MC102 - Algoritmos e Programação de Computadores

Turma Z - Segundo Semestre de 2019

MC102 - Algoritmos e Programação de Computadores

Turma Z - Segundo Semestre de 2019

URL: ic.unicamp.br/~antonio.guimaraes/mc102/

Aula passada

O que vimos?

Aula de Hoje

- Entrada e saída (input e print)
- Conversão de tipos de dados
- Expressões/operações aritméticas, relacionais e lógicas.

Aplicações no terminal

- MS-DOS https://www.pcjs.org/disks/pcx86/dos/microsoft/3.20/
- WORD no MS-DOS: https://www.pcjs.org/disks/pcx86/apps/microsoft/word/1.15/
- Windows 1.0: https://www.pcjs.org/disks/pcx86/windows/1.00/
- Linux: https://bellard.org/jslinux/vm.html?url=https://bellard.org/jslinux/buildroot-x86.cfg

A partir desse slide, os slides com fundo cinza claro são parte do material desenvolvido pela professora Sandra Avila e disponível em http://www.ic.unicamp.br/~sandra/

A Função print()

Escrevendo na Tela: `print()`

- Para imprimir um texto, utilizamos o comando **`print()`**.
- O texto pode ser um literal do tipo **string**.

```
>>> print(“De novo isso?”)
De novo isso?
```

- No meio da **string** pode-se incluir caracteres de formatação especiais.
- O símbolo especial `\n` é responsável por pular uma linha na saída.

```
>>> print(“De novo isso? \n Olá Pessoal!”)
De novo isso?
Olá Pessoal!
```

Escrevendo o Conteúdo de uma Variável na Tela

- Podemos imprimir, além de texto puro, o conteúdo de uma variável utilizando o comando **`print()`**.
- Separamos múltiplos argumentos a serem impressos com uma vírgula.

```
>>> a = 10
>>> print(“A variável contém o valor”, a)
A variável contém o valor 10
```

Escrevendo o Conteúdo de uma Variável na Tela

- A impressão com múltiplos argumentos inclui um **espaço extra** entre cada argumento.

```
>>> a = 10
>>> b = 3.14
>>> print(“a contém o valor”, a, “e b contém o valor”, b)
a contém o valor 10 e b contém o valor 3.14
>>> print(“a contém o valor ”, a, “ e b contém o valor ”, b)
a contém o valor  10  e b contém o valor  3.14
```

Escrevendo o Conteúdo de uma Variável na Tela

- Podemos converter todos os valores em strings e usar o **operador +** para concatenar strings de forma a imprimir sem estes espaços:

```
>>> a = 10
>>> b = 3.14
>>> print(“a contém o valor” + str(a) + “e b contém o valor” + str(b))
a contém o valor10e b contém o valor3.14
>>> print(“a contém o valor ” + str(a) + “ e b contém o valor ” + str(b))
a contém o valor 10 e b contém o valor 3.14
```

Formatos Ponto Flutuante

- Podemos especificar o número de casas decimais que deve ser impresso em um número ponto flutuante usando **%.Nf**, onde N especifica o número de casas decimais.

```
>>> pi = 3.1415
>>> r = 7
>>> area = pi * r * r
>>> print(“Área do círculo de raio %.2f ” %r + “é: %.2f” %area)
Área do círculo de raio 7.00 é: 153.93
>>> print(“Área do círculo de raio ” + str(r) + “é: ” + str(area))
Área do círculo de raio 7 é: 153.9335
```

Imprimindo sem Pula Linha

- A função **`print()`** sempre pula uma linha ao final da impressão.
- Se você não quiser que pule uma linha, inclua o argumento **end=' '** no **`print()`**.

```
>>> print(“3, ”, end=“”)
>>> print(“4, ”, end=“”)
>>> print(“5 ”, end=“”)
3, 4, 5
```

A Função input()

A Função `input()`

- Realiza a leitura de dados a partir do teclado.
- Aguarda que o usuário digite um valor e atribui o valor digitado a uma variável.
- **Todos** os dados lidos são do tipo string.

```
>>> print(“Digite um número: ”)
>>> numero = input()
>>> print(“O número digitado é: ” + numero)
```

A Função `input()`

- Podemos converter uma string lida do teclado em um número inteiro usando a função **int()**.

```
>>> print(“Digite um número: ”)
>>> numero = int(input())
>>> numero = numero * 10
>>> print(“O número digitado vezes 10 é: ”, numero)
```

A Função `input()`

- Podemos fazer o mesmo para números ponto flutuante usando a função **float()**.

```
>>> print(“Digite um número: ”)
>>> numero = float(input())
>>> numero = numero * 10
>>> print(“O número digitado vezes 10 é %.2f ” %numero)
```

A Função `input()`

- Nos dois exemplos anteriores é esperado que o usuário digite um número.
- Se o usuário digitar um texto não numerico o programa encerrará com um erro de execução.

```
>>> print("Digite um número: ")
Digite um número:
>>> numero = float(input())
mc102
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: could not convert string to float: 'mc102'
```

A Função `input()`

- O programa abaixo lê dois números e imprime a soma destes.
- Perceba que podemos incluir um texto a ser impresso diretamente no comando `input()`.

```
>>> numero1 = float(input("Digite um número: "))
>>> numero2 = float(input("Digite um número: "))
>>> print(“A soma dos números é: %.2f” %(numero1 + numero2))
```

Expressões

Expressões

- Já vimos que constantes e variáveis são expressões.
- Uma expressão também pode ser um conjunto de operações aritméticas, lógicas ou relacionais utilizadas para fazer “cálculos” sobre os valores das variáveis.
  - Exemplo: a + b

Expressões Aritméticas

- Os operadores aritméticos são: +, –, *, /, //, %, **

- **Adição**: expressão + expressão
- **Subtração**: expressão – expressão
- **Multiplicação**: expressão * expressão

```
>>> 30 + 5
35
>>> 30 - 5
25
>> 30 * 5
150
```

Expressões Aritméticas

- **Divisão**: expressão / expressão
  - O resultado é sempre um número ponto flutuante.
- **Divisão**: expressão // expressão
  - Se os operandos forem inteiros, a divisão é inteira. Se um deles for ponto flutuante faz uma divisão truncada.

```
>>> 5 / 2
2.5
>>> 5 // 2
2
>>> 5 // 2.0
2.0
```

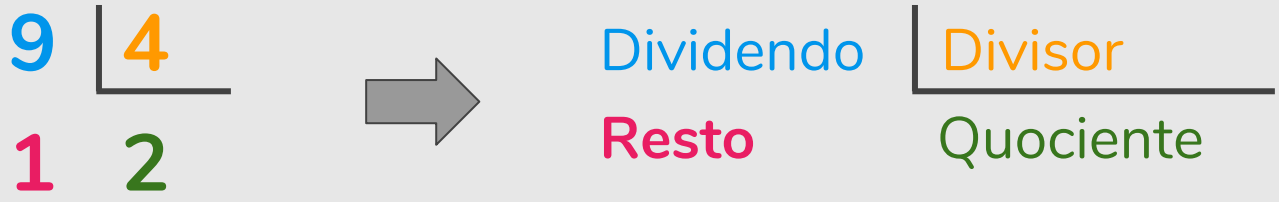
Expressões Aritméticas

- **Exponenciação (potenciação)**: expressão `**` expressão
  - Calcula o valor da expressão à esquerda elevado ao valor da expressão à direita.
  - $a^n = a \times a \times a \times \ldots \times a$ (n vezes)

```
>>> 2 ** 4
16
>>> 2.2 ** 4
23.425600000000006
```

Expressões Aritméticas

- **Resto da Divisão**: expressão % expressão
  - Calcula o resto da divisão inteira de duas expressões.



Dividendo = Divisor * Quociente + **Resto**

Expressões Aritméticas

- **Resto da Divisão**: expressão % expressão
  - Calcula o resto da divisão inteira de duas expressões.

```
>>> 5 % 2
1
>>> 9 % 7
2
>>> 2 % 5
2
>>> 4 % 2
0
```

Expressões Aritméticas

- Exemplo: Converter segundos em horas, minutos e segundos.
  - 87426 segundos = horas, minutos e segundos?

```
>>> segundos_str = input(“Por favor, digite o número de segundos que deseja converter: ”)
>>> total_segundos = int(segundos_str)
>>> horas = total_segundos // 3600
>>> segundos_restantes = total_segundos % 3600
>>> minutos =  segundos_restantes // 60
>>> segundos_restantes_final = segundos_restantes % 60
>>> print(“Horas =”, horas, “minutos =”, minutos, “segundos =” , segundos_restantes_final)
```

Expressões

- As expressões aritméticas (e todas as expressões) operam sobre outras expressões.
- É possível compor expressões complexas como por exemplo
  a = b * ((2 / c) + (9 % d * 8))

```
>>> 5 + 10 % 3
6
>>> 5 * 10 % 3
2
```

Precedência

- Precedência é a **ordem** na qual os operadores serão avaliados quando o programa for executado.
- Em Python, os operadores são avaliados na seguinte ordem:
  1. **
  2. *, /, //, na ordem em aparecerem na expressão
  3. %
  4. +, -, na ordem em aparecerem na expressão
- Exemplo: 8 + 10 * 6 é igual a 68

Precedência

- Operadores com a mesma precedência são executados da esquerda para a direita.
  - 6-3+2 é igual a ?
    - Esquerda para direita: 6-3 é igual a **3**, **3**+2 é igual a 5
    - Direita para esquerda: 3+2 é igual a **5**, 6-**5** é igual a 1

- **Atenção**: Uma exceção é o operador exponenciação **.
  - 2 ** 3 ** 2 é igual a ?
    - Esquerda para esquerda: 2 ** 3 é igual a **8**, **8** ** 2 é igual a 64
    - Direita para esquerda: 3 ** 2 é igual a **9**, 2 ** **9** é igual a 512

Alterando a Precedência

- **(expressão)** também é uma expressão, que calcula o resultado da expressão dentro dos parênteses, para só então calcular o resultado das outras expressões.
  - 5 + 10 % 3 é igual a 6
  - (5 + 10) % 3 é igual a 0
- Você pode usar quantos parênteses desejar dentro de uma expressão.
- Use sempre parênteses em expressões para deixar claro em qual ordem a expressão é avaliada!

Exercícios

- https://panda.ime.usp.br/pensepy/static/pensepy/02-Conceitos/conceitos.html#exercicios

Tipo bool

- Em Python o tipo **bool** especifica os valores booleanos falso (`False`) e verdadeiro (`True`).
- Podemos criar variáveis associadas a booleanos, mas o uso mais comum é na verificação de resultados de expressões relacionais e lógicas.

```
a = True
print(type(a))
<class 'bool'>
```

Exercício

Problema: Sabendo que eu comprei **ou não** um chocolate por X reais. Quanto eu gastei?

Entrada:

- Primeira linha: valor em reais do chocolate. Exemplo: 10.59
- Segunda linha: “True” se eu comprei ou “False” se eu não comprei

Saída: Você gastou N reais.

Expressões

- Já vimos que constantes e variáveis são expressões.

```
a = 10
b = 50
a = b
```

- Vimos também que operações aritméticas também são expressões.

```
a = 2 * 2
a = 10 / 3
a = a + 1
```

Expressões Relacionais

Expressões Relacionais

- Expressões relacionais são aquelas que realizam uma **comparação** entre duas expressões e retornam
  - `False`, se o resultado é falso.
  - `True`, se o resultado é verdadeiro.

Operadores Relacionais

- Os operadores relacionais da linguagem Python são:
  - `==` : igualdade
  - `!=` : diferente
  - `>` : maior que
  - `<` : menor que
  - `>=` : maior ou igual que
  - `<=` : menor ou igual que

Expressões Relacionais

- expressão == expressão : Retorna verdadeiro quando as expressões forem iguais.

```
9 == 9
True
9 == 10
False
```

- expressão != expressão : Retorna verdadeiro quando as expressões forem diferentes.

```
9 != 9
False
9 != 10
True
```

Expressões Relacionais

- expressão > expressão : Retorna verdadeiro quando a expressão da esquerda tiver valor maior que a expressão da direita.

```
9 > 5
True
```

- expressão < expressão : Retorna verdadeiro quando a expressão da esquerda tiver valor menor que a expressão da direita.

```
9 < 5
False
```

Expressões Relacionais

- expressão >= expressão : Retorna verdadeiro quando a expressão da esquerda tiver valor maior ou igual que a expressão da direita.

```
9 >= 5
True
```

- expressão <= expressão : Retorna verdadeiro quando a expressão da esquerda tiver valor menor ou igual que a expressão da direita.

```
9 <= 5
False
```

Expressões Relacionais

- Quais das seguintes opções é uma expressão booleana?
  a. **True**
  b. **3 == 4**
  c. 3 + 4
  d. **3 + 4 == 7**
  e. “False”

```
True          # sim, é uma expressão booleana
True
3 == 4        # sim, é uma expressão booleana
False
3 + 4         # não é uma expressão booleana
7
3 + 4 == 7    # sim, é uma expressão booleana
True
"False"       # não é uma expressão booleana
'False'
```

Expressões Relacionais

```
a = 3
b = 4
c = a < b       # c recebe o valor da comparação a < b
d = a > b       # d recebe o valor da comparação a > b
e = a == b      # e recebe o valor da comparação a == b

print("Valor de c:", c)
print("Valor de d:", d)
print("Valor de e:", e)
```

```
Valor de c: True
Valor de d: False
Valor de e: False
```

Expressões Lógicas

Expressões Lógicas

- Expressões lógicas são aquelas que realizam uma operação lógica (**ou**, **e**, **não**, etc...) e retornam `True` ou `False` (como as expressões relacionais).

- Na linguagem Python temos os seguintes operadores lógicos:
  - **`and`** : operador E
  - **`or`**: operador OU
  - **`not`**: operador NÃO

Expressões Lógicas

- expressão **`and`** expressão : Retorna verdadeiro quando **ambas** as expressões são verdadeiras. Sua tabela verdade é:

Op1	Op2	Op1 `and` Op2
V	V	V
V	F	F
F	V	F
F	F	F

Qual o resultado da expressão lógica abaixo?

```
a = 0
b = 0
( a == 0 and b == 0 )
True
```

Expressões Lógicas

- expressão **`or`** expressão : Retorna verdadeiro quando **pelo menos uma** das expressões é verdadeira. Sua tabela verdade é:

Op1	Op2	Op1 `or` Op2
V	V	V
V	F	V
F	V	V
F	F	F

Qual o resultado da expressão lógica abaixo?

```
a = 0
b = 0
( a == 1 or b == 0 )
True
```

Expressões Lógicas

- **`not`** expressão : Retorna verdadeiro quando a expressão é falsa e vice-versa. Sua tabela verdade é:

Op1	`not` Op1
V	F
F	V

Qual o resultado da expressão lógica abaixo?

```
a = 0
b = 0
not(a != b)
False
```

Expressões Lógicas

- O que será impresso pelo programa?

```
print(8 > 9 and 10 != 2)
print(14 > 100 or 2 > 1)
print(not(14 > 100) and not(1 > 2))
```

```
False
True
True
```

Expressões Lógicas

- Qual é a expressão correta em Python para verificar se um número armazenado na variável x está entre 0 e 5? (múltiplas respostas)
  a. **0 < x < 5**
  b. x > 0 or x < 5
  c. **x > 0 and x < 5**
  d. x > 0 and < 5

```
x = 6
0 < x < 5
False
x > 0 or x < 5
True
x > 0 and x < 5
False
x > 0 and < 5
  File "<stdin>", line 1
    x > 0 and < 5
              ^
```

Precedência de Operadores

Nível	Categoria	Operadores
7 (alto)	exponenciação	`**`
6	multiplicação	`*`, `/`, `//`, `%`
5	adição	`+`, `-`
4	relacional	`==`, `!=`, `<=`, `>=`, `>`, `<`
3	lógico	`not`
2	lógico	`and`
1 (baixo)	lógico	`or`

Exercício

Inverter a ordem de dois números utilizando o seguinte código:

```
a = int(input())
b = int(input())
... Insira operações aritméticas aqui ...
print(a, b)
```

Exemplo

Entrada:

```
1
5
```

Saída:

```
5 1
```

Regras:

- Não utilize outras variáveis além de *a* e *b*
- Não modifique o código já escrito.
- Utilize apenas operações aritméticas e atribuições simples.

Referências

- O slides dessa aula foram baseados no material de MC102 do Prof. Eduardo Xavier (IC/Unicamp)

- Decisões e Seleção
  - https://panda.ime.usp.br/pensepy/static/pensepy/06-Selecao/selecao.html
  - https://runestone.academy/runestone/static/thinkcspy/Selection/toctree.html