Tipos de Dados, Operadores, Estruturas Condicionais e de Repetição

Instrutor: Tarik Ponciano

Senac Fecomércio Sesc

Links da Disciplina

1. Discord: https://discord.gg/wt5CVZZWJs

2. Drive: https://drive.google.com/drive/folders/1hOl0DaPeAor7gnhKBUIlZ5n8lLRDNvUY?usp=sharing

3. Github: https://github.com/TarikPonciano/Programador-de-Sistema-SENAC

Tipos de Dados

Um tipo de dado nada mais que é algo do mundo real que pode ser representado computacionalmente. Por exemplo, os números que pertencem ao conjunto dos números inteiros, os números que pertencem ao conjunto dos números reais, letras, caracteres especiais, acentuação, pontuação, palavras, etc.

As linguagens de programação implementam formas de representar e manipular esses dados, que podem ser classificados em dois grandes grupos: os tipos de dados primitivos e os tipos de dados não primitivos.

Tipos de Dados

Os tipos de dados primitivos são os tipos básicos que devem ser implementados por todas as linguagens de programação, como os números reais, inteiros, booleanos, caracteres e strings.

Os tipos de dados não primitivos, normalmente são os vetores, matrizes, classes, enumerações, etc., que costumam ser estruturas de dados mais complexas do que os tipos de dados primitivos.

Tipos de Dados

- **Inteiro**: informação pertencente ao conjunto dos números inteiros. Números do conjunto dos números inteiros, positivos, negativos e nulos. Exemplo: -10, 0, 5, 100.

- **Real**: informação pertencente ao conjunto dos números reais. O conjunto de números inteiros está contido no conjunto dos números reais. A casa decimal é representada por ponto e não vírgula. Exemplo: 5.2, -3.93, 0.0, 7.

Tipos de Dados

• **Lógico**: informação que pode receber (assumir) apenas dois valores possíveis: verdadeiro (V) ou falso

Ex: x = 7 >= 2 então x = Verdadeiro

• **Caracter**: informação composta por um conjunto de caracteres alfanuméricos.). O caracter tem que vir com apóstrofo – ou aspas simples. Exemplo: ‘a’, ‘ABC’, “F10B5’, ‘$?!5’.

Tipos de Dados

•**STRING**: não é um tipo de dados primitivo, mas sim derivado. Uma String é um conjunto de caracteres. Exemplos: “Elaine”, “Funcionário”, etc. Portanto, uma palavra, uma frase ou um texto é representado por Strings.

Tipos de Dados

TIPOS DE DADOS PRIMITIVOS INTERNOS	
TIPO	**SIGNIFICADO**
Boolean	Representa valores verdadeiro ou falso
Byte	Representa um valor inteiro de 8 bits
Char	Representa um caractere (uma letra, símbolo, pontuação, etc)
Double	Representa um valor de ponto flutuante de precisão dupla
Float	Representa um valor de ponto flutuante de precisão simples
Int	Representa um valor inteiro
Long	Representa um valor inteiro longo
Short	Representa um valor inteiro curto

Tipos de Dados

TIPOS DE DADOS INTEIROS		
TIPO	**BITS**	**INTERVALO**
byte	8	-128 a +127
short	16	32.768 a +32.767
int	32	-2.147.483.648 a +2.147.483.648
long	64	-9.223.372.036.854.775.808 a +9.223.372.036.854.775.808

Tipos de Dados

TIPOS DE DADOS DE PONTO FLUTUANTE		
TIPO	**BITS**	**INTERVALO**
float	32	1.40239846e-46 a é 3.40282347e+38 Tamanho do maior literal float: $3,4x10^{38}$ Norma IEEE 754 Ponto flutuante com sinal de precisão simples Números positivos e negativos
double	64	4.94065645841246544e-324 a 1.7976931348623157e+308 Tamanho do maior literal double: $1,8x10^{308}$ Norma IEEE 754 Ponto flutuante com sinal com precisão dupla Números positivos e negativos

Senac
Fecomércio
Sesc

Tipos de Dados

TIPOS DE DADOS CARACTERE		
TIPO	**BITS**	**INTERVALO**
char	16	0 a 65.536 Pode ser usado como tipo inteiro de 16 bits sem sinal '\u0000' a '\uFFFF'

TIPOS DE DADOS BOOLEAN		
TIPO	**BITS**	**INTERVALO**
boolean	8	True ou false Verdadeiro ou falso

Tipos de Dados no Python

Example	Data Type
`x = str("Hello World")`	str
`x = int(20)`	int
`x = float(20.5)`	float
`x = complex(1j)`	complex
`x = list["apple", "banana", "cherry"]`	list
`x = tuple("apple", "banana", "cherry")`	tuple
`x = range(6)`	range
`x = dict{name="John", age=36}`	dict
`x = set{"apple", "banana", "cherry"}`	set
`x = frozenset({"apple", "banana", "cherry"})`	frozenset
`x = bool(5)`	bool
`x = bytes(5)`	bytes
`x = bytearray(5)`	bytearray
`x = memoryview(bytes(5))`	memoryview

https://www.w3schools.com/python/python_datatypes.asp

Senac
Fecomércio
Sesc

Operadores

Em **Lógica de Programação** e Algoritmos, **Operadores** são símbolos que dizem ao compilador para realizar manipulações (operações) matemáticas, lógicas e de comparação específicas.

1. Aritméticos
2. Relacionais
3. Atribuição
4. Lógicos
5. Incremento / Decremento
6. Concatenação
7. Condicional

Operadores Aritméticos

Os **operadores aritméticos** são usados na realização de cálculos aritméticos simples, usando as quatro operações básicas da matemática mais operações como o módulo.

Operador	Operação	Como é usado	Explicação
+	Soma	a + b	Realiza a adição dos dois operandos
–	Subtração	a – b	Subtrai o operando da direita do operando à esquerda
*	Multiplicação	a * b	Realiza a multiplicação entre os dois operandos
/	Divisão	a / b	Divide o operando à esquerda pelo operando à direita
%	Módulo	a % b	Retorna o resto da divisão inteira do operando à esquerda pelo operando à direita
** ^	Exponenciação	a ** b	Retorna o valor de a elevado a b: a^b (potência)

Operadores Relacionais

Os **operadores relacionais** permitem estabelecer uma relação entre dois valores (operandos). Funcionam com quaisquer tipos de dados – desde que os operandos sejam do mesmo tipo.
Comparam o valor à esquerda com o valor à direita do operador, retornando um valor lógico (verdadeiro ou falso) de acordo com o resultado da comparação.

Operador	Operação	Como é usado	Explicação
==	Igualdade	a == b	Verifica se os dois valores são iguais entre si. Se forem, retorna verdadeiro; caso contrário, retorna falso.
!=	Diferença	a != b	Verifica se os dois valores são diferentes um do outro. Se forem, retorna verdadeiro; caso contrário, retorna falso.
>	Maior que	a > b	Verifica se o valor à esquerda é maior do que o valor à direita. Se for, retorna verdadeiro; senão, retorna falso.
<	Menor que	a < b	Verifica se o valor à esquerda é menor do que o valor à direita. Se for, retorna verdadeiro; senão, retorna falso.
>=	Maior ou igual a	a >= b	Verifica se o valor à esquerda é maior ou igual ao valor à direita. Se for, retorna verdadeiro; senão, retorna falso.
<=	Menor ou igual a	a <= b	Verifica se o valor à esquerda é menor ou igual ao valor à direita. Se for, retorna verdadeiro; senão, retorna falso

Operadores de Atribuição

São operadores empregados para realizar a atribuição de valores entre os operandos, como por exemplo a atribuição de um valor a uma variável.

Operador	Operação	Como é usado	Equivale a	Explicação
=	Atribuição	a = b		Atribui (grava) no operando à esquerda o valor do operando à direita.
+=	Soma com atribuição	a += b	a = a + b	Adiciona o operando do lado direito ao operando do lado esquerdo e atribui o resultado ao operando do lado esquerdo.
-=	Subtração com atribuição	a -= b	a = a – b	Subtrai o operando do lado direito do operando do lado esquerdo e atribui o resultado ao operando do lado esquerdo.
*=	Multiplicação com atribuição	a *= b	a = a * b	Multiplica o operando do lado direito pelo operando do lado esquerdo e atribui o resultado ao operando do lado esquerdo.
/=	Divisão com atribuição	a /= b	a = a / b	Divide o operando do lado esquerdo pelo operando do lado direito e atribui o resultado ao operando do lado esquerdo.
%=	Módulo com atribuição	a %= b	a = a % b	Obtém o módulo entre os operandos do lado esquerdo e direito e atribui o resultado ao operando do lado esquerdo.
&=	AND bitwise com atribuição	a &= b	a = a & b	Realiza a operação **a & b** e armazena o resultado de volta em a
|=	OR bitwise com atribuição	a |= b	a = a | b	Realiza a operação **a | b** e armazena o resultado de volta em a
^=	XOR bitwise com atribuição	a ^= b	a = a ^ b	Realiza a operação **a ^ b** e armazena o resultado de volta em a
<<=	Deslocamento à esquerda com atribuição	a << b	a = a << b	Desloca o valor de a por **b** bits para a esquerda e armazena o resultado de volta em a.
>>=	Deslocamento à direita com atribuição	a >> b	a = a >> b	Desloca o valor de a por **b** bits para a direita e armazena o resultado de volta em a.

Senac Fecomércio Sesc

Operadores de Atribuição

Operador	Operação	Como é usado	Equivale a	Explicação
=	Atribuição	a = b		Atribui (grava) no operando à esquerda o valor do operando à direita.
+=	Soma com atribuição	a += b	a = a + b	Adiciona o operando do lado direito ao operando do lado esquerdo e atribui o resultado ao operando do lado esquerdo.
-=	Subtração com atribuição	a -= b	a = a – b	Subtrai o operando do lado direito do operando do lado esquerdo e atribui o resultado ao operando do lado esquerdo.
*=	Multiplicação com atribuição	a *= b	a = a * b	Multiplica o operando do lado direito pelo operando do lado esquerdo e atribui o resultado ao operando do lado esquerdo.
/=	Divisão com atribuição	a /= b	a = a / b	Divide o operando do lado esquerdo pelo operando do lado direito e atribui o resultado ao operando do lado esquerdo.
%=	Módulo com atribuição	a %= b	a = a % b	Obtém o módulo entre os operandos do lado esquerdo e direito e atribui o resultado ao operando do lado esquerdo.
&=	AND bitwise com atribuição	a &= b	a = a & b	Realiza a operação **a & b** e armazena o resultado de volta em a
|=	OR bitwise com atribuição	a |= b	a = a | b	Realiza a operação **a | b** e armazena o resultado de volta em a
^=	XOR bitwise com atribuição	a ^= b	a = a ^ b	Realiza a operação **a ^ b** e armazena o resultado de volta em a
<<=	Deslocamento à esquerda com atribuição	a << b	a = a << b	Desloca o valor de a por **b** bits para a esquerda e armazena o resultado de volta em a.
>>=	Deslocamento à direita com atribuição	a >> b	a = a >> b	Desloca o valor de a por **b** bits para a direita e armazena o resultado de volta em a.

Operadores Lógicos ou Booleanos

Os **operadores lógicos** recebem valores booleanos (verdadeiro ou falso) como entrada e retornam valores também booleanos como saída.
São úteis para trabalhar com múltiplas condições relacionais na mesma expressão

Operador	Operação	Como é usado	Explicação
&& and	AND lógico (E)	a && b	Retorna verdadeiro se a e b forem verdadeiros, caso contrário retorna falso.
|| or	OR lógico (OU)	a || b	Retorna falso se nem a nem b forem verdadeiros; caso contrário, retorna verdadeiro
! not	NOT lógico (NÃO)	!b	Retorna verdadeiro se b for falso, senão retorna falso. Operador unário.

Operadores de Incremento e Decremento

Estes são operadores unários, ou seja, somente recebem um operando como entrada.

Operador	Operação	Como é usado	Explicação
++	Pós-incremento	a++	Incrementa a em 1 após usar seu valor
++	Pré-incremento	++a	Incrementa a em 1 antes de usar seu valor
—	Pós-decremento	a–	Decrementa a de 1 após usar seu valor
—	Pré-decremento	–a	Decrementa a de 1 antes de usar seu valor

Operadores de Concatenação

Concatenação: junção de duas ou mais sequências de caracteres (strings), formando uma nova sequência.

Operador	Operação	Como é usado	Explicação
+	Concatenação	a + b	O conteúdo de texto em a é unido ao conteúdo de texto em b, formando assim um novo texto (cadeia de caracteres: string)

Senac Fecomércio Sesc

Operador Condicional

Similar ao uso de um **condicional if-else** (se-senão).

c = (condição)? a : b

Se a *condição* retornar verdadeiro, então o valor de a é atribuído a c. Caso contrário, o valor de b será atribuído a c.

Trata-se de um operador ternário, pois emprega três operandos: a, b e a condição testada.

Estruturas de Condição

Em **Python**, assim como na maioria das linguagens de programação, o programa deve ser capaz de tomar decisões com base em valores e resultados gerados durante sua execução, ou seja, deve ser capaz de decidir se determinada instrução deve ou não ser executada de acordo com uma condição. Para atender a esse tipo de situação, podemos utilizar instruções especiais denominadas **estruturas condicionais**.

- **if**
- **Operadores de comparação**
- **if-elif-else**

Estruturas de Condição com o IF

O `IF` é uma **estrutura de condição** que permite avaliar uma expressão e, de acordo com seu resultado, executar uma determinada ação.

No **Código 1** a seguir, temos um exemplo de uso do IF no qual verificamos se a variável `idade` é menor que 20. Em caso positivo, imprimimos uma mensagem na tela, e em caso negativo, o código seguirá normalmente, desconsiderando a linha 3.

```
1  idade = 18
2  if idade < 20:
3      print('Você é jovem!')
```

Como podemos notar, essa estrutura é formada pela palavra reservada `if`, seguida por uma condição e por dois pontos (`:`). As linhas abaixo dela formam o bloco de instruções que serão executadas se a condição for atendida. Para isso, elas devem ser indentadas corretamente.

Estruturas de Condição com IF-ELIF-ELSE

Vimos anteriormente como utilizar o IF para executar uma ação caso uma condição seja atendida. No entanto, nenhum comportamento específico foi definido para o caso de a condição não ser satisfeita. Quando isso é necessário, precisamos utilizar a palavra reservada `else`. Adicionalmente, se existir mais de uma condição alternativa que precisa ser verificada, devemos utilizar o `elif` para avaliar as expressões intermediárias antes de usar o `else`.

```
1  idade = int(input('Digite sua idade: '))
2  if idade >= 10 and idade < 20:
3      print('Você é adolescente')
4  elif idade >= 20 and idade < 30:
5      print('Você é jovem')
6  elif idade >= 30 and idade <= 100:
7      print('Você é adulto')
8  else:
9      print('Valor não encontrado!')
```

https://www.w3schools.com/js/js_if_else.asp

Senac Fecomércio Sesc

Estruturas de Condição com IF-ELIF-ELSE

```
1  idade = int(input('Digite sua idade: '))
2  if idade >= 10 and idade < 20:
3      print('Você é adolescente')
4  elif idade >= 20 and idade < 30:
5      print('Você é jovem')
6  elif idade >= 30 and idade <= 100:
7      print('Você é adulto')
8  else:
9      print('Valor não encontrado!')
```

Na linha 2, verificamos se o valor informado está dentro de uma faixa de valores específica. Caso a condição seja satisfeita, o programa executará a linha 3. Por outro lado, caso o resultado não seja o esperado, então o programa verificará o próximo condicional, na linha 4 e, caso ele seja verdadeiro, a linha 5 será executada. O mesmo ocorre para a verificação da linha 6. Por fim, se nenhuma das condições foi satisfeita, o programa executará o que é especificado no bloco `else`.

https://www.w3schools.com/js/js_if_else.asp

Estruturas de Condição com IF-ELIF-ELSE

```
1  idade = int(input('Digite sua idade: '))
2  if idade >= 10 and idade < 20:
3      print('Você é adolescente')
4  elif idade >= 20 and idade < 30:
5      print('Você é jovem')
6  elif idade >= 30 and idade <= 100:
7      print('Você é adulto')
8  else:
9      print('Valor não encontrado!')
```

Na linha 2, verificamos se o valor informado está dentro de uma faixa de valores específica. Caso a condição seja satisfeita, o programa executará a linha 3. Por outro lado, caso o resultado não seja o esperado, então o programa verificará o próximo condicional, na linha 4 e, caso ele seja verdadeiro, a linha 5 será executada. O mesmo ocorre para a verificação da linha 6. Por fim, se nenhuma das condições foi satisfeita, o programa executará o que é especificado no bloco `else`.

https://www.w3schools.com/js/js_if_else.asp

Estruturas de Repetição

Em algumas situações, é comum que uma mesma instrução (ou conjunto delas) precise ser executada várias vezes seguidas. Nesses casos, normalmente utilizamos um loop (ou laço de repetição), que permite executar um bloco de código repetidas vezes, enquanto uma dada condição é atendida.

Em Python, os loops são codificados por meio dos comandos `for` e `while`. O primeiro nos permite percorrer os itens de uma coleção e, para cada um deles, executar um bloco de código. Já o `while`, executa um conjunto de instruções várias vezes enquanto uma condição é atendida.

Estruturas de Repetição FOR

```
1 | nomes = ['Pedro', 'João', 'Leticia']
2 | for n in nomes:
3 |     print(n)
```

A variável definida na linha 1 é uma lista inicializada com uma sequência de valores do tipo string. A instrução `for` percorre todos esses elementos, um por vez e, em cada caso, atribui o valor do item à variável `n`, que é impressa em seguida. O resultado, então, é a impressão de todos os nomes contidos na lista.

Estruturas de Repetição While

O comando `while`, por sua vez, faz com que um conjunto de instruções seja executado enquanto uma condição for atendida. Quando o resultado passa a ser falso, a execução é interrompida, saindo do loop, e passa para o próximo bloco.

Vemos um exemplo de uso do laço `while`, onde definimos a variável `contador`, iniciando com 0, e enquanto seu valor for menor que 5, executamos as instruções das linhas 3 e 4.

```
1 contador = 0
2 while contador < 5:
3     print(contador)
4     contador = contador + 1
```

Observe que na linha 4 incrementamos a variável `contador`, de forma que em algum momento seu valor igual a 5. Quando isso for verificado na linha 2, o laço será interrompido. Caso a condição de parada nunca seja atingida, o loop será executado infinitamente, gerando problemas no programa.

Estruturas de Repetição

https://pythonacademy.com.br/blog/estruturas-de-repeticao

http://curso.grupysanca.com.br/pt/latest/repeticao.html

Exercícios

Faça um algoritmo que leia os valores A, B, C e imprima na tela se a soma de A + B é menor que C.

Faça um algoritmo para receber um número qualquer e informar na tela se é par ou ímpar.

Faça um algoritmo que leia dois valores inteiros A e B se os valores forem iguais deverá somar os dois, caso contrário multiplique A por B. Ao final de qualquer um dos cálculos deve-se atribuir o resultado para uma variável C e mostrar seu conteúdo na tela.

Escreva um algoritmo que leia três valores inteiros e diferentes e mostre-os em ordem decrescente.

Escrever um algoritmo que leia o nome e as três notas obtidas por um aluno durante o semestre. Calcular a sua média (aritmética), informar o nome e sua menção aprovado (media >= 7), Reprovado (media <= 5) e Recuperação (media entre 5.1 a 6.9)

Exercícios

O cardápio de uma lanchonete é o seguinte:

Especificação	Preço unitário
100 Cachorro quente	1,10
101 Bauru simples	1,30
102 Bauru c/ovo	1,50
103 Hamburger	1,10
104 Cheeseburger	1,30
105 Refrigerante	1,00

Escrever um algoritmo que leia o código do item pedido, a quantidade e calcule o valor a ser pago por aquele lanche. Considere que a cada execução somente será calculado um item.

Exercícios

1. Calcule a tabuada do 13.

2. Ler do teclado 10 números e imprima a quantidade de números entre 10 e 50.

3. Ler do teclado uma lista com 5 inteiros e imprimir o menor valor.

4. Calcule o somatório dos números de 1 a 100 e imprima o resultado.

5. Receba dois números inteiros correspondentes à largura e altura. Devolva uma cadeia de caracteres # que representa um retângulo com as medidas fornecidas.

6. Ler do teclado um número inteiro e imprimir se ele é primo ou não.

Obrigado!!

Senac Fecomércio Sesc

Referências

1. https://sites.google.com/site/esdicapsi/operadores
2. http://www.nce.ufrj.br/ginape/js/conteudo/variaveis/operadores.htm
3. http://www.bosontreinamentos.com.br/logica-de-programacao/resumo-basico-de-operadores-em-programacao/
4. https://www.w3schools.com/python/python_datatypes.asp
5. https://embarcados.com.br/tipos-de-dados/
6. https://www.devmedia.com.br/estruturas-de-condicao-em-python/37158
7. https://www.devmedia.com.br/estruturas-de-repeticao-em-python/41551
8. https://pythonacademy.com.br/blog/estruturas-de-repeticao
9. https://docente.ifrn.edu.br/jonathanpereira/disciplinas/algoritmos/lista-de-exercicios-2/view

Senac Fecomércio Sesc