
Python Fundamentals



Entrada e Saída de Dados

1.3

Anotações



Entrada e Saída de Dados

Nesta aula compreenderemos como interagir com o usuário usando entrada e saída de dados, deixando nossos programas dinâmicos.

Anotações

Entrada e Saída de Dados

Saída de Dados

✔ Utilizamos a função “print ()” para retornar os dados do programa para a saída padrão (a tela).

✔ Também utiliza-se a função, caso o programa apresente algum comportamento inesperado, assim conseguimos verificar o retorno dos valores esperados.

✔ Além desta função, podemos enviar a saída de dados para um arquivo e armazená-la para uso futuro.

Saída de Dados

No Python 3, o **print** deve ser realizado com parênteses, da seguinte forma: print (variavel).
Já no Python 2, não precisamos utilizar o parênteses, basta utilizar: print variavel.
Note também que em Python 3, conseguimos imprimir uma quantidade de valores, sendo possível definir o seu separador.

Veja todas as modificações na documentação oficial: https://docs.python.org/3.0/whatsnew/3.0.html

Anotações

Entrada e Saída de Dados

Utilizando o Print

Para retornar um valor para o usuário, utilizaremos o “print”:

```
#/usr/bin/python3
animal = "Leão"
print(animal)
```

Podemos definir o separador quanto retornamos dois valores:

```
nome = "Guido"
sobrenome = "van Rossum"
print(nome, sobrenome, sep=".", end="\n\n")
```

Parâmetros da função print.

A Função print possui parâmetros:
*args: permite passar diversos valores para impressão na tela.
sep: é o separador de cada valor, por padrão, recebe um espaço vazio como default sep=’ ‘.
end: utilizado para quebrar linhas ou organizar a exibição dos valores após a execução da função. Por padrão, recebe um ‘\n’ como default, equivalente a uma quebra de linha end=”\n”.
Na ausência da especificação destes parâmetros, a função executará com os valores default.

Anotações

Entrada e Saída de Dados

Entrada de Dados

✔ Até este momento trabalhamos de forma estática, ou seja, sem interagir com o usuário. Python permite que o usuário faça inserção de dados.

✔ Os dados inseridos pelo usuário, sempre serão interpretados como “string” e, deverão ser tratados em nosso programa.

✔ O que for inserido pelo usuário, é armazenado em uma variável e utilizada no programa.

Entrada de Dados

No Python 3, a entrada de dados é realizada apenas com o comando **“input”**, que tratará todas as inserções da mesma forma (como string).

No Python 2, há dois comandos para a entrada de dados, **“raw_input”** para guardar dados do tipo texto e “input” para guardar qualquer outro tipo de dado. Por exemplo:

```
>>> nome = raw_input("Digite o seu nome: ")
Digite o seu nome: Mariana
>>> idade = input("Digite a sua idade: ")
Digite a sua idade: 50
>>> print type(nome)
<type 'str'>
>>> print type(idade)
<type 'int'>
```

Anotações

Entrada e Saída de Dados

Utilizando o Input

Para interagir com usuário, utilizaremos a função “input”:

```
#!/usr/bin/python3
input('Digite o nome de um animal: ')
```

Podemos gravar esta interação em uma variável:

```
#!/usr/bin/python3
animal = input('Digite o nome de um animal: ')
print(animal)
```

Entrada e Saída de Dados

Com a função input recebemos dados do usuário, serão interpretados como string, armazenados em uma variável e, posteriormente exibido o valor da variável com a função print.

Anotações