Comandos Básicos

Programação Estruturada de Computadores (PEC)
Professor: Ritomar Torquato

1

Comandos Básicos

Objetivo: Variáveis; Operadores aritméticos; Constantes; Entrada e saída; Conversão de tipos e formatação de strings; Comentários; Teste de mesa;

2

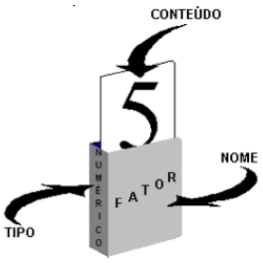
Variáveis

- São espaços identificados na **memória** para armazenar um dado de um determinado tipo, previamente especificados.

3

Variáveis

- Espaço na memória para armazenar um conteúdo.
  - Identificador
  - Tipo de dado
  - Conteúdo



Python tem **tipagem dinâmica**, isso significa que o tipo na caixa muda de acordo com o seu conteúdo.

4

Variáveis

- Como a memória é organizada?

Endereço	Dado	Endereço	Dado	Endereço	Dado
0000		0008		0016	
0001		0009		0017	
0002		0010		0018	
0003		0011		0019	
0004		0012 ←	8 bits	→ 0020	
0005		0013		0021	
0006		0014		0022	
0007		0015		...	

Qual o maior valor inteiro que pode ser armazenado em cada espaço?

5

Variáveis

```
nome = conteúdo
```

Lê-se:
[variável] recebe [conteúdo].

```
fator = 5
```

Lê-se:
fator recebe cinco.

Endereço	Dado	
...	?	
0065	?	
0066	5	Fator
0067		
0068		
0069		
0070	?	
...	?	

6

Principais Operadores

Operador	Função	Exemplo	Resultado
=	Atribuição (recebe)	n = 10	10
+	Adição	n = 1 + 2	3
-	Subtração	n = 1 – 2	-1
*	Multiplicação	n = 2 * 3	6
/	Divisão (Real)	n = 5 / 2	2.5
//	Divisão (Inteira)	n = 5 // 2	2
**	Exponenciação	n = 5 ** 2	25
%	Módulo (Resto da Divisão)	n = 5 % 2	1

7

Constantes

- Uma CONSTANTE é uma variável especial que não altera seu valor durante todo o programa.

Python não permite declarar CONSTANTES

Convenciona-se uma variável com TODOS CARACTERES MAIÚSCULOS uma constante.

```
# A variável PI é considerada uma CONSTANTE,
# seu valor NÃO deve alterar após declarada.
PI = 3.1415
```

8

Comando de Entrada

- **input()**: função que recebe informações digitadas pelo teclado sempre do tipo string.

```
>>> nome = input("Nome: ")
Nome: Maria
>>> type(nome)
<class 'str'>
>>> idade = input("Idade: ")
Idade: 20
>>> type(idade)
<class 'str'>
```

9

Comando de Entrada

- **input()**: função que recebe informações digitadas pelo teclado sempre do tipo string.

```
>>> idade = idade + 5
Traceback (most recent call last):
  File "<pyshell#59>", line 1, in <module>
    idade = idade + 5
TypeError: Can't convert 'int' object to str implicitly
>>> idade = idade + "5"
>>> print(idade)
205
>>> type(idade)
<class 'str'>
```

10

Conversão Explícita

```
>>> idade = input("Idade: ")
Idade: 20
>>> type(idade)
<class 'str'>
>>> idade = int(idade)
>>> type(idade)
<class 'int'>
>>> idade = int(input("Idade: "))
Idade: 25
>>> type(idade)
<class 'int'>
```

11

Conversão Explícita

```
>>> str(10)
'10'
>>> bool('10')
True
>>> int('10')
10
>>> float('10')
10.0
```

O número zero (0) ou uma string vazia ('') são convertidos para o valor boleano **False**. Qualquer outro valor é convertido para **True**.

12

Conversão Explícita

```
>>> preco = input('Digite o preço: ')
Digite o preço: 9.99
>>> type(preco)
<class 'str'>
>>> preco = float(preco)
>>> type(preco)
<class 'float'>
>>> preco = float(input('Digite o preço: '))
Digite o preço: 9.99
>>> type(preco)
<class 'float'>
```

13

Comando de Saída

- **print()**: Exibe uma informação na tela. No Python 3 deixou de fazer parte da linguagem (keyword) e tornou-se uma **função**. Sendo assim, é obrigatório o uso de parênteses.

```
#Python 2
print "Alô Mundo!"

#Python 3
print ("Alô Mundo!")
```

14

Comando de Saída

- **print()**: é possível imprimir vários valores separando-os por vírgula. Strings são impressas com 'aspas simples' ou "duplas".

```
>>> a = b = c = 42
>>> print(a, b, c)
42 42 42
```

Automaticamente é adicionado um espaço em branco entre os valores impressos na tela.

```
>>> print('Alô,', 'mundo!')
Alô, mundo!
>>> print('"Esse texto está entre aspas duplas"')
"Esse texto está entre aspas duplas"
>>> print("'Esse entre aspas simples'")
'Esse entre aspas simples'
```

15

Formando saída de dados

- Interpolação de Strings

%d – números inteiros
%s – strings
%f – números reais

```
>>> x = 40
>>> y = 2
>>> print('O valor de x é %d e o de y é %d.' % (x, y))
O valor de x é 40 e o de y é 2.
```

```
>>> n = 42
>>> print('n = %d' % n)
n = 42
```

Quando houver apenas um valor para ser interpolado o (parênteses) pode ser omitido.

```
>>> nome = input('Digite seu nome: ')
Digite seu nome: João
>>> print('Oi, %s.' % nome)
Oi, João.
```

16

Formando saída de dados

- Interpolação de Strings

```
>>> print("%8s" % 'IFPI')
    IFPI
```

O 8 diz que serão 8 espaço na tela

```
>>> print("%8.1f - 10%% = %8.1f" % (6.99, 6.99 * 0.90))
     7.0 - 10% =      6.3
>>> print("%8.2f - 10%% = %8.2f" % (6.99, 6.99 * 0.90))
    6.99 - 10% =     6.29
>>> print("%8.3f - 10%% = %8.3f" % (6.99, 6.99 * 0.90))
   6.990 - 10% =    6.291
>>> print("%8.4f - 10%% = %8.4f" % (6.99, 6.99 * 0.90))
  6.9900 - 10% =   6.2910
```

%x.yf indica que devem ser ocupados x espaços na tela com y casas decimais na interpolação de números reais (ponto flutuante).

%% indica que deve ser impresso o caractere %

17

Formando saída de dados

Embora os exemplos mostrados usem a função de saída, a formatação de uma string independe da função **print()**.

```
>>> "%d + %d = %d" % (1, 2, 3)
'1 + 2 = 3'
```

18

Formando saída de dados

- O método format (a partir da versão 3.1):

```
>>> '{0}, {1}, {2}'.format('a', 'b', 'c')
'a, b, c'
>>> '{}, {}, {}'.format('a', 'b', 'c')
'a, b, c'
>>> '{2}, {1}, {0}'.format('a', 'b', 'c')
'c, b, a'
>>> '{a}, {b}, {c}'.format(a='a', b='b', c='c')
'a, b, c'
>>> print('{}, {}, {}'.format('a', 'b', 'c'))
a, b, c
```

19

Formando saída de dados

- O método format (a partir da versão 3.1):

```
>>> '{:5.2f}'.format(3.14159)
' 3.14'
>>>
>>> votos_sim = 42_572_654
>>> votos_nao = 43_132_495
>>> porcentagem = votos_sim / (votos_sim + votos_nao)
>>> '{:9} votos SIM {:2.2%}'.format(votos_sim, porcentagem)
' 42572654 votos SIM 49.67%'
>>>
```

Também é possível fazer a formatação de números ponto flutuante ou percentuais usando dois pontos (:)

20

Formando saída de dados

- Strings literais formatadas ou (f-string) (a partir da versão 3.6):

```
>>> a = 'a'
>>> b = 'b'
>>> c = 'c'
>>> f'{a}, {b}, {c}'
'a, b, c'
>>> print(f'{a}, {b}, {c}')
a, b, c
>>> x = 40
>>> y = 2
>>> print(f'O valor de x é {x} e o valor de y é {y}')
O valor de x é 40 e o valor de y é 2
```

Permite incluir uma expressões dentro de uma string escrevendo entre chaves { }

21

Formando saída de dados

- Strings literais formatadas ou (f-string) (a partir da versão 3.6):

```
>>> n = 42
>>> print(f'n = {n}')
n = 42
>>> valor = 6.99
>>> print(f'{valor:8.1f}')
   7.0
>>> import math
>>> print(f'Valor aproximado de PI: {math.pi:.3f}')
Valor aproximado de PI: 3.142
```

É possível incluir um formato após a expressão usando dois pontos (:)

https://docs.python.org/pt-br/3/tutorial/inputoutput.html

22

Comentários no Python

- Comentários de uma linha

```
print ("Aqui não é comentário.") #Aqui é um comentário de linha

#Aqui também é um comentário de linha
```

- Comentários de várias linhas (DocStrings)

```
"""
    Este é um comentário
    com válias linhas ou DocStrings.
"""
```

Não iremos detalhar DocStrings, por hora basta saber que podem ser usadas como comentários de multi-linhas.

23

Comentários no Python

- Uma boa prática para quem está começando programar é escrever uma linha de comentário acima de cada comando para explicar o algoritmo. Por exemplo:

```
1  # Faz a leitura de um texto qualquer pelo teclado.
2  texto = input("Digite um texto: ")
3  # Imprime 3 vezes o texto lido na tela.
4  print(texto * 3)
```

24

Teste de mesa

Verificar se um algoritmo ou programa chega ao resultado esperado através do rastreamento e simulação de valores.

Simular a execução de um algoritmo sem utilizar o computador, permitindo acompanhar o raciocínio lógico da resolução do problema.

Prefira usar papel e lápis.

25

Teste de mesa

- Sugestões para realização:
  - **Numere** todas as linhas do algoritmo;
  - **Identifique** todas as **variáveis** do no início;
  - Crie uma tabela para representar a **memória** do computador, com colunas para cada variável identificada;
  - Crie uma área para representar a **tela** do computador. Pode ser uma coluna da tabela
  - Represente a linha de execução (entre parênteses).

26

Teste de mesa

```
1 nome = input("Digite seu nome: ")
2 sobrenome = input("Digite seu sobrenome: ")
3 print(nome + ' ' + sobrenome)
```

Memória		Tela
nome	**sobrenome**	(1) Digite seu nome: Nilo ↵
(1) "Nilo"	(2) "Coutinho"	(2) Digite seu sobrenome: Coutinho↵
		(3) Nilo Coutinho

27

```
1 print("Soma de dois números inteiros.")
2 num1 = input("Digite o primeiro número: ")
3 num1 = int(num1)
4
5 num2 = input("Digite o segundo número: ")
6 num2 = int(num2)
7
8 soma = num1 + num2
9 print(f'A soma de {num1} com {num2} é igual a {soma}')
```

Memória			Tela
num1	**num2**	**soma**	(1) Soma de dois números inteiros.
(2) "30"	(5) "12"	(8) 42	(2) Digite o primeiro número: 30 ↵
(3) 30	(6) 12		(5) Digite o segundo número: 12 ↵
			(9) A soma de 30 com 12 é igual a 42

28

```
1 print("Soma do dobro de dois números inteiros.")
2 num1 = int(input("Digite o primeiro número: "))
3 num1 = num1 * 2
4
5 num2 = int(input("Digite o segundo número: "))
6 num2 = num2 * 2
7
8 soma = num1 + num2
9 print(f'A soma de {num1} com {num2} é igual a {soma}')
```

Memória			Tela
num1	**num2**	**soma**	(1) Soma do dobro de dois números inteiros.
(2) 15	(5) 6	(8) 42	(2) Digite o primeiro número: 15 ↵
(3) 30	(6) 12		(5) Digite o segundo número: 6 ↵
			(9) A soma de 30 com 12 é igual a 42

29