



OPERADORES, EXPRESSÕES E VARIÁVEIS

The background of the slide features a complex digital theme. It includes a network diagram on the left with nodes and connecting lines, and a large, faint circular graphic on the right that resembles a stylized 'C' or a data path. The entire background is overlaid with binary code (0s and 1s) in various orientations and sizes. A solid red horizontal band runs across the middle of the slide, framing the central text.

TIPOS DE VARIÁVEIS NUMERICAS

TIPOS DE VARIÁVEIS - NUMERICAS

TIPOS DE VARIÁVEIS

- Variáveis têm outras propriedades além de nome e conteúdo. Uma das propriedades é o tipo no qual define a natureza dos dados que a variável armazena. Os mais comuns são os números inteiros, números de ponto flutuantes, string (cadeia de caracteres) e booleano (armazena estado lógico Verdadeiro ou Falso).
- A seguir veremos algumas dos tipos de variáveis:

VARIÁVEIS NUMERICAS

- Variáveis numerais são aquelas que armazenam números inteiros ou de ponto flutuantes.
 - **NUMERO INTEIRO** são aqueles sem parte decimal: 1, 0, -5, 550, -45, 3000.
 - O **TAMANHO DA VARIÁVEL** para **NÚMEROS INTEIROS EM PYTHON**, utiliza um sistema de precisão **ILIMITADO**.

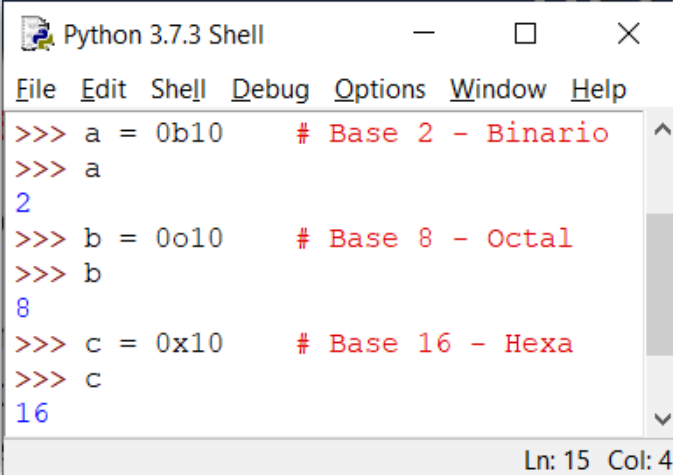
TIPOS DE VARIÁVEIS - NUMERICAS

- **NUMERO PONTO FLUTUANTE (DECIMAIS)** são aqueles com parte decimal: 1.0, 5.478, 10.5, 30000.4, 1.0
- O **TAMANHO DA VARIÁVEL** para **NÚMEROS PONTO FLUTUANTE EM PYTHON**, utiliza um sistema de precisão **LIMITADO** que vai de **$2.2250738585072014 \cdot 10^{-308}$** até **$1.7976931348623157 \cdot 10^{308}$** .
- Em Python é utilizado o ponto “.” e não a virgula “,” para separar parte inteira da fracionaria.
- Números grandes podem ser separados com sublinhado entre os dígitos (apenas a partir do Python 3.6).
- Exemplo:
 - ✓ a) 1_000 é o mesmo que 1000.
 - ✓ b) 1_000_000 é o mesmo que 1 milhão 1000000
 - ✓ c) 1_980.10 é o mesmo que 1980,10 (Podendo ser combinado com ponto)
- **CONSTANTE** em Python não é possível criar constantes, apenas crie uma variável e não mude o seu valor. Existem “artifícios”, não tão seguro que será visto mais a frente;

TIPOS DE VARIÁVEIS - NUMERICAS

REPRESENTAÇÃO NUMERICA EM BASES DIFERENTES

- Por padrão Python adota a base decimal (0, 1, 2, 3, 4, 5, 6, 7, 8 e 9) na base 10.
- Em Python é possível trabalhar com outras bases (2 - binaria, 8 - octal, 16 - hexadecimal), porem independente da base utilizada para introduzir o numero, o mesmo será mostrado sempre em decimal (base 10) como valor já convertido.
- Para trabalhar com outras bases diferente da decimal, utiliza-se os seguintes prefixos ao entrar com o valor da variável:
 - **BINARIO:** 0b (zero b)
 - **OCTAL:** 0o (zero ó)
 - **HEXADECIMAL:** 0x (zero x)



```
Python 3.7.3 Shell
File Edit Shell Debug Options Window Help
>>> a = 0b10      # Base 2 - Binario
>>> a
2
>>> b = 0o10      # Base 8 - Octal
>>> b
8
>>> c = 0x10      # Base 16 - Hexa
>>> c
16
Ln: 15 Col: 4
```

The background of the slide features a complex digital theme. It includes a network diagram on the left with nodes and connecting lines, and a large, faint circular graphic resembling a globe or a data sphere in the center. The entire background is overlaid with a pattern of binary code (0s and 1s).

TIPOS DE VARIÁVEIS LOGICAS / RELACIONAIS

TIPOS DE VARIÁVEIS – LÓGICAS / RELACIONAIS

EXPRESSÕES LÓGICAS

- Os operadores lógicos podem ser combinados em expressões lógicas.
- Quando uma expressão tiver mais de um operador lógico, avalia-se na ordem de prioridades:
 1. Operador not (Negação)
 2. Operador and (E)
 3. Operador or (OU)

EXEMPLO:

True or False and not True
True or False and False
True or False
True

TIPOS DE VARIÁVEIS – LÓGICAS / RELACIONAIS

OPERADORES RELACIONAIS

- Os operadores relacionais são utilizados para realizar comparações lógicas.
- O resultado de uma comparação com os operadores relacionais sempre serão True (Verdadeiro) ou False (Falso).
- Se utiliza o termo “avaliar” para indicar a resolução de uma expressão:
- Os operadores relacionais são:

OPERADOR	OPERAÇÃO	SIMBOLO MATEMATICO
==	Igualmente	=
>	Maior que	>
<	Menor que	<
!=	Diferente	≠
>=	Maior ou igual	≥
<=	Menor ou igual	≤

TIPOS DE VARIÁVEIS – LÓGICAS / RELACIONAIS

EXEMPLO:

```
Python 3.7.3 Shell
File Edit Shell Debug Options Window Help
Python 3.7.3 (v3.7.3:ef4ec6ed12, Mar 25 2019, 21:26:53) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> a = 1          # a recebe 1
>>> b = 5          # b recebe 5
>>> c = 2          # c recebe 2
>>> d = 1          # d recebe 1
>>> a == b         # a é igual a b ?
False
>>> b > a          # b é maior que a ?
True
>>> a < b          # a é menor que b ?
True
>>> a == d         # a é igual a d ?
True
>>> b >= a         # b é maior ou igual a a ?
True
>>> c <= b         # c é menor ou igual a b ?
True
>>> d != a         # d é diferente de a ?
False
>>> d != b         # d é diferente de b ?
True
>>>
```

Ln: 23 Col: 4

TIPOS DE VARIÁVEIS – LÓGICAS / RELACIONAIS

OPERADORES LÓGICOS

- São operações com logicas booleana, nos quais utilizam operadores lógicos.
- Em Python existem 3 tipos de operadores básicos:

OPERADOR PYTHON	OPERAÇÃO
not	Negação
and	E
or	OU

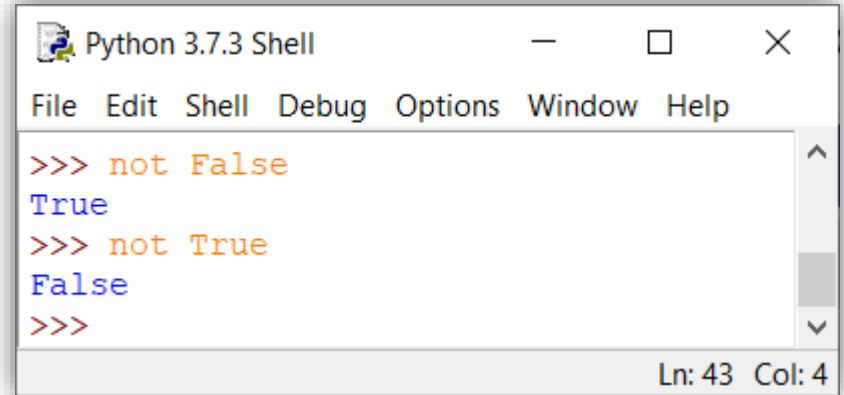
- Os OPERADORES LÓGICOS podem ser classificados em UNARIO e BINARIO:
 - ✓ **OPERADOR UNARIO** – Quando utiliza apenas um operador (Ex: negação).
 - ✓ **OPERADOR BINARIO** – Quando utiliza dois operador (Ex: E e OU)

TIPOS DE VARIÁVEIS – LÓGICAS / RELACIONAIS

OPERADOR NOT

- Obedece a logica booleana a seguir (revisão):

X	not X
V	F
F	V



```
Python 3.7.3 Shell
File Edit Shell Debug Options Window Help
>>> not False
True
>>> not True
False
>>>
```

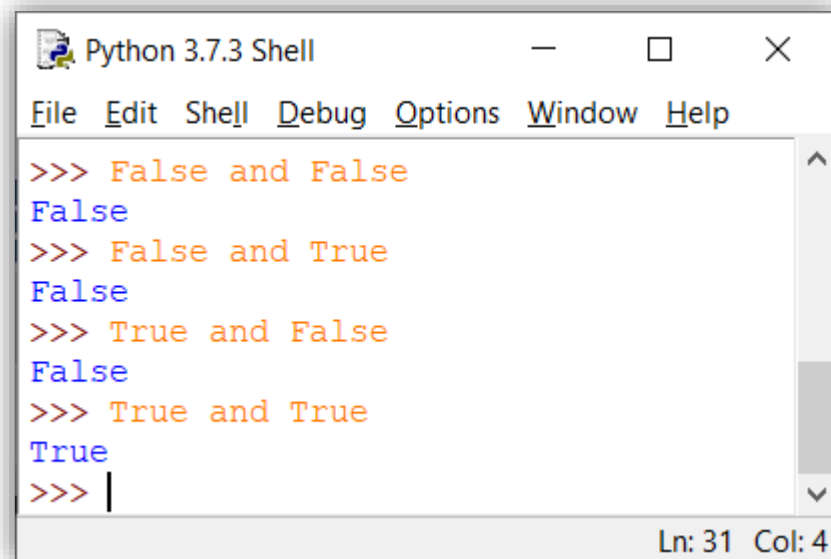
Ln: 43 Col: 4

TIPOS DE VARIÁVEIS – LÓGICAS / RELACIONAIS

OPERADOR AND

- Obedece a logica booleana a seguir (revisão):

X	Y	X and Y
F	F	F
F	V	F
V	F	F
V	V	V



```
Python 3.7.3 Shell
File Edit Shell Debug Options Window Help
>>> False and False
False
>>> False and True
False
>>> True and False
False
>>> True and True
True
>>> |
```

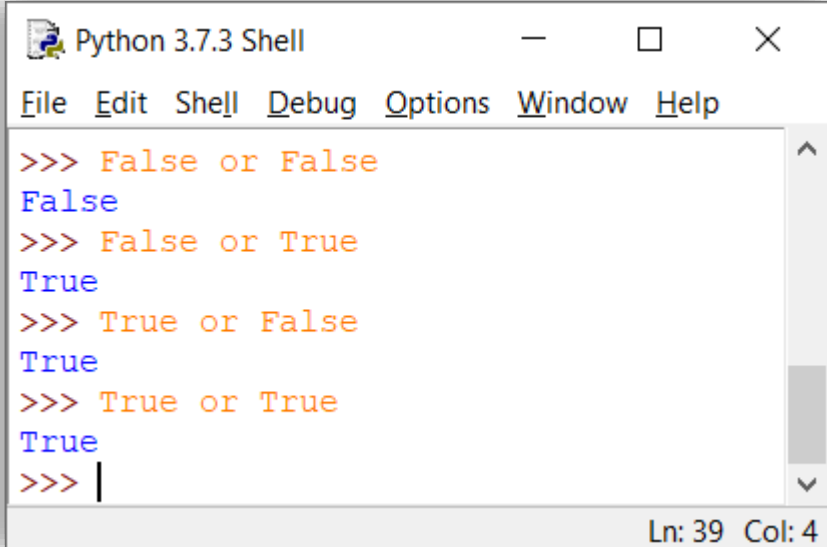
Ln: 31 Col: 4

TIPOS DE VARIÁVEIS – LÓGICAS / RELACIONAIS

OPERADOR OR

- Obedece a logica booleana a seguir (revisão):

X	Y	X or Y
F	F	F
F	V	V
V	F	V
V	V	V



```
Python 3.7.3 Shell
File Edit Shell Debug Options Window Help
>>> False or False
False
>>> False or True
True
>>> True or False
True
>>> True or True
True
>>> |
```

Ln: 39 Col: 4

TIPOS DE VARIÁVEIS – LÓGICAS / RELACIONAIS

EXPRESSÕES LÓGICAS

- Os operadores lógicos podem ser combinados em expressões lógicas.
- Quando uma expressão tiver mais de um operador lógico, avalia-se na ordem de prioridades:
 1. Operador not (Negação)
 2. Operador and (E)
 3. Operador or (OU)

EXEMPLO:

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

TIPOS DE VARIÁVEIS – LÓGICAS / RELACIONAIS

- Os operadores relacionais podem ser utilizados em expressões com operadores lógicos.

EXEMPLO:

Adotando Salario = R\$ 100 e Idade 20

Salario > 1000 and idade > 18

Salario > 1000 and idade > 18

100 > 1000 and 20 > 18

False and True

False

EXEMPLO:

Adotando Salario = R\$ 2000 e Idade 30

Salario > 1000 and idade > 18

Salario > 1000 and idade > 18

2000 > 1000 and 30 > 18

True and True

True

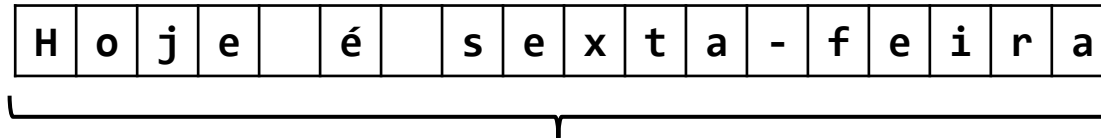
The background of the slide features a complex digital aesthetic. It includes a grid of binary digits (0s and 1s) in various shades of gray. Overlaid on this are several concentric, semi-transparent circular lines that create a sense of depth and motion. In the upper left corner, there is a small, stylized circular logo. In the lower left corner, a network diagram shows several nodes connected by lines, with a small box containing the binary sequence '01001011' below it. The central text is prominently displayed in a dark gray rectangular area.

TIPOS DE VARIÁVEIS STRING

TIPOS DE VARIÁVEIS – STRING

VARIAVEIS DO TIPO STRING

- São variáveis que armazenam cadeias de caracteres como nomes e textos em geral.
- Chama-se **CADEIA DE CARACTERES** uma **SEQUENCIA DE SÍMBOLOS** como letras, números, sinais de pontuação, espaço, etc.
- Exemplo: A frase “Hoje é sexta-feira” temos a seguinte string:



String

- É utilizado aspas (") para delimitar o inicio e o fim da sequencia de caracteres. Exemplo:

Print (“mensagem”)

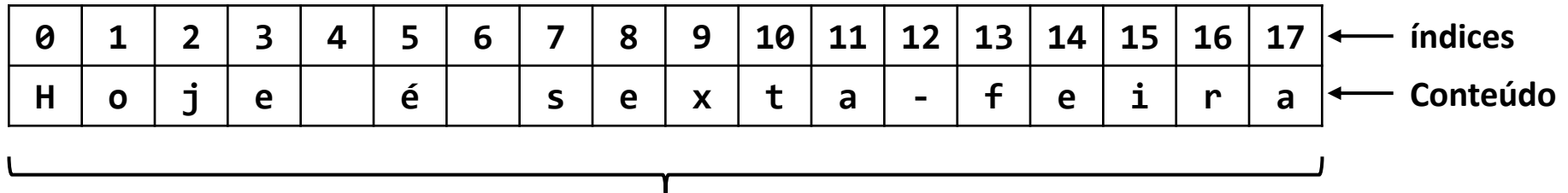
- O espaço é entendido como uma string vazia, porem é contabilizada.

TIPOS DE VARIÁVEIS – STRING

ÍNDICE DE UMA STRING

- Cada caractere de uma string é chamada de **ÍNDICE**.
- O tamanho de uma string depende da quantidade de índices no qual a compõem. No exemplo “**Hoje é sexta-feira**” a string é composta por 18 índices.
- A contagem do índice começa da esquerda para direita e começa pelo numeral 0;
- Exemplo: A frase “Hoje é sexta-feira” temos a seguinte string e índices:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	← índices
H	o	j	e		é		s	e	x	t	a	-	f	e	i	r	a	← Conteúdo



String com 18 índices

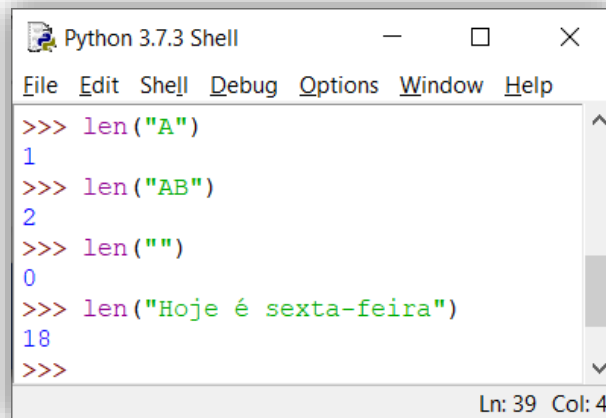
TIPOS DE VARIÁVEIS – STRING

TRABALHANDO COM STRING

- Cada string tem um tamanho associado e o seu conteúdo pode ser acessado caractere a caractere.

FUNÇÃO LEN

- Para consultar o tamanho de uma string utiliza-se a função **len**. Essa associação retorna o numero de caracteres na string.
- A função len retorna um valor do tipo inteiro, representando a quantidade de caracteres contidos na string.



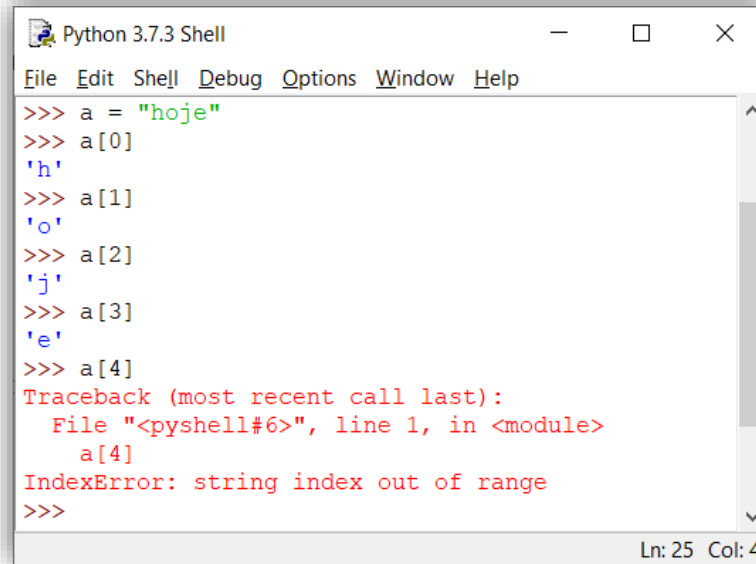
```
Python 3.7.3 Shell
File Edit Shell Debug Options Window Help
>>> len("A")
1
>>> len("AB")
2
>>> len("")
0
>>> len("Hoje é sexta-feira")
18
>>>
```

Ln: 39 Col: 4

TIPOS DE VARIÁVEIS – STRING

ACESSANDO O INDICIE DE UMA STRING

- Para acessar os caracteres (índice) de uma string, deve-se informar o índice ou posição do caractere entre colchetes.
- Como o primeiro caractere de uma string é índice 0, pode-se acessar valores de 0 até o tamanho da string menos 1;
- Exemplo: A string “hoje” tem tamanho 4 e pode ser acessada os índices de 0 a 3. O acesso de um índice maior que a quantidade de caracteres da string, o interpretador emitirá uma mensagem de erro:



```
Python 3.7.3 Shell
File Edit Shell Debug Options Window Help
>>> a = "hoje"
>>> a[0]
'h'
>>> a[1]
'o'
>>> a[2]
'j'
>>> a[3]
'e'
>>> a[4]
Traceback (most recent call last):
  File "<pyshell#6>", line 1, in <module>
    a[4]
IndexError: string index out of range
>>>
```

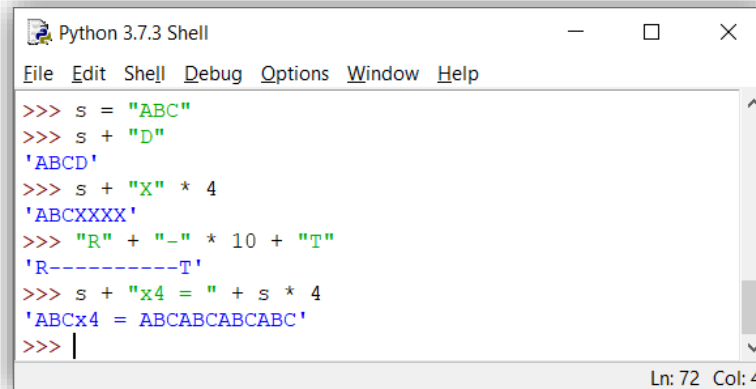
Ln: 25 Col: 4

TIPOS DE VARIÁVEIS – STRING

OPERAÇÕES COM STRING

CONCATENAÇÃO

- Concatenação é um termo usado em computação para designar a operação de unir o conteúdo de duas strings.
- Para concatenar duas string, utiliza-se o operador de adição (+).
- A concatenação pode ocorrer apenas com strings.
- Para concatenar repetindo uma string por varias vezes (caso especial) é utilizado o operador de multiplicação (*)



```
Python 3.7.3 Shell
File Edit Shell Debug Options Window Help
>>> s = "ABC"
>>> s + "D"
'ABCD'
>>> s + "X" * 4
'ABCXXXX'
>>> "R" + "-" * 10 + "T"
'R-----T'
>>> s + "x4 = " + s * 4
'ABCx4 = ABCABCABCABC'
>>> |
```

Ln: 72 Col: 4

TIPOS DE VARIÁVEIS – STRING

COMPOSIÇÃO

- Composição de string é utilizada para apresentar mensagens com conteúdo de variável ou variáveis.
- Exemplo: “**João tem x anos**” onde x é o valor da idade de João.
- Existem 3 formas diferentes de executar essa finalidade.

OPÇÃO 1 : MARCADOS DE POSIÇÃO

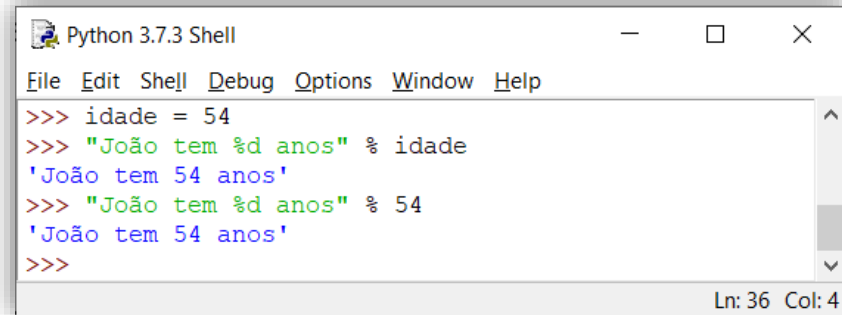
- Escreve-se a string, substituindo a variável por um marcador de posição e referenciando a variável na qual será alocado na marcação
- A marcação pode ser do tipo numero inteiro, string ou numero decimal, conforme representado na tabela abaixo:

MARCADOR	TIPO
%d	Numero Inteiro
%s	Strings
%f	Numero Decimal

TIPOS DE VARIÁVEIS – STRING

NUMERO INTEIRO - %d

- É alocado na string o %d onde será substituído pela variável ou valor indicado do tipo inteiro.
- Exemplo: “João tem 54 anos”



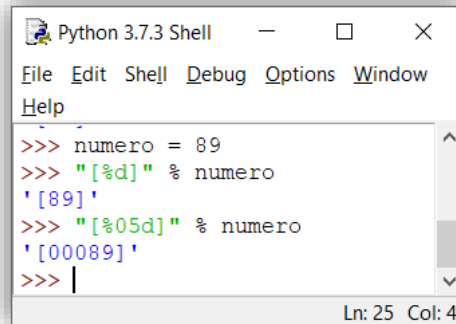
```
Python 3.7.3 Shell
File Edit Shell Debug Options Window Help
>>> idade = 54
>>> "João tem %d anos" % idade
'João tem 54 anos'
>>> "João tem %d anos" % 54
'João tem 54 anos'
>>>
```

Ln: 36 Col: 4

FORMATAÇÃO DO NUMERO INTEIRO:

➤ FIXAR POSIÇÕES NUMÉRICAS, COMPLETANDO COM ZERO A ESQUERDA.

- **%0nd** Onde: **n** é o numero de posições numéricas reservadas para a variável
- Exemplo:



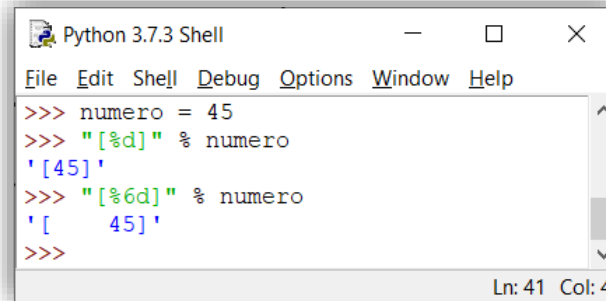
```
Python 3.7.3 Shell
File Edit Shell Debug Options Window Help
>>> numero = 89
>>> "[%d]" % numero
'[89]'
>>> "[%05d]" % numero
'[00089]'
>>> |
```

Ln: 25 Col: 4

TIPOS DE VARIÁVEIS – STRING

➤ FIXAR POSIÇÕES NUMÉRICAS, SEM COMPLETAR ZERO A ESQUERDA.

- **%nd** Onde: **n** é o numero de posições numéricas reservadas para a variável

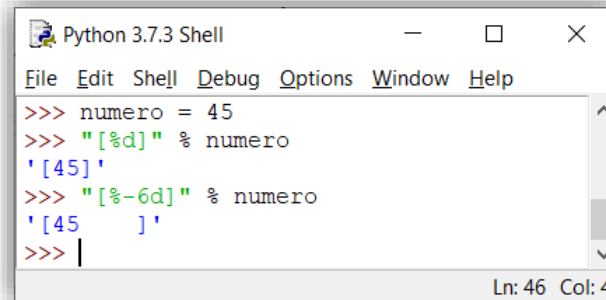


```
Python 3.7.3 Shell
File Edit Shell Debug Options Window Help
>>> numero = 45
>>> "[%d]" % numero
'[45]'
>>> "[%6d]" % numero
'[      45]'
>>>
```

Ln: 41 Col: 4

➤ FIXAR POSIÇÕES NUMÉRICAS, ALINHADO A PARTIR DA ESQUEURDA.

- **%-nd** Onde: **n** é o numero de posições numéricas reservadas para a variável

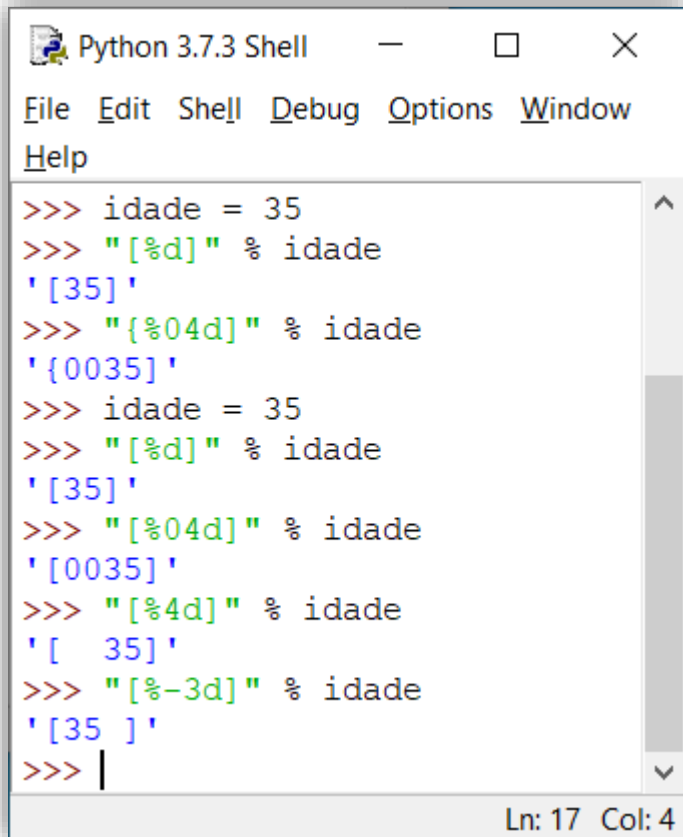


```
Python 3.7.3 Shell
File Edit Shell Debug Options Window Help
>>> numero = 45
>>> "[%d]" % numero
'[45]'
>>> ["%-6d]" % numero
'[45      ]'
>>> |
```

Ln: 46 Col: 4

TIPOS DE VARIÁVEIS – STRING

- Exemplo: “João tem **idade** anos”



```
Python 3.7.3 Shell
File Edit Shell Debug Options Window Help
>>> idade = 35
>>> "[%d]" % idade
'[35]'
>>> "{%04d}" % idade
'{0035}'
>>> idade = 35
>>> "[%d]" % idade
'[35]'
>>> "[%04d]" % idade
'[0035]'
>>> "[%4d]" % idade
'[ 35]'
>>> "[% -3d]" % idade
'[35 ]'
>>> |
```

Ln: 17 Col: 4

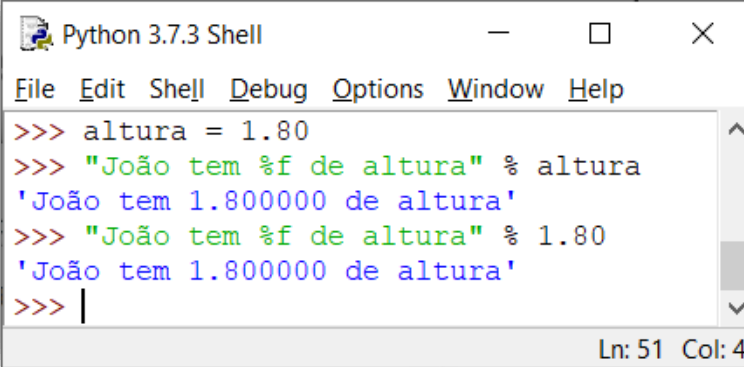
RESUMO:

- %d** – Padrão, sem formatar.
- %0nd** – Limitando 'n' casas e preenchendo 0 a esquerda.
- %nd** – Limitando 'n' casas e sem preencher 0 a esquerda.
- %-nd** – Limitando 'n' casas e alinhando a esquerda.

TIPOS DE VARIÁVEIS – STRING

NUMERO DECIMAL - %f

- É alocado na string o %f onde será substituído pela variável ou valor indicado do tipo decimal.
- Exemplo: **“João tem 1.80 de altura”**



```
Python 3.7.3 Shell
File Edit Shell Debug Options Window Help
>>> altura = 1.80
>>> "João tem %f de altura" % altura
'João tem 1.800000 de altura'
>>> "João tem %f de altura" % 1.80
'João tem 1.800000 de altura'
>>> |
```

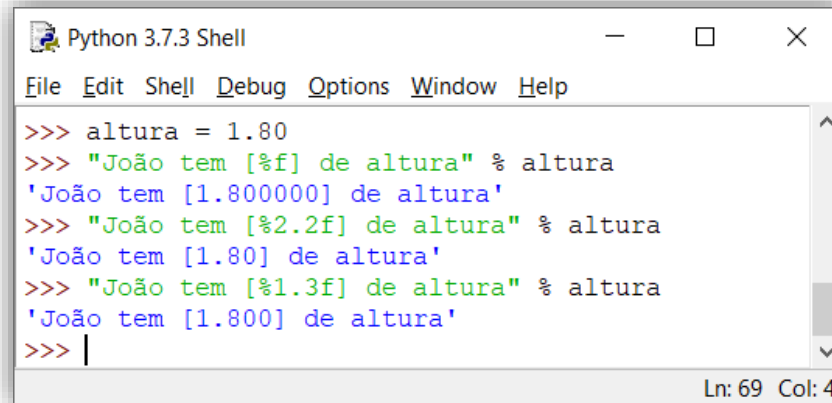
Ln: 51 Col: 4

- A formatação dos números decimais utilizam dois valores entre o símbolo % e a letra f

TIPOS DE VARIÁVEIS – STRING

FORMATAÇÃO DO NUMERO DECIMAL:

- `% X . Y f`
- Onde **X** é o numero de posição da **parte inteira** e **Y** é o numero de posição da **parte decimal**.
- Exemplo: “João tem 1.80 de altura”



```
Python 3.7.3 Shell
File Edit Shell Debug Options Window Help
>>> altura = 1.80
>>> "João tem [%f] de altura" % altura
'João tem [1.800000] de altura'
>>> "João tem [%2.2f] de altura" % altura
'João tem [1.80] de altura'
>>> "João tem [%1.3f] de altura" % altura
'João tem [1.800] de altura'
>>> |
```

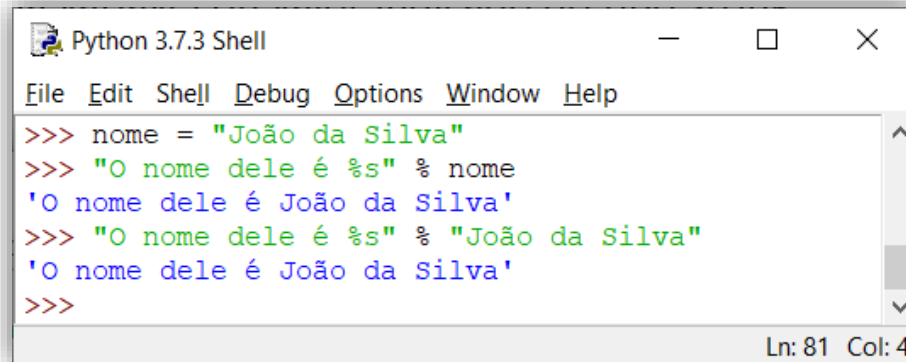
Ln: 69 Col: 4

- O valor da parte inteira é reservado, porem o interpretador não muda em decorrência dessa variação.

TIPOS DE VARIÁVEIS – STRING

STRING - %s

- É alocado na string o %s onde será substituído pela variável ou valor indicado do tipo string.
- Exemplo: **“O nome dele é João da Silva”**



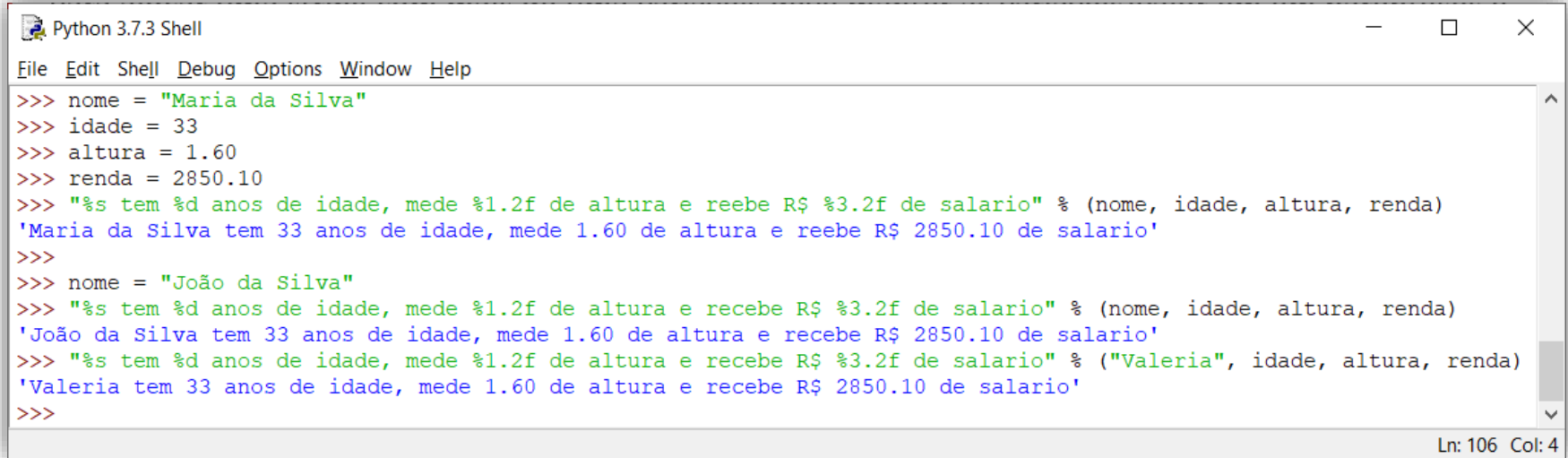
```
Python 3.7.3 Shell
File Edit Shell Debug Options Window Help
>>> nome = "João da Silva"
>>> "O nome dele é %s" % nome
'O nome dele é João da Silva'
>>> "O nome dele é %s" % "João da Silva"
'O nome dele é João da Silva'
>>>
```

Ln: 81 Col: 4

TIPOS DE VARIÁVEIS – STRING

COMPOSIÇÃO DE STRING COM MARCADOR DE POSIÇÃO:

- Python suporta diversas operações com marcadores.
- Quando se tem mais de um marcador na string, os valores devem ser escritos na ordem na qual é chamada na string e entre parênteses.



```
Python 3.7.3 Shell
File Edit Shell Debug Options Window Help
>>> nome = "Maria da Silva"
>>> idade = 33
>>> altura = 1.60
>>> renda = 2850.10
>>> "%s tem %d anos de idade, mede %1.2f de altura e reebe R$ %3.2f de salario" % (nome, idade, altura, renda)
'Maria da Silva tem 33 anos de idade, mede 1.60 de altura e reebe R$ 2850.10 de salario'
>>>
>>> nome = "João da Silva"
>>> "%s tem %d anos de idade, mede %1.2f de altura e recebe R$ %3.2f de salario" % (nome, idade, altura, renda)
'João da Silva tem 33 anos de idade, mede 1.60 de altura e recebe R$ 2850.10 de salario'
>>> "%s tem %d anos de idade, mede %1.2f de altura e recebe R$ %3.2f de salario" % ("Valeria", idade, altura, renda)
'Valeria tem 33 anos de idade, mede 1.60 de altura e recebe R$ 2850.10 de salario'
>>>
```

Ln: 106 Col: 4

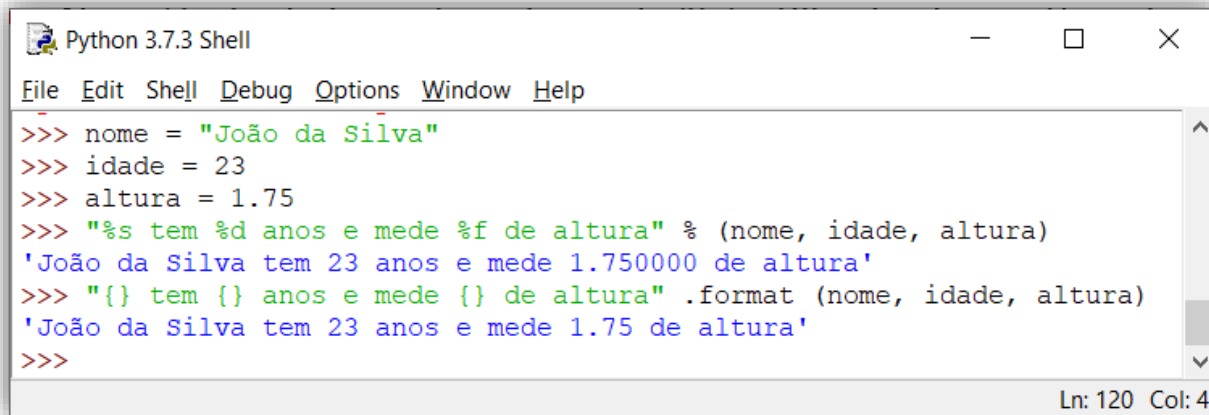
- O método de marcador de posição é análogo ao utilizado em outras linguagens, porem tem caído em desuso com métodos mais avançados. Porem seu domínio ajuda na interpretação de programas escritos no método (antigos).

TIPOS DE VARIÁVEIS – STRING

OPÇÃO 2 : METODO FORMAT

- No método de format, no lugar do % é utilizado chaves {} e dos parênteses o **.format**.

EXEMPLO:



```
Python 3.7.3 Shell
File Edit Shell Debug Options Window Help
>>> nome = "João da Silva"
>>> idade = 23
>>> altura = 1.75
>>> "%s tem %d anos e mede %f de altura" % (nome, idade, altura)
'João da Silva tem 23 anos e mede 1.750000 de altura'
>>> "{} tem {} anos e mede {} de altura".format(nome, idade, altura)
'João da Silva tem 23 anos e mede 1.75 de altura'
>>>
```

Ln: 120 Col: 4

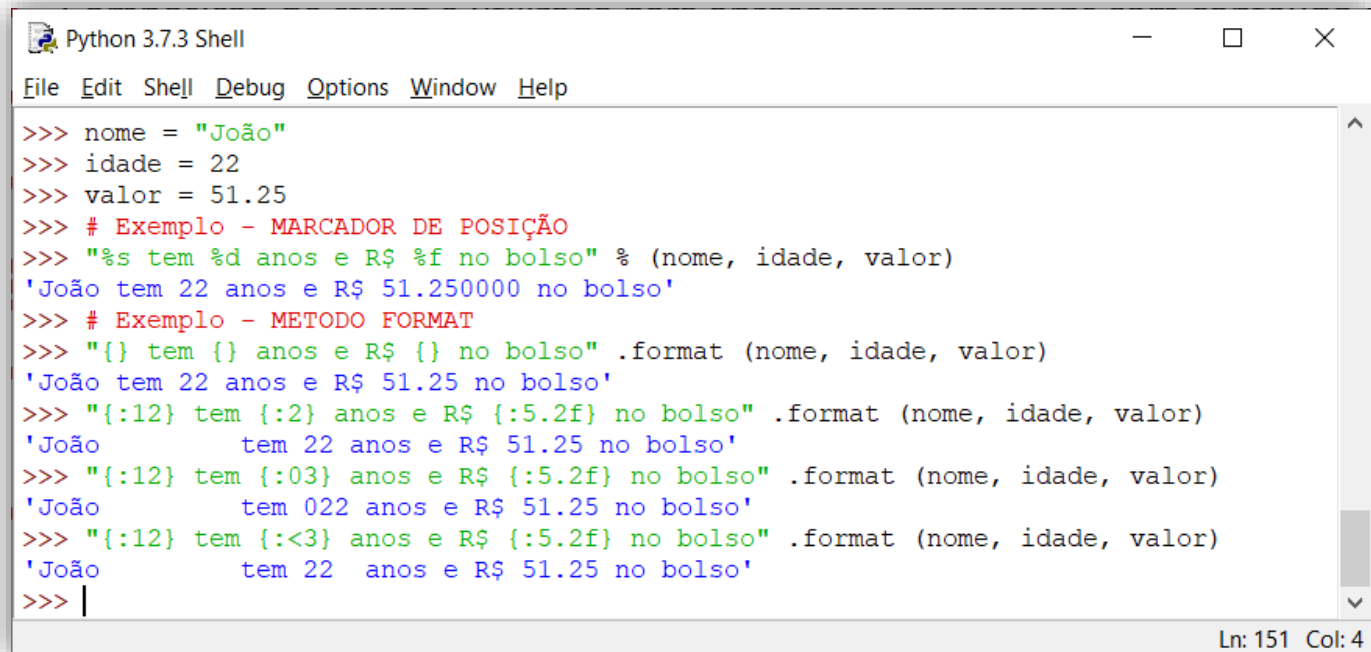
- No Método Format apresenta de forma mais “inteligente” os valores.
- Em resumo, substitui-se o % por **.format** e os %d, %s, %f por {}

TIPOS DE VARIÁVEIS – STRING

FORMATAÇÃO DE STRING NO METODO FORMAT:

- No método de format, para escrever o tamanho da mascaras é utilizado o : internamente nas chaves {}. Para numero inteiro, utiliza-se apenas o : com numero decimal exige o f completando e no caso de alinhamento a esquerda, ao invés de – utiliza o <

EXEMPLO:



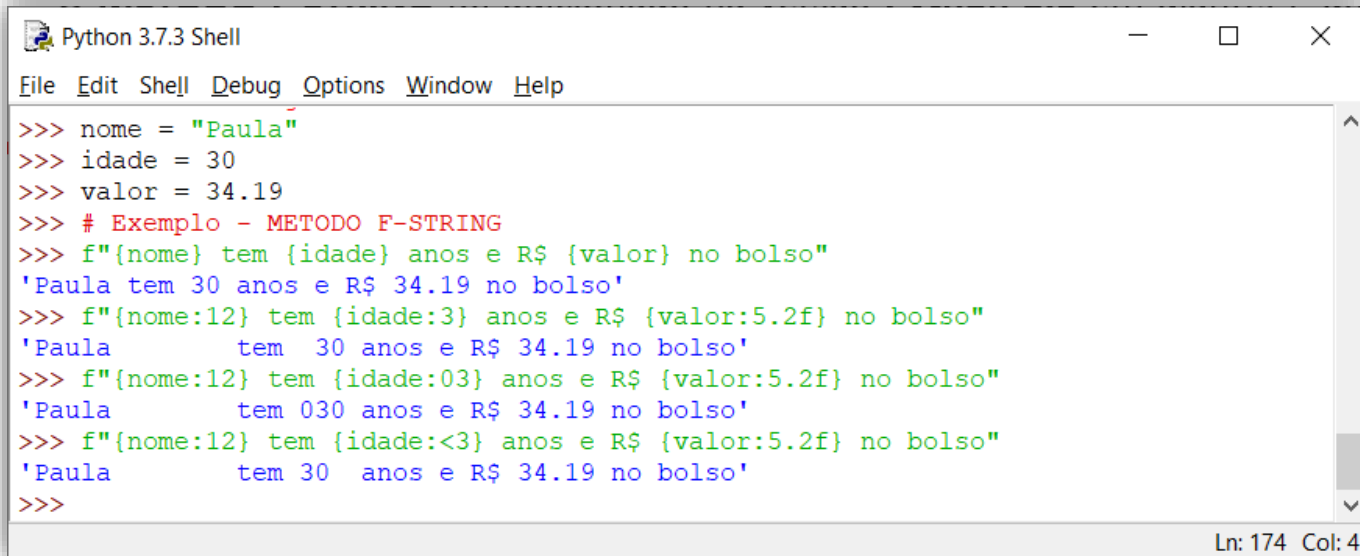
```
Python 3.7.3 Shell
File Edit Shell Debug Options Window Help
>>> nome = "João"
>>> idade = 22
>>> valor = 51.25
>>> # Exemplo - MARCADOR DE POSIÇÃO
>>> "%s tem %d anos e R$ %f no bolso" % (nome, idade, valor)
'João tem 22 anos e R$ 51.250000 no bolso'
>>> # Exemplo - METODO FORMAT
>>> "{} tem {} anos e R$ {} no bolso".format (nome, idade, valor)
'João tem 22 anos e R$ 51.25 no bolso'
>>> "{:12} tem {:2} anos e R$ {:.2f} no bolso".format (nome, idade, valor)
'João          tem 22 anos e R$ 51.25 no bolso'
>>> "{:12} tem {:03} anos e R$ {:.2f} no bolso".format (nome, idade, valor)
'João          tem 022 anos e R$ 51.25 no bolso'
>>> "{:12} tem {:<3} anos e R$ {:.2f} no bolso".format (nome, idade, valor)
'João          tem 22  anos e R$ 51.25 no bolso'
>>> |
```

Ln: 151 Col: 4

TIPOS DE VARIÁVEIS – STRING

OPÇÃO 3 : METODO F-STRING

- O **MÉTODO F-STRING** foi adicionado na versão **Python 3.6** em diante. É uma forma mais moderna e compacta.
- Neste método escreve-se a letra **f** antes de abrir as aspas e escreve-se o nome da variável diretamente na string, entre **{}**.
- A **FORMATAÇÃO** utilizada no **METODO FORMAT** se mantém no **METODO F-STRING**

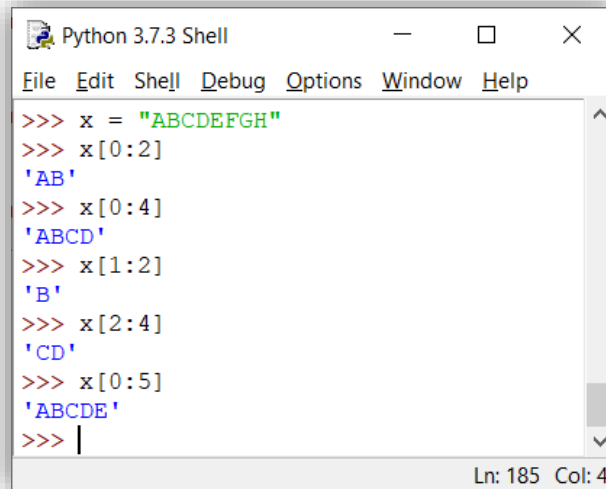
A screenshot of a Python 3.7.3 Shell window. The window has a title bar with the text 'Python 3.7.3 Shell' and standard window controls (minimize, maximize, close). Below the title bar is a menu bar with 'File', 'Edit', 'Shell', 'Debug', 'Options', 'Window', and 'Help'. The main area of the window contains a Python script. The script starts with three lines of variable assignment: 'nome = "Paula"', 'idade = 30', and 'valor = 34.19'. This is followed by a comment line: '# Exemplo - METODO F-STRING'. Then, there are five lines of F-string formatting. The first line is 'f"{nome} tem {idade} anos e R\$ {valor} no bolso"', which outputs 'Paula tem 30 anos e R\$ 34.19 no bolso'. The second line is 'f"{nome:12} tem {idade:3} anos e R\$ {valor:5.2f} no bolso"', which outputs 'Paula tem 30 anos e R\$ 34.19 no bolso'. The third line is 'f"{nome:12} tem {idade:03} anos e R\$ {valor:5.2f} no bolso"', which outputs 'Paula tem 030 anos e R\$ 34.19 no bolso'. The fourth line is 'f"{nome:12} tem {idade:<3} anos e R\$ {valor:5.2f} no bolso"', which outputs 'Paula tem 30 anos e R\$ 34.19 no bolso'. The fifth line is 'f"{nome:12} tem {idade:<3} anos e R\$ {valor:5.2f} no bolso"', which outputs 'Paula tem 30 anos e R\$ 34.19 no bolso'. The window ends with a status bar showing 'Ln: 174 Col: 4'.

The background of the slide is a dark gray gradient. It features a faint, light gray pattern of binary code (0s and 1s) scattered across the top and bottom sections. In the bottom left corner, there is a small, stylized network diagram consisting of several white dots connected by thin white lines, resembling a star or a small cluster. The central part of the slide is a solid dark gray rectangle where the title is located.

FATIAMENTO DE STRING

FATIAMENTO DE STRING

- O **FATIAMENTO DE STRING** é um poderoso recurso do Python, muito utilizado em **DATA SCIENCE** e outros recursos para resolução de problemas.
- O fatiamento funciona com a utilização de dois pontos no índice da string.
- O numero a esquerda dos dois pontos indica a posição de inicio da fatia e o à direita do fim.
- O final da fatia não é incluso na apresentação.



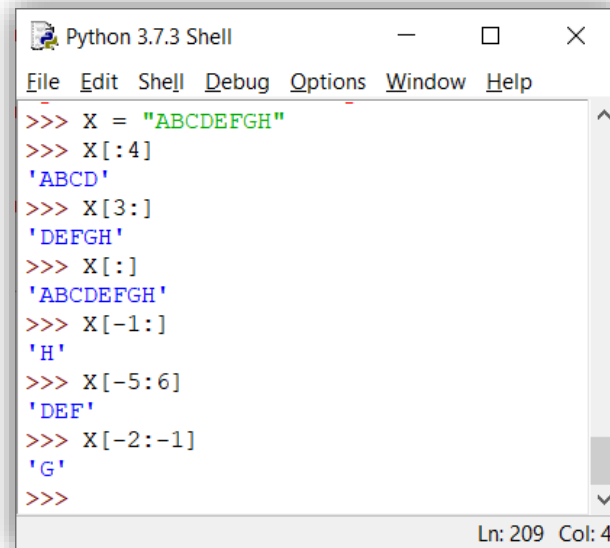
```
Python 3.7.3 Shell
File Edit Shell Debug Options Window Help
>>> x = "ABCDEFGH"
>>> x[0:2]
'AB'
>>> x[0:4]
'ABCD'
>>> x[1:2]
'B'
>>> x[2:4]
'CD'
>>> x[0:5]
'ABCDE'
>>> |
```

Ln: 185 Col: 4

FATIAMENTO DE STRING

VARIAÇÕES DO FATIAMENTO DE STRING:

- **Omitir o numero a esquerda** representa do **inicio até o índice determinado**.
- **Omitir o numero a direita** representa no **índice determinado até o final**.
- **Omitir os números (esquerda e direita)** irá fazer uma copia de **todos os caracteres** da string.
- Utilizar valor negativo para indicar posições a partir da direita (-1 é o ultimo caractere e -2 o penúltimo, etc.).



```
Python 3.7.3 Shell
File Edit Shell Debug Options Window Help
>>> X = "ABCDEFGH"
>>> X[:4]
'ABCD'
>>> X[3:]
'DFGH'
>>> X[:]
'ABCDEFGH'
>>> X[-1:]
'H'
>>> X[-5:6]
'DEF'
>>> X[-2:-1]
'G'
>>>
```

Ln: 209 Col: 4

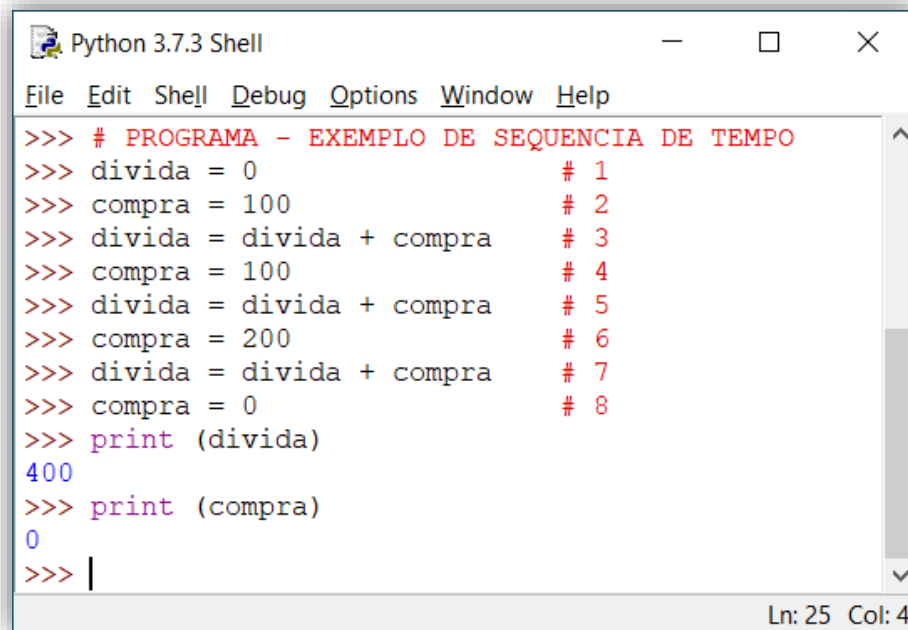
The background of the slide is a dark gray gradient. It features a faint, light gray circular pattern resembling a stylized globe or a series of concentric arcs. Overlaid on this are various binary code elements. In the top left, there is a small box containing the binary sequence '000101'. In the bottom left, another box contains '01001011'. Scattered throughout the background are numerous strings of binary digits (0s and 1s). In the bottom left corner, there is a network diagram consisting of several small white dots connected by thin white lines, forming a star-like or web-like structure.

SEQUENCIA DE TEMPO E RASTREAMENTO

SEQUENCIA DE TEMPO E RASTREAMENTO

SEQUENCIA DE TEMPO

- A execução de um programa é realizado linha por linha no computador (script). Porém alguns cuidados precisam ser tomados quando se adotam variáveis. O conteúdo de uma variável muda constantemente durante a rotina do programa, isso porque, a cada vez que o valor for alterado, o valor anterior é substituído por um valor novo.



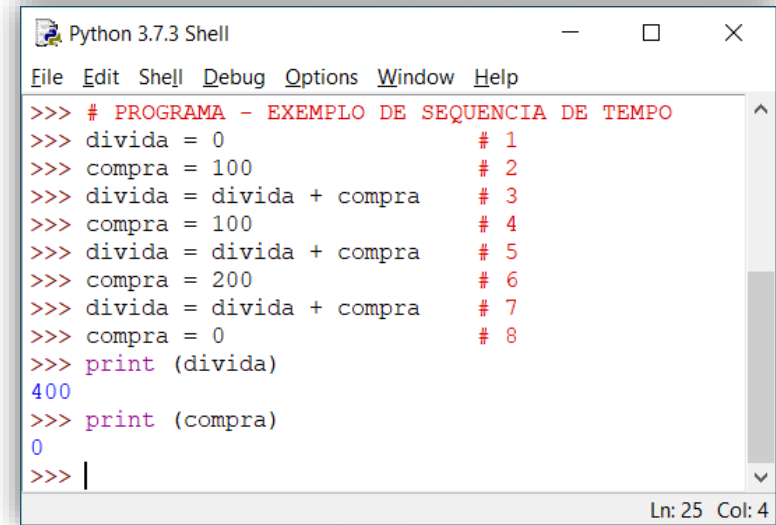
```
Python 3.7.3 Shell
File Edit Shell Debug Options Window Help
>>> # PROGRAMA - EXEMPLO DE SEQUENCIA DE TEMPO
>>> divida = 0 # 1
>>> compra = 100 # 2
>>> divida = divida + compra # 3
>>> compra = 100 # 4
>>> divida = divida + compra # 5
>>> compra = 200 # 6
>>> divida = divida + compra # 7
>>> compra = 0 # 8
>>> print (divida)
400
>>> print (compra)
0
>>> |
```

Ln: 25 Col: 4

SEQUENCIA DE TEMPO E RASTREAMENTO

ANALISE DO PROGRAMA:

- ✓ **#1** – O usuário começa sem divida. Divida é R\$ 0,00.
- ✓ **#2** – Houve uma compra de R\$ 100,00 e o valor da divida continua R\$ 0,00.
- ✓ **#3** – O valor da divida é atualizado para R\$ 100,00.
- ✓ **#4** – Uma nova compra de R\$ 100,00 é realizada e o valor da divida continua R\$ 100,00.
- ✓ **#5** – O valor da divida é atualizado com a divida anterior, adicionado a ultima compra, total R\$ 200,00.
- ✓ **#6** – Uma nova compra de R\$ 200,00 é realizada e o valor da divida continua R\$ 200,00.
- ✓ **#7** – O valor da divida é atualizado com a divida anterior, adicionado a ultima compra, total R\$ 400,00.
- ✓ **#8** – Não houve mais compra (R\$ 0,00). Observe que esse valor não é contabilizado na divida (problema);
- ✓ O valor da divida foi total de R\$ 400,00 e ultima compra foi de R\$ 0,00.

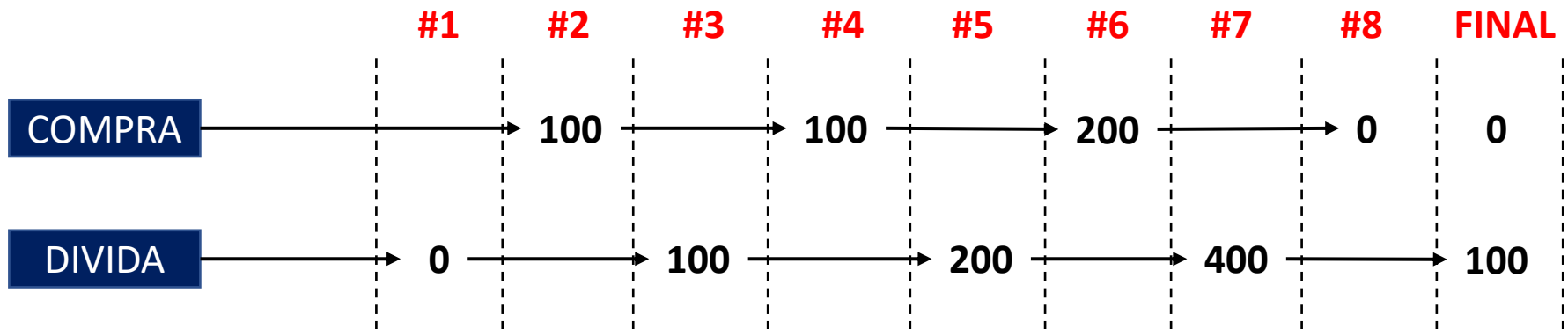


```
Python 3.7.3 Shell
File Edit Shell Debug Options Window Help
>>> # PROGRAMA - EXEMPLO DE SEQUENCIA DE TEMPO
>>> divida = 0 # 1
>>> compra = 100 # 2
>>> divida = divida + compra # 3
>>> compra = 100 # 4
>>> divida = divida + compra # 5
>>> compra = 200 # 6
>>> divida = divida + compra # 7
>>> compra = 0 # 8
>>> print (divida)
400
>>> print (compra)
0
>>> |
```

Ln: 25 Col: 4

SEQUENCIA DE TEMPO E RASTREAMENTO

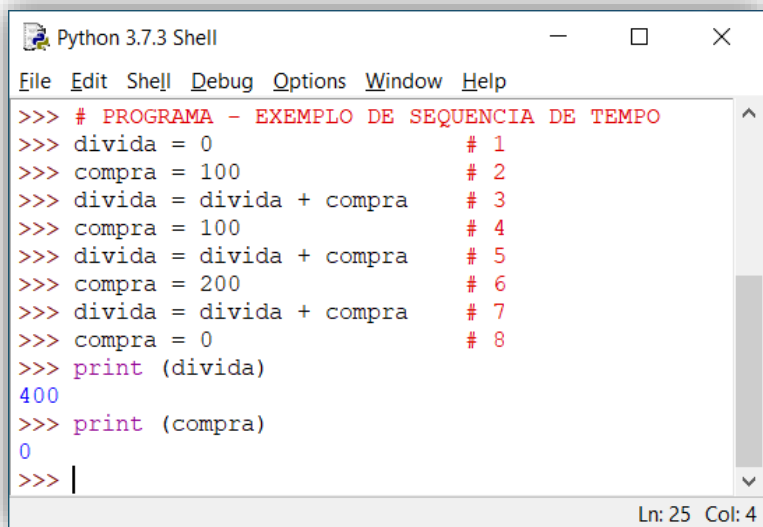
```
Python 3.7.3 Shell
File Edit Shell Debug Options Window Help
>>> # PROGRAMA - EXEMPLO DE SEQUENCIA DE TEMPO
>>> divida = 0 # 1
>>> compra = 100 # 2
>>> divida = divida + compra # 3
>>> compra = 100 # 4
>>> divida = divida + compra # 5
>>> compra = 200 # 6
>>> divida = divida + compra # 7
>>> compra = 0 # 8
>>> print (divida)
400
>>> print (compra)
0
>>> |
```



SEQUENCIA DE TEMPO E RASTREAMENTO

RASTREAMENTO

- Rastreamento é uma técnica na qual ao ler um programa ou testar seu desenvolvimento é simulado valores em uma folha de papel.
- Escreva o nome de suas variáveis. Leia uma linha do programa por vez e escreva o valor atribuído a cada variável folha, na mesma coluna em que escreve o nome da variável. Conforme a variável vai se alterando, escreva o novo valor e risque o anterior em forma de coluna.



```
Python 3.7.3 Shell
File Edit Shell Debug Options Window Help
>>> # PROGRAMA - EXEMPLO DE SEQUENCIA DE TEMPO
>>> divida = 0 # 1
>>> compra = 100 # 2
>>> divida = divida + compra # 3
>>> compra = 100 # 4
>>> divida = divida + compra # 5
>>> compra = 200 # 6
>>> divida = divida + compra # 7
>>> compra = 0 # 8
>>> print (divida)
400
>>> print (compra)
0
>>> |
```

Ln: 25 Col: 4

Tela	Divida	Compra
400	0	100
0	100	100
	200	200
	400	

CREDITOS DO MATERIAL:

Elaborado por:

Prof. Vinicius Heltai

Colaboração de Conteúdo:

Sem colaborador

Ultima atualização: 2023/1

MUITO OBRIGADO!

PROF. VINICIUS HELTAI

vinicius.pacheco3@fatec.sp.gov.br