Fundamentos de Programação

Conceitos Básicos de PROGRAMAÇÃO

Tipos de Dados

Aula 03

Post It da Aula Passada

Projetando um Programa

- Delimitação do problema
  - Um problema complexo
  - Vários problemas simples
- Entradas, dados, processos, recorrências e resultados

O Primeiro Programa: *"Hello World!"*

- Arquivos e projetos
- Comentários
- Comando de exibir na tela (*print*)
- Linha de código Vs. linha do programa
- Execução do programa

Dados

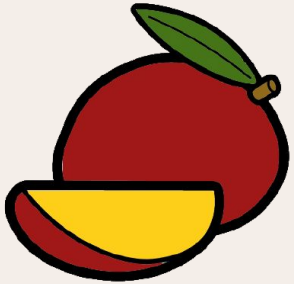
O que são dados?

Conteúdo que, sozinho, não apresenta contexto ou situação

- Um valor atribuído a algo (não a coisa em si)
  - O processamento e interpretação dos dados levam a compreensão destes como uma informação
  - A associação e depuração de informações leva ao entendimento das mesmas como conhecimento

Dados

MANGA



Fruta

Camiseta

D’água

Mangueira

Tipos de Dados

Dependendo de seu conteúdo, um dado é identificado como de um tipo ou outro. O tipo é uma forma de classificar os dados

Em linguagens de programação existem tipos de dados primitivos e derivados (ou customizados)

- **Tipos primitivos**: são tipos de dados *built-in* que não podem ser decompostos em outros tipos de dados
- **Tipos derivados**: são tipos de dados criados a partir dos tipos primitivos, gerando estruturas e classes

Tipos de Dados

É importante ressaltar que, ao nível de máquina, existe apenas um tipo de dados: **bit (BInary digiT)**

Um bit representa apenas um valor entre duas opções (0 ou 1) – ou seja, são possibilidades mutuamente exclusivas

Ao nível de máquina, um bit é representado por uma carga elétrica

Entretanto, normalmente, o bit não é um tipo manipulável de forma isolada ao nível de linguagens de programação

Tipos de Dados

Inicialmente, nosso foco será dado aos tipos primitivos mais comuns, que existem para a maioria das linguagens de programação — inclusive no Python!

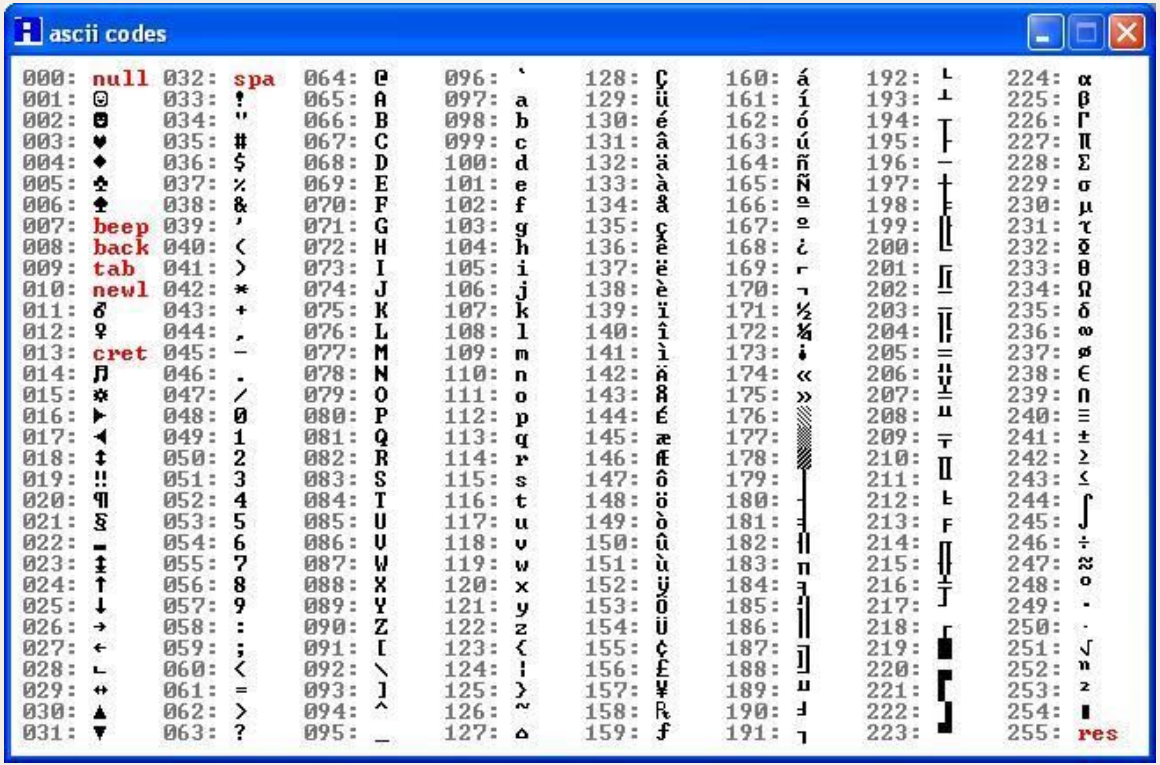
Sendo assim, os tipos primitivos que serão explorados neste momento são:

- Byte
- Lógico
- Inteiro
- Real
- Caractere/Cadeia

Tipos de Dados: Byte

- Um byte é uma unidade de armazenamento de **8 bits**
  - Com isso, pode assumir **256 valores** diferentes
- Tipicamente, é a unidade básica de manipulação de dados das linguagens de programação
- A adoção do byte como um octeto de bits não teve um motivo especial
  - O maior motivo era, nos primórdios da computação, o equilíbrio entre capacidade de representação de dados e de armazenamento dos mesmos
  - Padronização de representação em tabelas (**ASCII**)

Tipos de Dados: Byte



Tipos de Dados: Byte

Na linguagem Python, podemos expressar um byte ou uma cadeia de bytes escrevendo os símbolos desejados entre o **marcador b’ ’**

Por exemplo:

- **1** = Significa o valor numérico de 1 e todas as suas propriedades
- **b‘1’** = Significa o valor do símbolo ‘1’ na Tabela ASCII (49)

O tipo byte é utilizado como padrão de envio e recebimento de dados em Python.

Tipos de Dados: Lógico

Um valor lógico representa uma **escolha binária exclusiva**: verdadeiro ou falso; sim ou não; ausente ou presente; 0 e 1...

Por padrão, a maioria das linguagens de programação adotam os termos “verdadeiro” (“True”) e “falso” (“False”) para a manipulação do valor lógico

Frequentemente, o tipo lógico é chamado bool, boolean ou booleano em referência ao matemático inglês George Boole

Tipos de Dados: Lógico

Na linguagem Python, as palavras-chave (que são reservadas) utilizadas para o tipo lógico (que é chamado de **bool**) são **True** e **False**.

O tipo lógico é majoritariamente utilizado em operações lógicas, criação de condicionais, operações de bit, entre outros...

Abordaremos concretamente a utilização do tipo lógico nas próximas aulas!

Tipos de Dados: Inteiro

O tipo inteiro compreende **dados numéricos positivos e negativos**. Dos inteiros, são excluídos quaisquer valores fracionários

0; 10; -36; 40; 68; -24; -1; 1....

De forma geral, o tipo inteiro é utilizado em operações aritméticas, relacionais e lógicas

Tipos de Dados: Inteiro

Na linguagem Python, não há nenhuma notação especial para a definição de um número inteiro: **basta digitar o próprio número**

Além disso, dentro da linguagem, o tipo é chamado pela sua abreviação: **int**

Números negativos, entretanto, são sinalizados com um hífen (-), o qual assume o papel de um sinal negativo

Tipos de Dados: Real

A este tipo atribuem-se os **dados numéricos positivos e negativos pertencentes ao conjunto de números reais**. Portanto, são incluídos tanto valores inteiros quanto fracionários

-10,5; 1,92; 100,0; 0,0; 0,258; -1,5; 3,1416...

Assim como o tipo inteiro, o tipo de dados real é comumente utilizado em operações aritméticas, relacionais e lógicas

Tipos de Dados: Real

Em Python, é importante observar a troca entre o símbolo de separação decimal e o símbolo de separação de milhar: ponto (.) é utilizado nos decimais

1,85 (decimal em português) -> 1.85 (decimal em inglês)

Números reais são tipicamente chamados ponto flutuante (**float**) em linguagens de programação

As demais regras se assemelham às do tipo inteiro

Tipos de Dados: Caractere/Cadeia

Um caractere (ou uma cadeia de caracteres) consiste em **uma sequência de códigos ASCII** (nativamente) digitáveis ou não. Esse tipo é comumente chamado ***string* (str)**

Em geral, *strings* se comportam de maneira muito semelhante ao tipo byte, porém sempre tem uma codificação explícita envolvida: é preparado para a leitura por máquinas

Já o tipo *string* é utilizado majoritariamente como um recurso preparado para a leitura por humanos

Tipos de Dados: Caractere/Cadeia

Em geral, *strings* são representadas em linguagens de programação por meio de **caracteres escritos entre aspas duplas (“ ”)**. Entretanto, Python suporta a definição de *strings* também entre aspas simples (‘ ’)

Sendo assim:

- 1 = Significa o valor numérico de 1
- b‘1’ = Significa o valor do símbolo ‘1’ na Tabela ASCII (49)
- ‘1’ ou “1” = Significa o caractere ‘1’

Tipos de Dados: Outros

POLÊMICO

Os tipos deste *slide* não são unanimemente considerados primitivos!

Alguns tipos de dados são providos naturalmente como primitivas da linguagem Python, porém, em outras linguagens, os mesmos são tipos derivados

- Tupla (*tuple*)
- Lista (*list*)
- Dicionário (*dictionary*)

Identificando Tipos em Python

Se contamos com toda uma variedade de tipos de dados, como podemos saber com qual tipo estamos lidando em um programa?

Podemos identificar o tipo de um dado sendo manipulado através da função *type*!

type(dado**)**

A função type retornará como resultado o tipo do dado enviado como argumento (entre parênteses)

Identificando Tipos em Python

Qual é o resultado retornado para as linhas de código apresentadas a seguir?

- type(155)
- type(“155”)
- type(155.0)
- type(b’155’)

Como **exibir na tela o resultado da função *type*** em um programa escrito em Python?

Identificando Tipos em Python

Qual é o resultado retornado para as linhas de código apresentadas a seguir?

- type(155)
- type(“155”)
- type(155.0)
- type(b’155’)

Como **exibir na tela o resultado da função *type*** em um programa escrito em Python?

```
print(type(dado))
```

Convertendo Tipos de Dados

A conversão de tipos (*casting*) nada mais é do que o **processo de transformação de um tipo em outro**

Alguns tipos de dados apresentam compatibilidade semântica!

- “1” tem compatibilidade semântica com o int 1 (e vice-versa)
- “1” tem compatibilidade semântica com o *float* 1.0
- 1.0 tem compatibilidade semântica com a *string* “1.0”
- “1” não tem compatibilidade semântica com o bytes b‘1’

Convertendo Tipos de Dados

Em Python, castings são feitos explicitando para qual tipo de dados um determinado dado deve ser convertido

Isto é feito utilizando o próprio nome do tipo de dados alvo como uma função:

- str(1) -> Casting de um int para str
- float(5) -> Casting de um int para float
- int(5.7) -> Casting de um float para int (funciona?)
- str(b‘a’) -> Casting de um bytes para str (funciona?)
- bytes(“a”) -> Casting de uma str para bytes (funciona?)

Exercício #03

Faça um algoritmo em Python que use a função **print()** para mostrar cada dado e o seu tipo (função **type()**):

- 15
- 2.5
- “ABC”
- False

Depois, faça o *casting* para inteiro (int) dos dados anteriores com compatibilidade semântica para tal, exibindo o resultado e o tipo de dados resultante na tela

Obs.: utilize vírgulas para um *print* composto — print(10, “abc”) -> 10 abc

Fundamentos de Programação

Aula 03

Obrigado e ATÉ A PRÓXIMA!

©Prof. Vinicius Fulber-Garcia - 2025