

Python e Teste

Porque usar ?

Python

- Linguagem de programação multi-
 - Uma comunidade ampla;
 - Web, ciência de dados, simulação, comunicação, e imagem
- Tarefas complexas são executadas com um único comando
 - Programação fácil (escrita e leitura)
 - Acervo grande de funções (bibliotecas por áreas)
 - Integração com outras linguagens (C, C++, Java e Javascript)
- Um passo apenas
 - Linguagem interpretada;
 - O código produzido é executado rapidamente

Um bom testador...

- Habilidades fundamentais
 - Front end: CSS, HTML, script
 - Automação: webDriver, java, javascript, python, Ruby
 - Manuseio de código: git, git-*, BASH
- Frameworks para Automação de teste
 - Robot: (para que?)
 - PyTest:
 - PyUnit:
 - Selenium:

Canais de interação

- Grupo no Whats app
- Repositório no GitHub @cavmelo/IARTES

Modulo #1

Agenda

- Instalação
- Ambiente de programação
- Primeiro programa/comentário
- Tipos de dados
- Operadores
- Variável
- Entrada e saída

Instalação

- Tem passo a passo disponível para cada plataforma.
 - Windows: <https://python.org.br/instalacao-window>
 - Linux: <https://python.org.br/instalacao-linux>
 - Mac: <https://python.org.br/instalacao-mac>
- Instale a versão 3 (python3)
 - Python3
- Na linha de comando:
 - Python --version
 - Saida esperada: Python 3.x.x
 -

Ambiente de programação

- Crie uma pasta (e.g. modulo_python)
- Instale um ambiente virtual
 - Dicas no arquivo Ambiene.md (repositório GIT)
- Instale uma IDE
 - Dica: PYCharm
 - <https://www.jetbrains.com/pycharm/download>
- Associe o ambiente virtual ao IDE
 - Depende da IDE usada
 - Garanta que os ambientes estejam associados

Primeiro programa

- Abra o seu ambiente de programação
 - IDE para quem instalou
 - Editor de tex
- O clássico "olá Python"
- Linguagem interpretada
 - Execute o seu código sendo o resultado esperado a mensagem

Tipo de dado

Tipos de dados

- Inteiros - int
- Ponto flutuante - float
- Número complexo - complex
- Booleano - bool
- Strings - str
- Lista - list
- Dicionários - dict
- Tuplas - tup
- Conjuntos - set
- Representação do tipo de valor que se tem no contexto das computações que um programa pode realizar;
- Tipos representam conceitos oriundo da matemática
- Associados a palavras especiais;

Inteiros

- Representa conceito definido na matemática;
 - Sem parte fracionária
 - Exemplos: 3, 10, -15, -100
 - O inteiro é representado por **int**
 - Por ser positivo e negativo
- Como é usado?
 - `idade = int(input())`
 - `Idade: int`

Ponto flutuante

- Representa conceito definido na matemática;
 - Exemplos: 3.141516, 10.1924 e 1.2550
 - O ponto flutuante é representado por **float**
 - Por ser positivo e negativo
- Como é usado?
 - `peso = float(input())`
 - `peso: float`

Número complexo

- Representa conceito definido na matemática;
 - Exemplos: $3+3j$
 - O número complexo é representado por **complex**
 - Existe uma parte real e uma imaginária
- Como é usado?
 - Vamos

Booleano

- Representa dois valores
 - True - verdadeiro
 - False - falso
- O booleano é representado por **bool**
- **Construção de expressões lógicas**
- Como é usado?
 - cansado:bool
 - cansado = False

Strings

- Coleção de caracteres
 - Usa-se aspas simples ou dupla para caracterização da **string**
 - "Introdução a python"
 - 'Python é uma interessante'
 - A string é representado por **str**
- Como é usado?
 - endereco:str
 - endereco = 'Rua A, 100'

Lista

- Coleção de elementos (ordenado)
 - Pode conter elementos duplicados
 - caracterização da **lista**
 - []
 - ['Python' , "interessante"]
 - A lista é representado por **list**
- Como é usado?
 - compras:list
 - compras = ["arroz", "carne", "verduras"]
 - meu_estoque: list = ["arroz", 2,5.20, "feijão", 2, 3.90"]

Conjunto

- Coleção de elementos (não ordenado)
 - Não permite elementos **duplicados**
 - caracterização da **conjunto**
 - {}
 - {'Python', 'interessante'}
 - O Conjunto é representado por **set**
- Como é usado?
 - compras:set
 - compras = {"arroz", "carne", "verduras"}
 - item_estoque:set = {"arroz", 2,5.20}

Dicionário

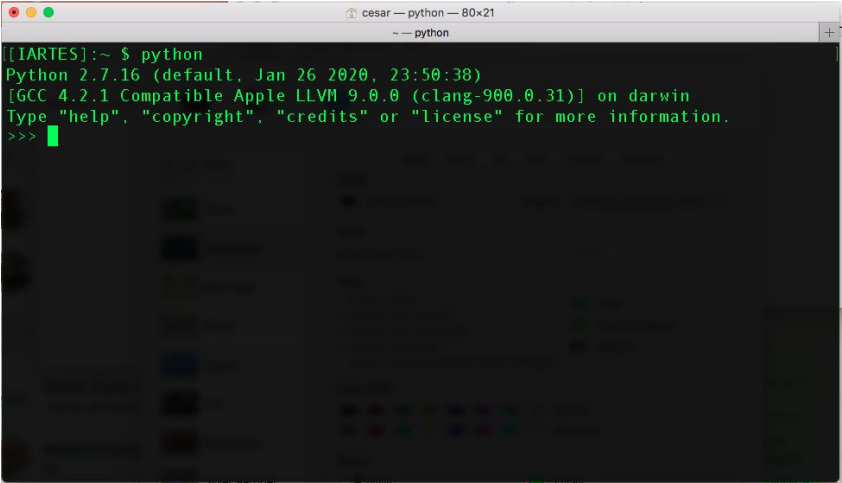
- Coleção de elementos (não ordenado) de pares chave-valor
 - Caracterização do **dicionário**:
 - **{"chave": "valor associado"}**
 - {'Python': 'interessante'}
 - O Dicionário é representado por **dict**
- Como é usado?
 - compras: dict
 - compras = {"arroz": "2kg", "carne": "3kg", "verduras": "1mc"}
 - precos: dict = {"arroz": "4.0", "frango": "10.0"}

Tuplas

- Coleção de elementos (ordenado) de objetos
- Tuplas são **imutáveis**
- Caracterização da **tupla**
 - `()`
 - `('Python','interessante', 2.8)`
- A Tupla é representada por **tuple**
- Como é usado?
 - comprado: tuple
 - comprado = ("carro",0K, 2022)
 - gasolina: tuple = ("normal", 5.79)

Vamos interagir com Python shell

- Tipos de dados (type)
- Verifique o tipo inteiro
- Verifique o tipo real
- Verifique o tipo string
- Verifique o tipo dicionário
- Verifique o tipo tupla
- Verifique o tipo conjunto
- Verifique o tipo lista



```
cesar — python — 80x21
python
[[IARTES]:~ $ python
Python 2.7.16 (default, Jan 26 2020, 23:50:38)
[GCC 4.2.1 Compatible Apple LLVM 9.0.0 (clang-900.0.31)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> █
```

Operadores

Operadores

- Operadores são símbolos especiais que indicam que certa computação deve ser realizada.
 - Os valores envolvidos em uma operação são chamados de **operandos**
 - Exemplos:
 - $a = 10$
 - $b = 20$
 - $a + b$
- Operandos podem ser variáveis (vamos já ver) e/ou literais
- Uma sequência de operandos e operadores é chamada de **expressão**
 - $a + b - 10$

Operadores Aritméticos

| Operador | Exemplo | Significado | Resultado |
|------------|---------|-----------------|--|
| +(unário) | +a | unário positivo | a |
| +(binário) | a + b | adição | Soma de a e b |
| -(unário) | -a | unário negativo | Inverte o sinal do valor |
| -(binário) | a - b | subtração | b subtraído de a |
| * | a * b | multiplicação | Produto de a e b |
| / | a / b | divisão | Quociente quando a é dividido por b. resultado é sempre um float |

Operadores Aritméticos (cont.)

| Operador | Exemplo | Significado | Resultado |
|----------|----------|-----------------|---|
| % | $a \% b$ | módulo | Resto quando a é dividido por b |
| // | $a // b$ | divisão inteira | Quociente quando a é dividido por b. O valor é o menor inteiro mais próximo. |
| ** | $a ** b$ | Exponenciação | a na potência b. |

Vamos interagir com o Python Shell

- `a = 10`
- `b = 3`
- `+a` (unário positivo)
- `a + b`
- `-b` (unário negativo)
- `a - b`
- `a * b`
- `a / b`
- `a // b`
- `a ** b`
- `a % b`

- Verifique os tipos de dados dos resultados
 - ◆ Lembre-se: `type(a+b)`
- Verifique o valor resultante das operações
- Atribua um valor real a variável **b**, e veja se ocorre alteração no tipo de dado dos resultados

Cuidado...

| Precedência | Operador | Descrição |
|-------------|-------------|---|
| Baixa | +, - | Soma e subtração |
| | *, /, //, % | Multiplicação, divisão, divisão inteira, módulo |
| | +x, -x | Unário (negativo e positivo) |
| Alta | ** | Potenciação |

- $a = 10$
- $b = 3$
- $a + b * 3$
- $(a + b) * 3$
- $a - b * 3$
- $(a - b) * 3$
- $a - b ** 2$
- $(a - b) ** 2$

Variável

Variável

Na programação de computadores, variáveis armazenam informações que serão referenciadas e usadas por programas.

- **Tem tipo**, exemplo: float, int, str e bool;
- **Tem nome**, formado por letras, alguns símbolos, e números;
- **Tem sensibilidade**, diferença entre Maiúscula e minúscula;
- **Tem contexto**, define a visibilidade do nome;

- Exemplos de definição:

- idade = 21
- type(idade)
- 1idade = 21
- idade1 = 21
- \$idade = 21
- idade\$ = 21
- IDADE = 21

Variáveis (cont.)

- É possível atribuir valores para múltiplas variáveis em uma linha;
 - É possível atribuir um valor para múltiplas variáveis;
 - Use o ***print()*** para imprimir os valores armazenados em uma variável
- `x, y, z = "laranja", "banana", "tucumã"`
 - `x = y = z = "tucumã"`
 - `print(x, y, z)`
 - `print(x + y + z)`

Entrada e Saída

Entrada e Saída

- Permite receber informações para realizar uma computação.
 - Entrada padrão é o teclado
 - Usa o comando `input()`;
 - **`cidade = input()`**
- A informação lida precisa ser tratada
 - String por padrão
 - Outros tipos precisam ser convertidos
 - `idade = int(input())`
 - `preco = float(input())`
- Escreva o script que faça a leitura de um nome próprio e depois imprima o valor lido.
- Tente fazer a leitura de dois nomes próprios usando duas variáveis diferentes na mesma linha.

Entrada e Saída

- Dados podem ser lidos de um arquivo
- O arquivo precisa estar disponível para a leitura
 - `f = open("meuarquivo.txt", "r");`
 - O primeiro parâmetro é no `nome_do_arquivo`;
 - O segundo parâmetro é o modo de leitura:
 - `"r"` - Read (leitura)
 - `"a"` - Append (escrita final)
 - `"w"` - Write (abre para escrita)
 - `"x"` - Create (cria o arquivo)
- O nome do arquivo pode conter o diretório (pasta) onde o arquivo está localizado;
- `f.read()` e `f.readline()`

Entrada e Saída

Permite que resultados de computações sejam enviados para a saída padrão (tela).

- O comando para realizar a saída de dados na tela é o **print()**
- Aceita qualquer tipo de dados
 - nome='Maria Júlia'
 - print(nome)
- Aceita imprimir resultados de expressões
 - nome='Maria Júlia'
 - print(nome + " da silva")

Entrada e Saída

Saída também pode ser feita para arquivo. Nesse caso é preciso ter um arquivo aberto para escrita.

- Pode ser feito usando o comando ***open()***
 - `f = open("meuarquivo.txt", "w")`
 - `f.write("olá mundo")`
- A escrita ocorre da esquerda para direita, de cima para baixo.
- É possível deslocar o ponteiro de escrita para qualquer parte:
 - `f.seek(2)` → escrita a partir do byte 2