

Algoritmos e Estruturas de Dados

Aula 01 - Introdução

Prof. Victor Lundgren



Sumário



01

Introdução

Objetivos, relevância e
motivação

02

Tipos de Dados

Tipos e variáveis

03

Recursão

Funções recursivas,
custo-benefício



01 Introdução

Objetivos, relevância e
motivação



Objetivos



- Investigar formas de armazenamento de dados em programas de computador;
- Entender como armazenar e buscar dados;
- Estudar algoritmos que manipulam diferentes estruturas de dados;
- Dar uma noção de como avaliar algoritmos em relação a sua eficiência;
- Praticar o aprendizado em problemas práticos de programação (Python);

Relevância

- Manipular informação do computador é uma atividade crítica
- Algoritmos usando diferentes estruturas podem ter resultados completamente distintos
- Prática do raciocínio algorítmico
- Aplicação de orientação a objetos



02

Tipos de Dados

Tipos e variáveis





Informação

- Computadores manipulam dados;
 - Dados X informação.
- Unidade mais básica de informação é o **bit**;
 - 0 ou 1, N chaves representam 2^N valores.





- Dados Primitivos:
 - Inteiros;
 - Reais;
 - Caracteres.
- Dados Compostos:
 - Estruturas (objetos);
 - Strings;
 - Array (vetores);



Inteiros

Representação Positiva

Para computadores (bits) em binário.

- $10 = 2 (0*1 + 1*2)$
- $100110 = 38 (1*0 + 2*1 + 4*1 + 8*0 + 16*0 + 32*1)$

N bits de representam um inteiro único, de **0** a **$2^n - 1$**

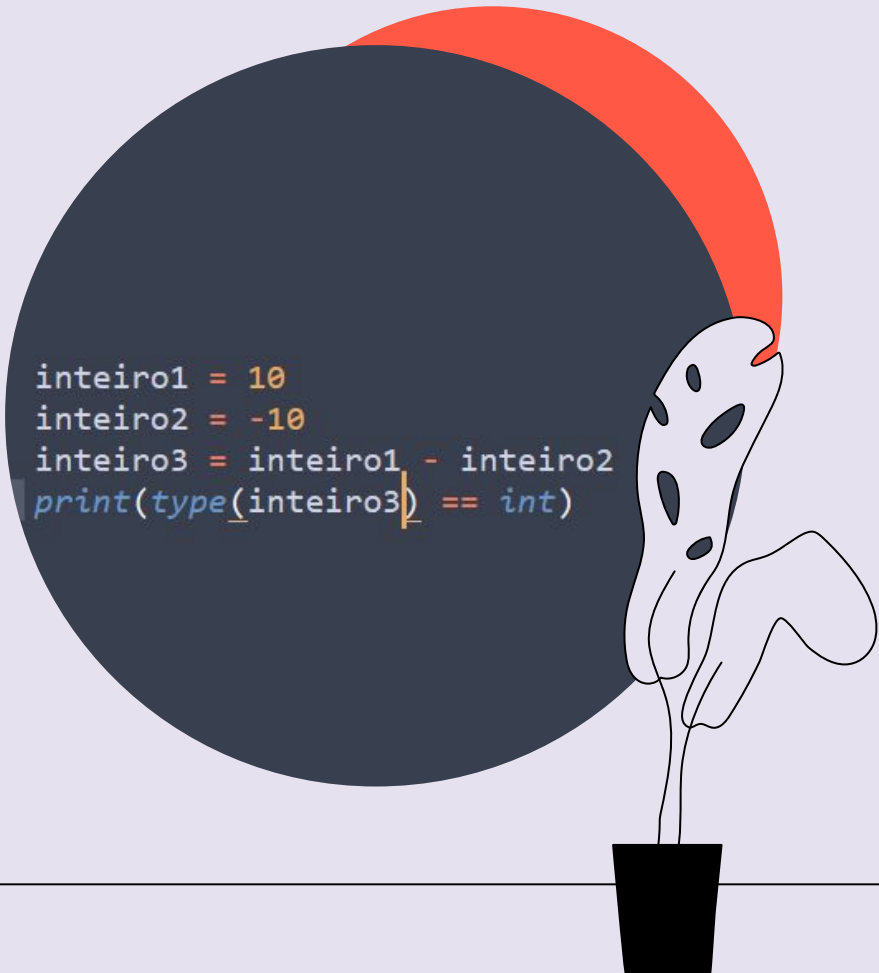
Representação Negativa

Complemento de 1, inversão dos bits:

- $0010 = 2 \mid 1101 = -2$

Complemento de 2, inversão dos bits + 1:

- $0010 = 2 \mid 1110 = -2$



```
inteiro1 = 10
inteiro2 = -10
inteiro3 = inteiro1 - inteiro2
print(type(inteiro3) == int)
```

Reais

Notação de Ponto Flutuante

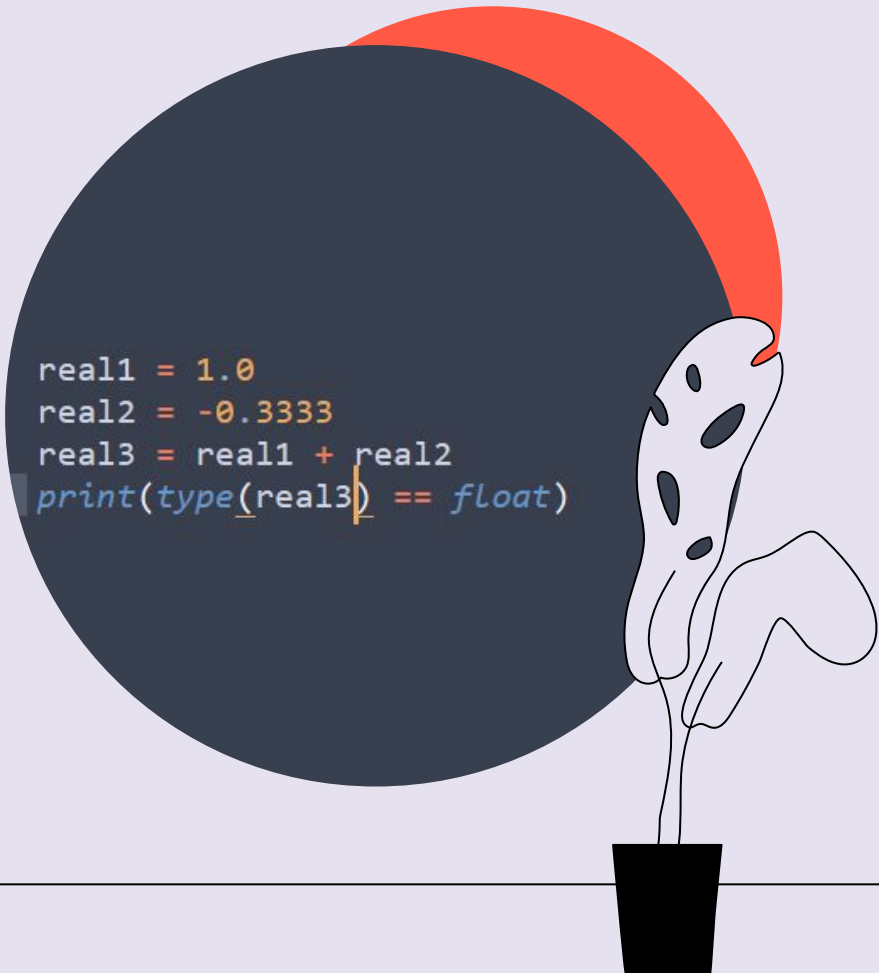
Mantissa x Base elevada a potência inteira

- $387,53 = 38753 * 10^{-2}$

Em representações de 32 bits, mantissa possui 24 bits e potência possui 8. Logo para 387,53:

- Em 24 bits = 00000000100101101100001
- Em 8 bits = 11111110
- = 0000000010010110110000111111110

Tal representação nos permite armazenar números muito grandes (ou pequenos).



```
real1 = 1.0
real2 = -0.3333
real3 = real1 + real2
print(type(real3) == float)
```

Em base binária?

$$38_{10} = ???$$



Caracteres

Valores Não-Numéricos

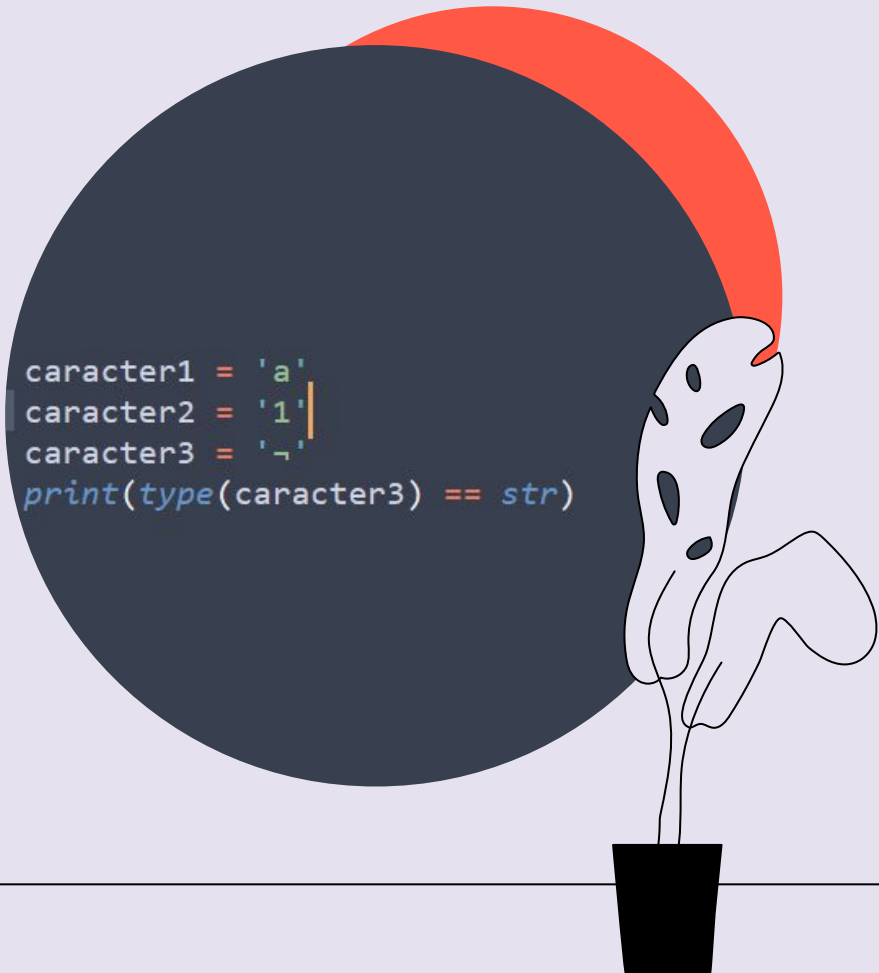
Um número pré-determinado de bits nos permite representar um código por referência (ASCII, Unicode, etc.).

- 1 ou 2 bytes (8 ou 16 bits).

Primitivos ou built-ins?

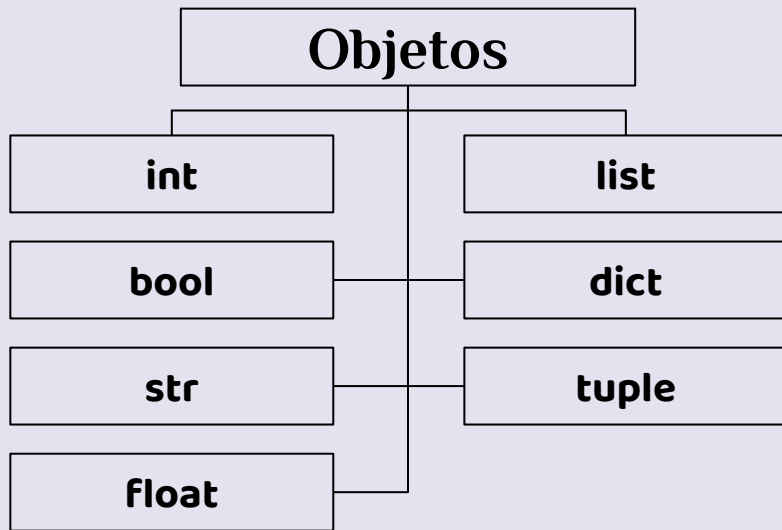
A maioria das linguagens define tipos de dados básicos, chamados **tipos primitivos**, as formas mais puras para a linguagem.

Em Python **não** é bem assim.



```
character1 = 'a'  
character2 = '1'  
character3 = '¬'  
print(type(character3) == str)
```


Tipos Built-ins



Tipos built-ins são objetos existentes dentro do interpretador Python.

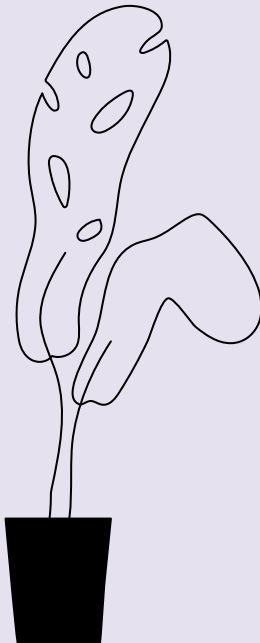
Não existe tamanho máximo para valores, podem expandir enquanto houver memória.

Tipos Built-ins



```
import sys
d = {
    "int": 0,
    "float": 0.0,
    "dict": dict(),
    "tuple": tuple(),
    "list": list(),
    "str": "a",
    "object": object(),
}
for k, v in d.items():
    print(k, type(v), sys.getsizeof(v))
```

Verificar os tamanhos de memória alocados para cada tipo de dado em Python, em bytes.



Variáveis

Guardam valores

De um tipo específico.

Em Python, temos tipagem forte e dinâmica.

Espaço de Memória

Um espaço de memória referenciada é reservado para aquela variável.

Identificador

Temos a comodidade de acessar o espaço reservado pelo identificado, não pelo endereçamento de memória



03

Recursão

Funções recursivas,
custo-benefício

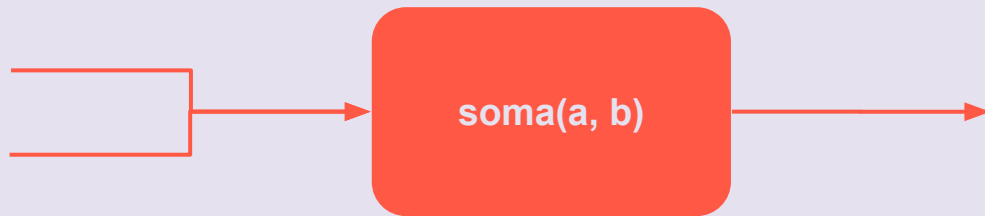


Funções Regulares



Funções regulares são compostas por:

- Declaração (inclui parâmetros);
 - Corpo;
 - Retorno;
 - Chamada (inclui argumentos);
- `def soma(a, b):`
 - `c = a + b`
 - `return c`
 - `soma(1, -1)`

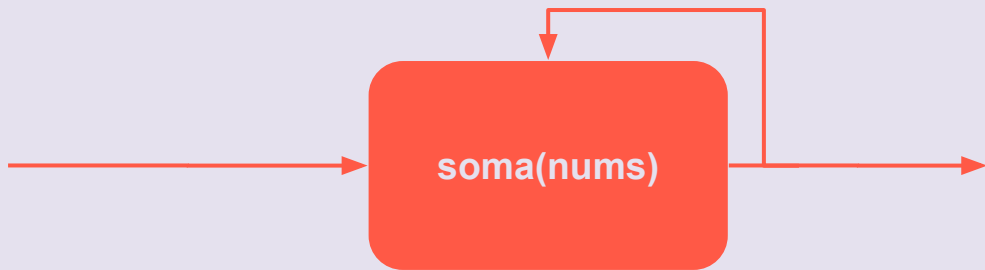


Funções Recursivas



Funções recursivas funcionam da mesma maneira, mas possuem chamada direta ou indireta à mesma função em seu corpo:

- Declaração (inclui parâmetros);
 - Corpo;
 - Retorno;
 - Chamada (inclui argumentos);
- `def soma(nums):`
 - `c = nums[0] + soma(nums[1:]) if len(nums) > 0 else 0`
 - `return c`
 - `soma([1, -1])`



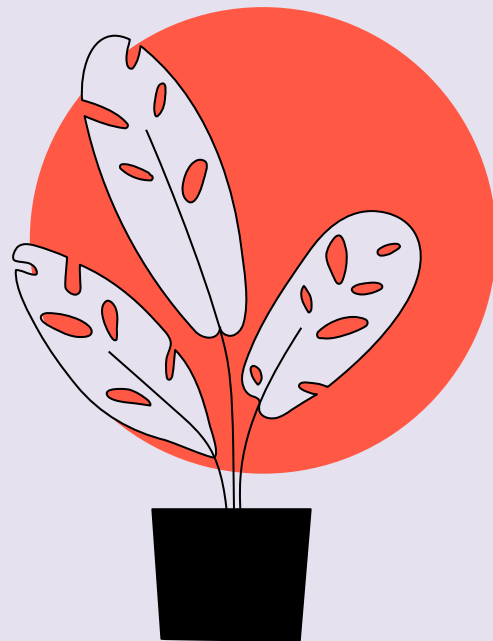
Funções Recursivas

Uma função pode ter uma definição indireta. Como escrever a função abaixo em Python?

- $f(0) = 0$
- $f(x) = 2f(x-1) + x^2$

É necessário a existência de um caso base e da chamada recursiva.

- Como escrever uma função de fatorial recursiva?
- $4! = 4 * 3 * 2 * 1 = 24$
- Sabemos que: $n! = n * (n-1)!$



Funções Recursivas

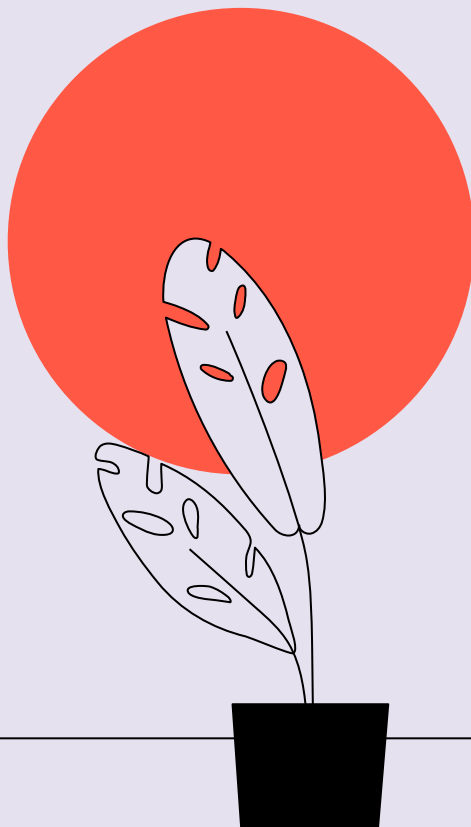


Com Iteração

```
def calc_fatorial(x):  
    fatorial = 1  
    for x in range(x, 0, -1):  
        fatorial *= x  
    return fatorial
```

Com Recursão

```
def calc_fatorial(x):  
    if x > 0:  
        return x * calc_fatorial(x-1)  
    else:  
        return 1
```



Funções Recursivas

Terminação



É necessário um caso base
que seja resolvido sem o
uso de recursividade.

Estouro de Pilha



Caso contrário a
recursividade entrará em
loop infinito, causando
estouro de pilha



Funções Recursivas



Eficiência

Métodos recursivos costumam ser menos eficientes



Frequência

Também é importante entender o quão comum é a chamada da função



Praticidade

Recursividade pode ser a maneira mais lógica de resolver



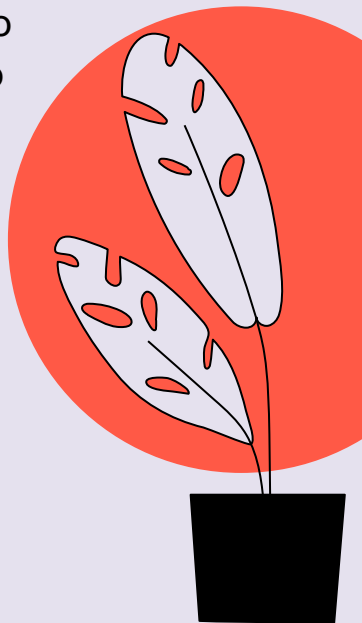
Complexidade

É importante ponderar o custo da máquina com o de programa



Questão Final

Por fim, é uma questão de problema x programador



Exercícios



Exercício 1

Escrever um programa em Python usando recursão que calcula o valor na série de Fibonacci, do enésimo número:

- $\text{fib}(n) = n$, se $n==0$ ou $n==1$
- $\text{fib}(n) = \text{fib}(n-2) + \text{fib}(n-1)$, se $n \geq 2$

Exercício 2

Método para imprimir um número inteiro não-negativo n , dado que só se pode imprimir um algarismo por vez.

Exercício 3

Método para calcular o m.d.c de dois inteiros (dica: usar %)

Exercícios

Listas de exercícios a cada semana

- Submissão e disponibilização pelo
 - <http://www.hackerrank.com/>





CREDITS: This presentation template was created by
Slidesgo, including icons by **Flaticon**, infographics & images
by **Freepik**