Curso Técnico Nível Médio Subsequente

Informática Para Internet

Fundamentos de Lógica e Algoritmos

Aula 06
Entrada e Saída, Tipos de Dados Básicos e Expressões

Bruno Emerson Gurgel Gomes

Instituto Federal de Educação, Ciência e Tecnologia
do Rio Grande do Norte
Natal-RN
2015

Presidência da República Federativa do Brasil

Ministério da Educação

Secretaria de Educação a Distância

Este Caderno foi elaborado em parceria entre o Instituto Federal de Educação, Ciência e Tecnologia e o Sistema Escola Técnica Aberta do Brasil – e-Tec Brasil.

Equipe de Elaboração
Cognitum

Coordenação Institucional
COTED

Professor-autor
Bruno Emerson Gurgel Gomes

Projeto Gráfico
Eduardo Meneses e Fábio Brumana

Diagramação
Rômulo França

Ficha catalográfica

G633c Gomes, Bruno Emerson Gurgel.

Curso Técnico Nível Médio Subsequente Informática para Internet : Fundamentos de Lógica e Algoritmos - Aula 06 : Entrada e saída, tipos de dados básicos e expressões / Bruno Emerson Gurgel Gomes. – Natal : IFRN Editora, 2015.

21 f. : il. color.

1. Fundamentos de Lógica e Algoritmos - EaD. 2. Linguagem de programação. 3. Entrada e saída de dados. 4. Algoritmos. I. Título.

RN/IFRN/EaD CDU 004.421

Ficha elaborada pela bibliotecária Edineide da Silva Marques, CRB 15/488

Apresentação da disciplina

Olá, aluno(a)! Na aula 5 aprendemos sobre os fundamentos da programação. Vimos que nossos programas são compostos por algoritmos, que são uma forma de descrever um problema usando um conjunto de instruções estruturadas que produzem um resultado. Convido você a continuar esta caminhada em busca de conhecimento na programação de computadores. Nesta aula, vamos aprender como pedir ao usuário do programa para inserir valores e como exibir um resultado para esse usuário. Cada valor possui um tipo de dados. Vamos conhecer quais os tipos de dados mais comuns da linguagem *Python*. Além disso, você irá aprender a criar expressões compostas de valores desses diversos tipos de dados.

Aula 6 - Entrada e Saída, Tipos de Dados Básicos e Expressões

Objetivos

Caro(a) aluno(a), nesta aula são apresentados comandos na linguagem *Python* que permitem inserir, modificar e exibir valores em seu programa. Posteriormente, são demonstrados os tipos de dados básicos utilizados na representação de informações. Por fim, vamos conhecer os operadores aritméticos, lógicos e relacionais e aprender a criar expressões que usem esses operadores. Ao término da aula, você dever estar apto(a) a:

- Ler do teclado e escrever valores na tela com o uso das instruções *input* e *print*;
- Utilizar os tipos básicos de dados utilizados na representação de informações em um programa;
- Utilizar operadores, variáveis e valores constantes para criar expressões.

Desenvolvendo o conteúdo

Instruções Básicas de Entrada e Saída de dados

Até o momento trabalhamos com valores constantes, por exemplo, números como 50.0 ou um texto como "Bom dia, Usuário!" na aula anterior, vimos também como salvar esses valores em variáveis para possibilitar o seu uso em um momento posterior no programa. Neste momento, vamos aprender como obter um valor que deve ser fornecido por quem está usando seu programa (usuário). Vamos ver também a operação contrária, ou seja, como exibir um valor ou texto de saída para o usuário.

É importante destacar que essas atividades de obter informações (entrada de dados) e exibir um resultado (saída) são uma das tarefas mais básicas em

programação. Em verdade, um *software* basicamente é formado por uma ou mais etapas de entrada de dados, processamento desses dados e saída (Figura 1).

Fonte: http://voufazersemreclamar.blogspot.com.br/p/6-ano.html

Figura 01: Entrada, processamento e saída

Observe que neste curso a entrada de dados será feita através do teclado do computador, em um ambiente de linha de comando. É possível também a entrada por uma interface gráfica composta por janelas, botões e ícones. Nesse caso, o usuário também pode usar o *mouse* como dispositivo de entrada. O processamento corresponde a usar esses dados recebidos para realizar a tarefa a qual o programa se propõe. A saída é a etapa final, onde o(s) resultado(os) do processamento é(são) exibido(s) para o usuário do programa.

Por exemplo, em um sistema de cadastro de um cliente em uma loja na Internet, a entrada seria o formulário que o cliente preenche com os seus dados, como nome, CPF, identidade, endereço etc. O processamento seria a etapa de salvar essas informações em um banco de dados no computador. Por fim, a saída corresponde à mensagem que é exibida após o processamento, indicando se a operação foi realizada com sucesso ou não.

Muito bem, você está pronto para aprimorar ainda mais seus conhecimentos na programação em *Python*? Pois vamos começar pela instrução (ou comando) *print*, que significa escrever ou imprimir. Abra o interpretador *Python* (*idle*) e digite a instrução abaixo.

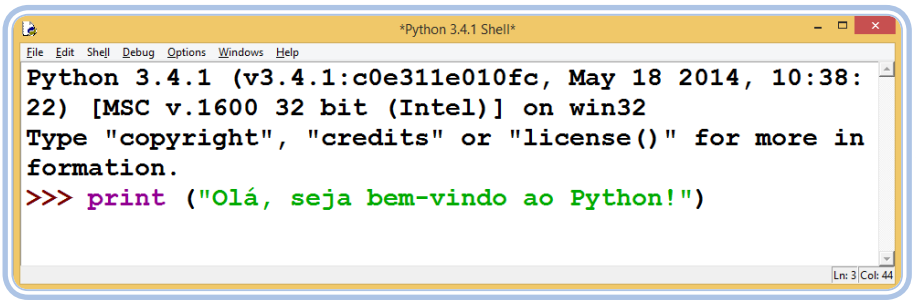
Código 1 - Imprime na tela uma mensagem para o usuário

```
>>> print ("Olá, seja bem-vindo ao Python!")
```

LEMBRE-SE

O sinal ">>>" no interpretador *Python* indica que ele está aguardando que você digite um comando.

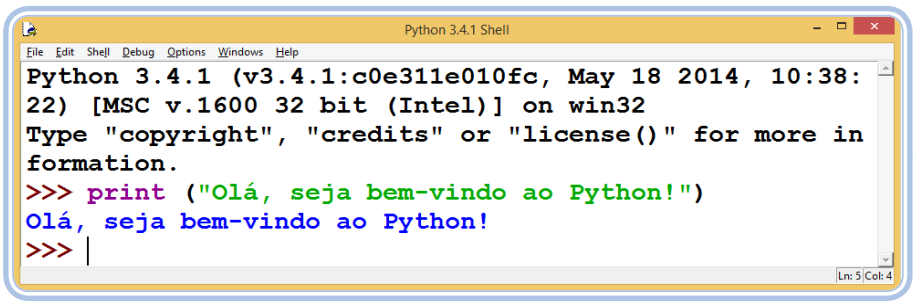
Para que você veja se está no caminho certo, a janela do interpretador com o código digitado deve se assemelhar a Figura 2. Após digitar o comando, ao apertar o botão ENTER, o comando vai ser processado pelo interpretador e, caso ele tenha algum resultado, esse resultado será exibido para você logo abaixo (Figura 3).



Fonte: Autoria própria.

Figura 02: Interpretador Python com instrução *print* digitada

O resultado que será exibido no comando *print* é o valor que está entre os parênteses. No nosso caso, o texto "*Olá, seja bem-vindo ao Python!*" nta verdade, vamos usar *print* sempre que for necessário fornecer uma mensagem ou resultado do programa para o usuário.



Fonte: Autoria própria.

Figura 03: Resultado do processamento da instrução *print* (texto em azul)

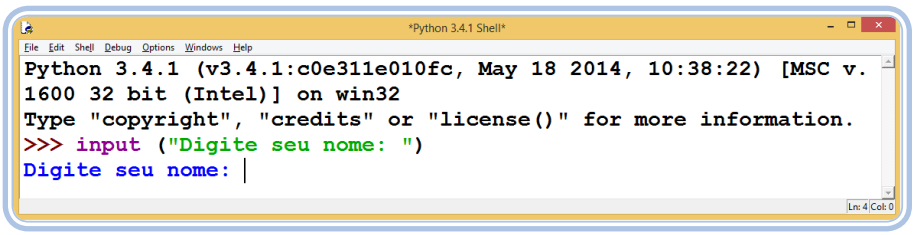
Para receber uma entrada do usuário pelo teclado do computador devemos usar o comando *input*, que tem a forma input ("*Mensagem*"). A mensagem que é recebida no comando deve ser qualquer texto que informe ao usuário do programa o que ele deve digitar.

Vamos fazer um programa para ler o nome de uma pessoa, que pode ser o seu. Nesse caso, devemos inserir uma instrução *input* solicitando que o nome seja digitado, como no código 2 a seguir.

Código 2 – Obtém o nome de uma pessoa usando a instrução input

```
>>> input ("Digite seu nome: ")
```

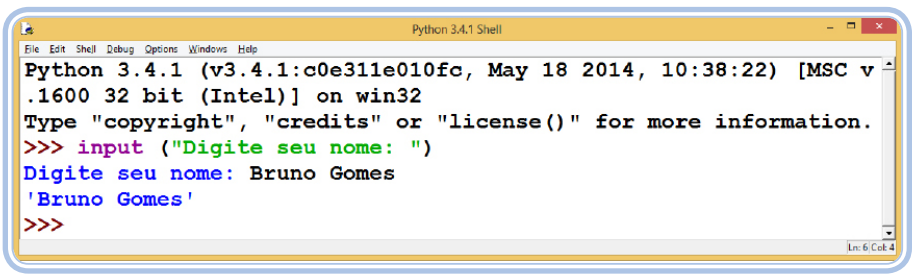
Inicialmente, o comando *input* insere na tela a mensagem pedindo a entrada de algum valor e fica aguardando o usuário digitá-lo. No caso do nosso exemplo, esse valor é o nome de uma pessoa (Figura 4).



Fonte: Autoria própria.

Figura 04: Entrada de dados com o comando *input*

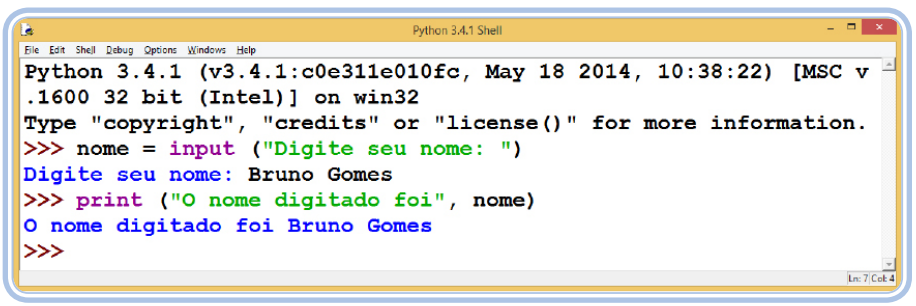
Após inserir o valor (o texto com o seu nome), você deve clicar no botão ENTER. Isso faz com que o valor seja enviado para o mesmo lugar do programa onde se encontra a instrução *input* (Figura 5).



Fonte: Autoria própria.

Figura 05: Resultado da leitura com input usando o interpretador Python

Caso você queira salvar esse valor em uma variável, é preciso apenas que você coloque o comando *input* à direita da atribuição (sinal de "="), conforme pode ser visto na Figura 6. Nesse caso, o nome digitado fica salvo na variável *nome*. A Figura 6 também exibe o resultado da impressão do valor armazenado usando a instrução *print* (*nome*).



Fonte: Autoria própria.

Figura 06: Salvando um nome lido com input e imprimindo com *print*

ATIVIDADE

1. Leia o primeiro nome de uma pessoa e a sua idade e salve nas variáveis nome e idade, respectivamente. Imprima a frase "*Olá, nome, você tem, idade, anos*", na qual os *nome* e *idade* representam os valores que estão armazenados nas variáveis.

OBS: Para imprimir mais de um valor em uma mesma frase, é preciso apenas que você separe os valores usando vírgula. E, sempre que for imprimir um texto, lembre-se de colocá-lo entre as aspas ou apóstrofo. Por exemplo: print ("Olá, ", nome, "hoje é dia", data), irá imprimir "Olá, Bruno, hoje é dia 22/02/2015, para os valores *nome = Bruno e data = 22/02/2015.*

2. Leia dois números, salve-os cada um em uma variável e imprima a soma desses números.

OBS: O que o usuário digita no teclado é recebido no comando *input* como um texto. Nesse programa, você deve converter a saída de *input* para um número com o comando *float* () envolvendo o *input*, como em:

```
n1 = float (input("Digite um número: "))
```

Na seção a seguir vamos detalhar os tipos de dados básicos que podem ser usados na linguagem *Python*. Os tipos presentes na linguagem não são restritos a esses. Porém, os tipos destacados abaixo são comumente usados na maioria dos programas.

Tipos de Dados Básicos

É importante você lembrar que criamos variáveis para guardar alguma informação que será útil ao programa. Essa informação, ou valor, deve ser de um tipo válido na linguagem. Desse modo, é possível fazer um cálculo, no caso de números, ou realizar alguma operação sobre um texto. A linguagem possui alguns tipos que são definidos por padrão e podem ser utilizados livremente nos seus programas.

Nesta seção, destacamos alguns tipos básicos que são usados com bastante frequência. Na seção de expressões, apresentamos como montar expressões

compostas com valores e operadores dos tipos detalhados a seguir.

Tipo inteiro (*int*)

Representa os números inteiros. Ou seja, qualquer valor numérico sem a parte fracionária. Exemplos: 0, -1, 234, -98.

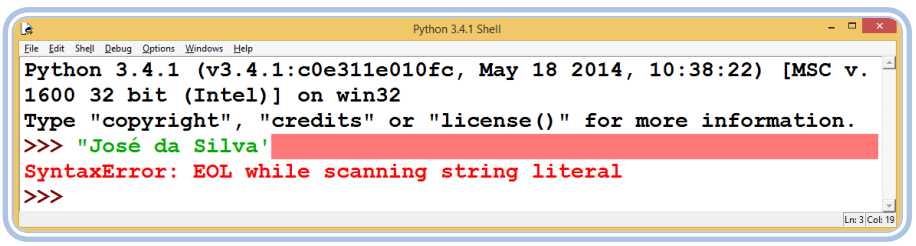
Tipo real (*float*)

Compreende os números reais, compostos por uma parte inteira e uma parte fracionária separadas por um ponto decimal. É importante reforçar que o separador decimal é um ponto, como na língua inglesa, e não uma vírgula, como no português. Exemplos: 0.0, 8.95, 2.1.

Tipo texto (*string*)

Informações compostas por zero ou mais caracteres alfanuméricos (letras, dígitos e símbolos especiais) colocados entre aspas ou apóstrofos. Na linguagem *Python*, o tipo texto é denominado de *string*. Exemplos: "IFRN", "José da Silva", 'Rua das flores, n. 44', ''a", '"".

É importante notar que se você começou o texto com aspas ('') você deve fechá-lo também com aspas. O mesmo vale para apóstrofo('). Caso você misture os dois formatos em um mesmo *string*, irá ocorrer um erro de sintaxe, como pode ser visto na Figura 7.



Fonte: Autoria própria.

Figura 07: Erro ao tentar iniciar um texto com aspas e fechar com apóstrofo

É interessante perceber que o espaço em branco também é um caractere significativo, podendo ser representado por (' ' ou " "). Se você juntar as aspas, sem dar nenhum espaço, você está definindo um texto vazio. Isso pode ser útil para criar uma nova variável do tipo *string* sem colocar nenhum valor significativo nela.

Tipo booleano (*bool*)

O tipo *bool* corresponde as informações que podem assumir os valores lógicos *verdadeiro* ou *falso*. Em *Python*, o valor verdadeiro é representado pela palavra *True* e *falso* pela palavra *False*. Essas palavras devem ser escritas da forma como foram apresentadas, com a primeira letra em maiúscula.

Por exemplo, suponha que no seu programa você esteja procurando um valor, como um número em uma lista de números. É possível criar uma variável denominada "encontrado" para representar se esse valor foi ou não encontrado. Na inicialização da variável é interessante inserir o valor falso, pois o valor ainda não foi encontrado (encontrado = *False*). Após o processamento do seu programa, caso o valor seja encontrado, você deve mudar o valor armazenado para verdadeiro (encontrado = *True*).

Consultando o tipo de um valor

Caso você queira consultar o tipo de um valor em *Python*, você pode usar o comando *type* (*valor*) como pode ser visto nos exemplos do Código 2.

Código 2 – Consultando o tipo de alguns valores em Python

```
>>> type (10)

<class 'int'>

>>> type ("João")

<class 'str'>

>>> x = 80.5

>>> type (x)

<class 'float'>
```

Você deve ter notado que não é necessário dizer de forma explícita qual o tipo de uma variável em *Python*. O tipo é estabelecido de acordo com o valor que a variável recebe. Caso ela receba um número inteiro, o seu tipo será inteiro, caso receba um texto, o tipo será *string* e assim por diante. Isso é o que chamamos de atribuição dinâmica de tipos.

Embora a mudança de tipo em uma variável seja permitida, não é aconselhável que você use este recurso, pois torna mais difícil a descoberta e a correção de erros em seu programa. Portanto, sempre pense em uma variável como sendo de apenas um tipo. Caso você precise guardar outro valor, é preferível criar uma nova variável.

Conversão entre tipos

É possível, e por vezes necessário, converter entre os tipos de dados. *Python* possui funções tais como *int*(expressão), *float*(expressão) e *str*(expressão) para forçar a conversão para inteiro (*int*), real (*float*) e texto (*string*), respectivamente. Por exemplo, caso você queira converter a constante 9.5 para o inteiro 9, basta fazer *int* (9.5).

A conversão é particularmente importante quando você quer ler um valor usando *input*, que sempre devolve um texto (*string*). Nesse caso, se você quer ler um número, é necessário converter a saída de *input* para *int* ou *float*, como no exemplo do Código 3.

Código 3 – Convertendo a saída de input para número

```
>>> x = input (“Digite a sua nota: “)

>>> nota = float (x)
```

Você também pode fazer a conversão direto na linha da leitura com *input*, como por ser visto no código 4.

Código 4 – Convertendo a saída de input para número direto na linha do input

```
>>> nota = float (input (“Digite a sua nota: “))
```

ATIVIDADE

1. Leia dois valores do tipo inteiro e armazene-os em duas variáveis, denominadas de *n1* e *n2*, respectivamente. Lembre-se de converter o a saída da função *input* para inteiro (*int*), nas duas leituras (*n1* e *n2*).

a) Realize e imprima o resultado a divisão de *n1* por *n2* (*n1/n2*).

b) Verifique, usando *type*, o tipo da divisão realizada no item 'a'. Exemplo: *type* (n1/n2) ou *type* (res), caso você tenha salvo o resultado em uma variável denominada de *res*.

Expressões

As expressões são úteis sempre que precisamos realizar um cálculo, fazer comparações ou realizar testes. Sendo assim, é de extrema importância saber como criar expressões e entender o valor que elas devolvem. Por exemplo, se digitarmos o número 10 no interpretador *Python*, nós temos uma expressão formada por um valor constante do tipo inteiro, o valor 10. Caso você digite 2 * 4.0, tem-se uma expressão do tipo real composta de duas constantes e do operador de multiplicação (*). Nesse caso, o resultado é o valor 8.0. Antes de seguir adiante, vamos definir o que é uma expressão.

[DEFINIÇÃO] Expressão

Uma expressão é uma variável, uma constante, ou qualquer combinação válida de variáveis, constantes e operadores que devolve um resultado após a sua avaliação.

Como pode ser visto pela definição, expressões usualmente são compostas por operadores. Há diversos operadores disponíveis, a depender do tipo da expressão. Nesta seção, você irá trabalhar com expressões aritméticas (inteiros e reais), lógicas (verdadeiro – *True* ou falso – *False*) e relacionais (resultados de comparações, verdadeiro ou falso). No caso de expressões aritméticas, são exemplos de operadores os símbolos + (adição), - (subtração), * (multiplicação) e / (divisão).

Observe que um mesmo operador pode aparecer em expressões de mais de um tipo, como o operador "+". Caso a expressão seja *string*, o operador "+" tem a função de unir duas ou mais *strings* na expressão. Por exemplo, "Olá" + "mundo!" irá gerar o texto "Olá mundo!".

Os operadores podem ser classificados como *unários*, quando são aplicados sobre apenas um operando (valor) ou *binários* quando devem ser aplicados a dois valores. Como exemplo, o operador "+" (soma) na expressão 5 + 6 é

binário, pois é aplicado à soma do valor 5 com o valor 6. Já o operador "–" (negação) é unário na expressão –6, representando a negação do número 6. Por outro lado, ele é binário e significa subtração na expressão 9 – 3.

Expressões aritméticas (numéricas)

São aquelas que operam sobre valores inteiros ou reais. Se os operandos em uma expressão são inteiros, o resultado da expressão será inteiro, exceto para a divisão, que na versão 3 do *Python* resulta em um real. Por exemplo, 4 / 2 (4 dividido por 2) irá resultar em 2.0. O resultado de uma expressão numérica será real sempre que ela for composta por valores reais ou quando há operandos inteiros e reais em uma mesma expressão.

A Tabela 1 a seguir apresenta os símbolos usados para os operadores aritméticos e o seu significado. Observe que devido ao teclado do computador não ser capaz de apresentar todos os símbolos, alguns não são exatamente como aprendemos na escola. Por exemplo, a multiplicação é o "*" (asterisco) e a exponenciação é representada por "**" (dois asteriscos juntos).

Tabela 01: Operadores Aritméticos

Operador	Significado	Exemplos
+	Adição	2 + 3, x = y + 2
–	Subtração	6 – 1, x – a – 2
*	Multiplicação	7 * 3, num * 2
/	Divisão	15 / 3, (4 + 4) / 2
//	Parte inteira da divisão (quociente)	15 // 2 irá resultar em 7
%	Resto da divisão	6 % 2 (resultado é 0)
**	Exponenciação	2 ** 3 (resultado é 8)

Fonte: Autoria própria.

Observe que as expressões são escritas na forma linear, ou seja, em uma linha de texto, usando os operadores descritos na tabela. Por exemplo, a expressão $5^{3/2}$ (5 vezes 3 dividido por 2) deve ser traduzida em *Python* na forma linear como *5 * 3 / 2.* No caso da expressão $a^2 + b^2 = c^2$ temos como equivalente em *Python* a forma *a**2 + b**2 = c**2*.

ATIVIDADE

Traduza as expressões matemáticas abaixo para a forma linear usando os operadores da linguagem Python.

a) $(4\pi r^3)/3$

b) $b^2 - 4ac$

c) $r = 1/2\ at^2 + v_o t + r_0$

Você deve ter observado pelos exemplos que é possível ter em uma mesma expressão diversos operadores. Nesse caso, saiba que a linguagem estabelece uma ordem para a avaliação de operadores, o que é chamado de precedência. Operadores com maior precedência são avaliados (calculados) primeiro que aqueles com menor precedência. Caso tenhamos operadores com a mesma precedência sendo avaliados, a ordem de avaliação é da esquerda para a direita.

Tabela 02: Ordem de execução (precedência) dos operadores aritméticos. Da maior precedência para a menor

Procedência	Operador(es)	Nome
1	**	Exponenciação
2	*, /, //, %	Multiplicação, divisão e resto da divisão
3	+, -	Adição e subtração

Fonte: Autoria própria.

A lista de precedência (ou prioridade) da Tabela 2 diz que, em uma expressão aritmética, primeiro devemos resolver a exponenciação, depois, na mesma prioridade, a multiplicação, divisão e o resto (nesse caso, da esquerda para a direita), e por último a adição e a subtração. Caso você queira forçar a avaliação de operadores com precedência menor, antes de um de prioridade maior, você deve colocar essa parte da expressão entre parênteses. Abaixo, vemos alguns exemplos de resolução de expressões passo-a-passo seguindo a ordem de precedência dos operadores. Caso você insira essas expressões no interpretador, você irá obter apenas o resultado final da resolução.

Exemplos:

5 + 9 + 7 + 8/4 5 + 9 + 7 + 2.0 21 + 2.0 23.0	**1 - 4 * 3//6 - 3**2** 1 - 4 * 3//6 - 9 1 - 12//6 – 9 1 - 2 – 9 -10
(5 + 9 + 7) + 8/4 21 + 8/4 21 + 2.0 23.0	**(1 - 4) * 3/6 - 3**2** -3 * 3/6 - 3**2 -3 * 3/6 - 9 -9/6 - 9 -1.5 - 9 -10.5

Expressões Lógicas

Uma expressão lógica é aquela formada por operadores lógicos ou de comparação (relacionas). O resultado da avaliação de uma expressão lógica é sempre do tipo lógico, ou seja, verdadeiro (*True*) ou falso (*False*). Você já deve estar familiarizado com estas expressões, pois de certo modo elas já foram tratadas no contexto da lógica proposicional, na unidade 1.

Tabela 03: Operadores lógicos

Operador	Significado	Exemplos
Or	Ou (disjunção)	x **or** y
And	E (conjunção)	(2 > x) **and** (y < z)
Not	Não (negação)	**not** (a == b)

Fonte: Autoria própria.

Embora já seja do seu conhecimento, pelo estudo da unidade 1, para relembrar exibimos a seguir a tabela verdade para cada operador. As tabelas verdade mostram o resultado de todas as possíveis combinações de valores entre operandos que estão conectados usando algum operador lógico.

Tabela 04: Tabela verdade do operador 'and' Fonte: autoria própria.

A	B	A and B
False	False	False
False	True	False
True	False	False
True	True	True

Fonte: Autoria própria.

Tabela 05: Tabela verdade do operador 'or'

A	B	A or B
False	**False**	**False**
False	**True**	**True**
True	**False**	**True**
True	**True**	**True**

Fonte: Autoria própria.

Tabela 06: Tabela verdade do operador 'not'

A	not B
True	False
False	True

Fonte: Autoria própria.

A precedência entre os operadores lógicos é exibida na Tabela 7. Inicialmente, é feito a negação (*not*). Posteriormente, é tratado o e (*and*) e, por fim, o ou (*or*).

Tabela 07: Precedência entre os operadores lógicos (da maior para a menor)

Prededência	Operador	Nome
1.	Not	Não
2.	And	E
3.	Or	OU

Fonte: Autoria própria.

Exemplos:

2 < 5 and 15 > 15 True and FalsetTrue	not (5 >= 3) or True not (True) or True False or True True

Operadores relacionais

São utilizados para se fazer comparações entre expressões. O resultado dessas comparações é lógico, ou seja, verdadeiro (*True*) ou falso (*False*).

Tabela 08: Operadores Relacionais

Operador	Significado	Exemplos
==	Igual a	3 == 3, x == y
>	Maior que	5 > 4, (y + z) > 8
<	Menor que	3 < 6, num < 12
>=	Maior ou igual a	5 >= 56, (a * b) >= (8 * 2)
<=	Menor ou igual a	3 <= 9, (a - b) + c >= 1
!=	Diferente de	8 != 9, a != b
is	É o mesmo que	x is y (x faz referência ao mesmo objeto de y)
is not	Não é o mesmo que	x is not y

Fonte: Autoria própria.

Exemplos:

2 * 4 == 24 / 3 8 == 8 True	3 * 5 / 4 <= 3**2 /0.5 3 * 5 / 4 <= 9 / 0.5 15 / 4 <= 18 3.75 <= 18 True
15 % 4 < 19 % 6 3 < 1 False	2 + 8 % 7 >= 3 * 6 - 15 2 + 1 >= 18 - 15 3 >= 3 True

Os operadores relacionais possuem a mesma precedência entre si. A precedência entre todos os operadores vistos é exibida na Tabela 9. Lembre-se que para forçar a avaliação de uma expressão com precedência menor, você deve colocá-la entre parênteses.

Tabela 09: Precedência entre todos os operadores vistos

Precedência	Operador(es)
1.	**
2.	*, /, //, **
3.	+, -
4.	>, >=, <, <=, ==, !=
5.	is, is not
6.	Not
7.	And
8.	Or

Fonte: Autoria própria.

RESUMINDO

Você aprendeu nesta aula a realizar a entrada e a saída de dados usando *input* e *print*. Vimos que o *input* lê do teclado e devolve um texto com o valor lido. Nesse caso, se você quiser usar um número você deve converter a saída com input usando *int* e *float*. Os tipos de dados básicos *int, float, string e bool* foram detalhados e aprendemos a criar expressões com esses tipos. Uma expressão é composta por variáveis, constantes e operadores e, após ser avaliada, devolve um resultado para o usuário.

LEITURAS COMPLEMENTARES

PYTHON.ORG (https://docs.python.org/3/tutorial/inputoutput.html) – Esta

página faz parte da documentação oficial da linguagem *Python* e é interessante para quem quer se aprofundar mais em entrada e saída de dados. Nela, você terá exemplos de saída com formatação e de entrada e saída em arquivos, dentre outros tópicos.

IBM (http://www.ibm.com/developerworks/br/library/os-python1/) – Neste *site*, você tem um material complementar sobre tipos básicos em Python e as expressões que vimos nesta aula.

AVALIANDO SEUS CONHECIMENTOS

1. Verifique o que está errado nos códigos a seguir e faça as devidas correções.

a) >>> print 'Instituto Federal"

b) >>> nome$ = 10

c) >>> x = input ("Número 1: ", "Número 2: ")

2. Qual o valor da variável "x" ao final da execução do código a seguir supondo que o usuário digite o valor 10 na instrução *input*?

```
>>> x = input ("Digite um número")

>>> x = (x + 5) * 2 – 10

>>> print ("Resultado", x)
```

3. Leia dois textos usando a instrução *input* e salve-os nas variáveis *txt1* e *txt2*. Posteriormente, junte esses textos usando o operador "+".

4. Insira as expressões abaixo no interpretador *Python*. Posteriormente, verifique o tipo de cada uma delas usando *type.*

a) 5 + 10 – 6

b) 89 / 10 * (2 + 1)

c) 100 * “3”

d) 10 > 5 or 9 <= 30 and ‘a’ == ‘b’

e) 8.5 != 3 and 5 > 10

5. Dadas as variáveis numéricas “**x**”, “**y**” e “**z**”, contendo os valores 3, 5 e 7, respectivamente; a variável **tipo**, contendo o literal “TEXTO”; e a variável lógica **teste**, contendo o valor lógico falso(*False*), assinale abaixo qual a expressão lógica cujo resultado possui o valor lógico verdadeiro (*True*). Desenvolva a opção escolhida passo-a-passo.

( ) tipo == “TEXTO” **and** teste

( ) teste **or** x + y < z

( ) x – y > z **and** tipo == “NUMÉRICO”

( ) (x**3) – y > z **and** teste **or** tipo == “TEXTO”

6. Com o auxílio do interpretador *Python* calcule o valor das expressões abaixo.

a) 2 * 3 + 4

b) 2 * (3 + 4)

c) 2 * (3 + 4) + 3 * 5

d) 8 / 3 + 4 * 6 - 2 ** 5

CONHECENDO AS REFERÊNCIAS

MANZANO, J. A. N. G.; OLIVEIRA, J. F. **Algoritmos:** Lógica para Desenvolvimento de Programação de Computadores. São Paulo: Érica, 2014.

IBM. Descobrir Python, Parte 1: Tipos Numéricos Integrados da Python. [200-?] Disponível em: <http://www.ibm.com/developerworks/br/library/os-python1/>. Acesso em: 02 mar. 2015.

PYTHON BRASIL. Disponível em: <http://wiki.python.org.br/>. Acesso em: 05 fev. 2015.

PYTHON.ORG. Input and Output. [20--?]. Disponível em: <https://docs.python.org/3/tutorial/inputoutput.html>. Acesso em: 02 mar. 2015.

PYTHON SOFTWARE FOUNDATION. Disponível em: <https://www.python.org/>. Acesso em: 05 fev. 2015.

SEVERANCE, C. **Python for Informatics:** Exploring Information. 2013. Disponível em: <http://www.pythonlearn.com/book.php>. Acesso em: 05 fev. 2015.