



Rafael Vieira Coelho

rafaelvc2@gmail.com

PARTE 1

2 - Variáveis e Expressões

2.1 Tipos de Dados

Se você estiver em dúvida sobre qual é o tipo de um determinado valor, o interpretador pode revelar:

```
>>> type("Alô, Mundo!")  
<class 'str'>
```

```
>>> type(17)  
<class 'int'>
```

```
>>> type(3.2)  
<class 'float'>
```

DICAS:

- Não coloque vígulas em números em ponto flutuante!
- Tudo que estiver entre aspas duplas é String.

2.2 Variáveis

Uma variável é um nome que se refere a um valor.

O comando de atribuição cria novas variáveis e dá a elas valores:

```
>>> mensagem = "E aí, Doutor?"  
>>> n = 17  
>>> pi = 3.14159
```

A função print também funciona com variáveis:

```
>>> print (mensagem)  
E aí, Doutor?  
>>> print (n)  
17  
>>> print (pi)  
3.14159
```

2.3 Nomes de variáveis e Palavras reservadas

- Os programadores geralmente escolhem nomes significativos para suas variáveis, pois os nomes documentam para o que a variável é usada.
- Nomes de variáveis podem ser arbitrariamente longos.
- Eles podem conter tanto letras quanto números, mas têm de começar com uma letra.

2.3 Nomes de variáveis e Palavras reservadas

- Embora seja válida a utilização de letras maiúsculas, por convenção, não usamos.
- Maiúsculas e minúsculas são diferentes. Bruno e bruno são variáveis diferentes.
- Tome cuidado ao usar a letra minúscula l e a letra maiúscula O, pois elas podem ser confundidas com os números 1 e 0.

Nomes de variáveis

- Espaços não são permitidos (usa-se underline _ para separar nomes)
- Ele é muito utilizado em nomes compostos, tal como em `meu_nome` ou `preco_do_cha_na_china`.

Nomes de variáveis

- Se você der a uma variável um nome inválido, causará um erro de sintaxe:

```
>>> 76trombones = "grande parada"  
SyntaxError: invalid syntax
```

```
>>> muito$ = 1000000  
SyntaxError: invalid syntax
```

```
>>> class = "Ciencias da Computacao 101"  
SyntaxError: invalid syntax
```

Palavras Reservadas

- Palavras reservadas definem as regras e a estrutura da linguagem e não podem ser usadas como nomes de variáveis.

```
import keyword  
print (keyword.kwlist)
```

and	def	for	is	raise	False
as	del	from	lambda	return	None
assert	elif	global	nonlocal	try	True
break	else	if	not	while	
class	except	import	or	with	
continue	finally	in	pass	yield	

Entendendo Erros em Python

- Escreveremos um código que gera um erro propositadamente.
- Digite o código a seguir, incluindo a palavra **message** com um erro de ortografia, conforme mostrada em **negrito**:

```
message = "Hello Python Crash Course reader!"  
print(mesage)
```

Entendendo Erros em Python

```
message = "Hello Python Crash Course reader!"  
print(mesage)
```

- Eis um exemplo do *traceback* fornecido por Python após o nome da variável ter sido digitado incorretamente por engano:

Traceback (most recent call last):

u File "hello_world.py", line 2, in <module>

v print(mesage)

w NameError: name 'mesage' is not defined

Um traceback é um registro do ponto em que o interpretador se deparou com problemas quando tentou executar seu código.

2.4 Avaliando expressões

Uma expressão é uma combinação de valores, variáveis e operadores. Se você digitar uma expressão na linha de comando, o interpretador avalia e exibe o resultado:

```
>>> 1 + 1  
2
```

Embora expressões contenham valores, variáveis e operadores, nem toda expressão contém todos estes elementos.

Um valor por si só é considerado uma expressão, do mesmo modo que uma variável:

```
>>> 17  
17  
>>> x  
2
```

2.5 Operadores e operandos

Operadores são símbolos especiais que representam computações como adição e multiplicação.

Os valores que o operador usa são chamados operandos.

Todas as expressões seguintes são válidas em Python e seus significados são mais ou menos claros:

```
>>> 2 + 3
```

```
5
```

```
>>> 3 - 2
```

```
1
```

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

```
6
```

```
>>> 3 / 2
```

```
1.5
```

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

```
9
```

```
>>> 3 ** 3
```

```
27
```

```
>>> 10 ** 6
```

```
1000000
```

```
>>> 2 + 3*4
```

```
14
```

```
>>> (2 + 3) * 4
```

```
20
```

DICAS:

**** é o símbolo para potenciação**

// é o símbolo para divisão inteira.

2.6 Operadores e operandos

Você pode ficar surpreso com a divisão.

Observe as seguintes operações:

```
>>> minuto = 59
>>> minuto/60
0.98333333333333328
```

```
>>> minuto = 59
>>> minuto//60
0
```

O valor de minuto é 59 e, em aritmética convencional (/), 59 dividido por 60 é 0,98333. Já a divisão inteira (//) de 59 por 60 é 0.

2.7 Ordem dos operadores

Quando mais de um operador aparece em uma expressão, a ordem de avaliação depende das regras de precedência da matemática.

1. Parênteses têm a mais alta precedência e podem ser usados para forçar uma expressão a ser avaliada na ordem que você quiser. Expressões entre parênteses são avaliadas primeiro, $2 * (3-1)$ é 4, e $(1+1)**(5-2)$ é 8.
2. Exponenciação ou potenciação, assim $2**1+1$ é 3 e não 4.
3. Multiplicação e Divisão (mesma precedência).
4. Adição e Subtração (mesma precedência).

Operadores com a mesma precedência são avaliados da esquerda para a direita.

2.8 Operações com strings

Você não pode executar operações matemáticas em strings, ainda que as strings se pareçam com números.

Para strings, o operador + representa concatenação, que significa juntar os dois operandos ligando-os pelos extremos.

Por exemplo:

```
>>> fruta = "banana"
>>> assada = " com canela"
>>> print (fruta + assada)
banana com canela.
```

O espaço antes da palavra com é parte da string e é necessário para produzir o espaço entre as strings concatenadas.

2.8 Operações com strings

- O operador `*` também funciona com strings; ele realiza repetição.

Por exemplo, `"Legal"*3` é `"LegalLegaLegal"`.

Um dos operadores tem que ser uma string; o outro tem que ser um inteiro.

- **Aspas triplas para textos longos:**

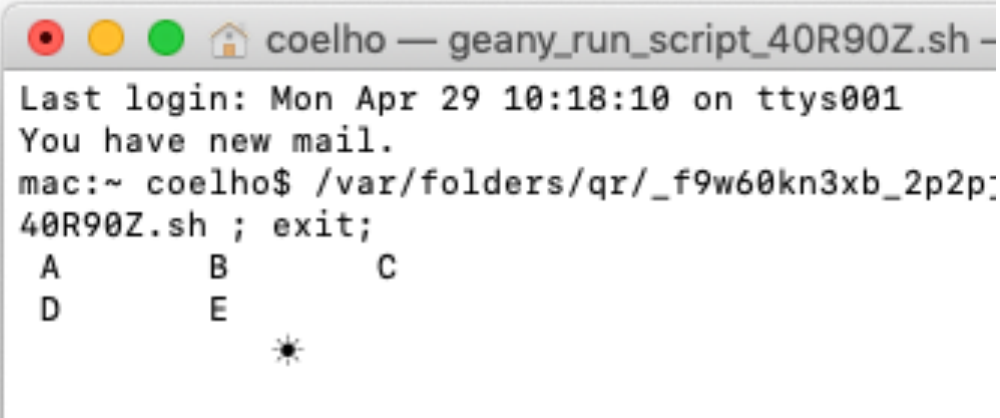
```
b = """uma linha,  
segunda linha"""
```


\\	imprime uma barra ao contrário (invertida) ou <i>backslash</i> .	\
\'	aspas simples	'
\"	aspas duplas	"
\a	chamado de ASCII <i>bell</i> ou <i>beep</i> do sistema. Se houver suporte, aciona um bipe.	
\b	aciona o <i>backspace</i> , ou seja, apaga o caractere anterior	<code>print("py"+"\\b"+"c") = pc</code>
\f	insere um <i>form feed</i> ou quebra de página	
\n	insere uma quebra de linha	
\r	insere um <i>Carriage Return</i> , equivalente ao efeito da tecla <i>Enter</i>	
\u	insere um caractere UNICODE. Deve acompanhar um código com 4 números	<code>print("Isto é um sol: \\u2600") = Isto é um sol: ☀</code>
\t	insere tabulação horizontal	
\v	insere tabulação vertical	

```

1 print(" A \t B \t C \n D \t E \v \u2600")
2

```



coelho — geany_run_script_40R90Z.sh -

Last login: Mon Apr 29 10:18:10 on ttys001
 You have new mail.
 mac:~ coelho\$ /var/folders/qr/_f9w60kn3xb_2p2p:40R90Z.sh ; exit;

































































































```

A      B      C
D      E
      ☀

```

```
>>> print("O meu signo é \u264C")
```

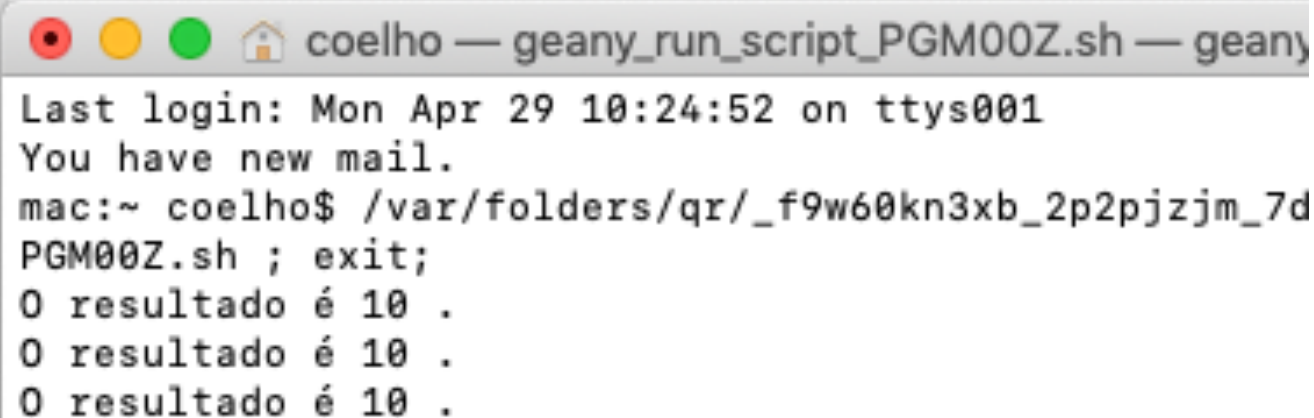
O meu signo é ♈

															
2600	2601	2602	2603	2604	2605	2606	2607	2608	2609	260A	260B	260C	260D	260E	260F
															
2610	2611	2612	2613	2614	2615	2616	2617	2618	2619	261A	261B	261C	261D	261E	261F
															
2620	2621	2622	2623	2624	2625	2626	2627	2628	2629	262A	262B	262C	262D	262E	262F
															
2630	2631	2632	2633	2634	2635	2636	2637	2638	2639	263A	263B	263C	263D	263E	263F
															
2640	2641	2642	2643	2644	2645	2646	2647	2648	2649	264A	264B	264C	264D	264E	264F
															
2650	2651	2652	2653	2654	2655	2656	2657	2658	2659	265A	265B	265C	265D	265E	265F

<https://pythonbay.com>

Formatação de Strings

```
1  
2 print("0 resultado é", 10, ".")  
3  
4 print("0 resultado é %i ." % 10)  
5  
6 print("0 resultado é {}".format(10))  
7  
8
```



The image shows a terminal window titled "coelho — geany_run_script_PGM00Z.sh — geany". The terminal output shows the results of running a script that prints the number 10 using three different string formatting methods: concatenation, the % operator, and the .format() method. The output for each method is "0 resultado é 10 .".

```
Last login: Mon Apr 29 10:24:52 on ttys001  
You have new mail.  
mac:~ coelho$ /var/folders/qr/_f9w60kn3xb_2p2pjzjm_7d  
PGM00Z.sh ; exit;  
0 resultado é 10 .  
0 resultado é 10 .  
0 resultado é 10 .
```

2.9 Composição

Uma das características mais práticas das linguagens de programação é a possibilidade de pegar pequenos blocos e combiná-los numa composição.

Por exemplo, nós sabemos como somar números e sabemos como exibí-los; acontece que podemos fazer as duas coisas ao mesmo tempo:

```
>>> print (17 + 3)  
20
```

O ponto é que qualquer expressão envolvendo números, strings, e variáveis pode ser usada dentro de uma chamada da função print.

Ex:

```
>>> print ("Número de minutos desde a meia-noite: ", hora*60+minuto)
```

2.10 Ler dados literais

Podemos usar a função **input** para que o usuário possa informar dados e armazená-los em variáveis.

O formato típico de um comando leitura é

```
variavel = input("TEXTO QUE APARECE PARA O USUÁRIO")
```

O programa pára e espera pela digitação de algum texto seguido do ENTER. "Prompt" é opcional e pode indicar o que programa deseja. Por exemplo,

```
nome = input("Qual é o seu nome? ")  
print(nome, ", me fale sobre você.")
```

2.10 Ler dados inteiros

Podemos usar a função **input** para que o usuário possa informar dados e armazená-los em variáveis.

O formato típico de um comando leitura é

```
variavel = int(input("Informe um valor:"))
```

```
variavel = variavel + 10
```

```
print(str(variavel), " é o resultado.")
```

2.10 Ler dados em ponto flutuante

Podemos usar a função **input** para que o usuário possa informar dados e armazená-los em variáveis.

O formato típico de um comando leitura é

```
variavel = float(input("Informe um valor:"))
```

```
variavel = variavel + 10
```

```
print(str(variavel), " é o resultado.")
```

Entendendo Erros em Python

- Ao executar o código abaixo, o que aconteceu?

```
a = input("Qual valor você quer somar com 10?")  
b = 10  
print(a+b)
```

```
Qual valor você quer somar com 10?5  
Traceback (most recent call last):  
  File "Exemplo1.py", line 12, in <module>  
    main()  
  File "Exemplo1.py", line 10, in main  
    print(a+b)  
TypeError: must be str, not int
```


- Comentários são um recurso extremamente útil na maioria das linguagens de programação.
- Tudo que você escreveu em seus programas até agora é código Python.
- À medida que seus programas se tornarem mais longos e complicados, você deve acrescentar notas que descrevam a abordagem geral adotada para o problema que você está resolvendo.
- Um comentário permite escrever notas em seus programas em linguagem natural.

Comentários

```
# Diga olá a todos  
print("Hello Python people!")
```

Python ignora a primeira linha e executa a segunda.

```
Hello Python people!
```

2.11 Estrutura de um Programa

Utilizaremos a seguinte estrutura para um programa em Python.

```
def main():  
    # comandos  
    ...  
  
#-----  
# a linha a seguir inicia a execução do programa  
main()
```

2.11 Estrutura de um Programa

Utilizaremos a seguinte estrutura para um programa em Python.

```
1  def main():
2      print('Olá mundo')
3
4  if __name__ == '__main__':
5      main()
```

Ex6_Ola_Mundo.py

Ex7_Exemplo_Main1.py

```
def main():  
    # comandos  
    ...  
  
#-----  
# a linha a seguir inicia a execução do programa  
main()
```

```
1  #!/usr/bin/env python  
2  # coding: utf8  
3  
4  __AUTHOR__ = "Rafael Vieira Coelho"  
5  __DATE__ = "30/04/2019"  
6  
7  def main():  
8      a = int(input("Qual valor você quer somar com 10?"))  
9      b = 10  
10     print(a+b)  
11  
12     main()  
13
```

```
Qual valor você quer somar com 10?5  
15
```

Ex8_Exemplo_Main2.py

```
1  #!/usr/bin/env python
2  # coding: utf8
3
4  __AUTHOR__ = "Rafael Vieira Coelho"
5  __DATE__ = "30/04/2019"
6
7  def main():
8      # a_str e b_str guardam strings
9      a_str = input("Digite o primeiro numero: ")
10     b_str = input("Digite o segundo numero: ")
11
12     # a_int e b_int guardam inteiros
13     a_int = int(a_str) # converte string/texto para inteiro
14     b_int = int(b_str) # converte string/texto para inteiro
15
16     # calcule a soma entre valores que são números inteiros
17     soma = a_int + b_int
18
19     #texto_soma guarda strings
20     texto_soma = str(soma) # converte inteiro para string/texto
21
22     # imprima a soma
23     print("A soma de " + a_str + " + " + str(b_int) + " eh igual a " + texto_soma)
24
25  if __name__ == "__main__":
26     main()
27
```

EXERCÍCIOS

1) Execute os comandos abaixo no interpretador (nesta sequencia):

```
width = 17
```

```
height = 12.0
```

```
delimiter = '.'
```

```
width/2
```

```
width/2.0
```

```
height/3
```

```
1 + 2 * 5
```

```
delimiter * 5
```

Quais são as saídas?

EXERCÍCIOS

2) Dados dois inteiros a e b, deve-se calcular a sua soma. Qual alteração no código abaixo deveria ser realizado para que isto ocorresse?

```
1 def main():
2     a = 3
3     b = 4
4     soma = a + b
5     print("A soma de a + b eh igual a soma")
6
7 main()
8
9
```

Observe que tudo entre aspas (") define um texto ou string. Mas queremos imprimir o valor das variáveis e não o nome delas.

EXERCÍCIOS

3) Sabendo que o operador `+` concatena dois strings e que existem funções para converter string-int e int-string, execute o programa abaixo e veja como ele se comporta. Como você corrigiria o erro?

```
Exemplo2.py x
1  #!/usr/bin/env python
2  # coding: utf8
3
4  def main():
5      # a_str e b_str guardam strings
6      a_str = input("Digite o primeiro numero: ")
7      b_str = input("Digite o segundo numero: ")
8
9      # a_int e b_int guardam inteiros
10     a_int = int(a_str) # converte string/texto para inteiro
11     b_int = int(b_str) # converte string/texto para inteiro
12
13     # calcule a soma entre valores que são números inteiros
14     soma = a_int + b_int
15
16     #texto_soma guarda strings
17     texto_soma = str(soma) # converte inteiro para string/texto
18
19     # imprima a soma
20     print("A soma de " + a_int + " + " + b_int + " eh igual a " + texto_soma)
21
22     main()
23
```


EXERCÍCIOS

4) Armazene uma mensagem em uma variável e, em seguida, exiba essa mensagem. Então altere o valor de sua variável para uma nova mensagem e mostre essa nova mensagem.

5) Crie um programa no qual o usuário pode escolher qual operação ele quer fazer (+, -, * ou /) e digitar dois números. Como saída do programa, deve ser mostrado o resultado. Lembre-se que não é possível dividir um número por zero.