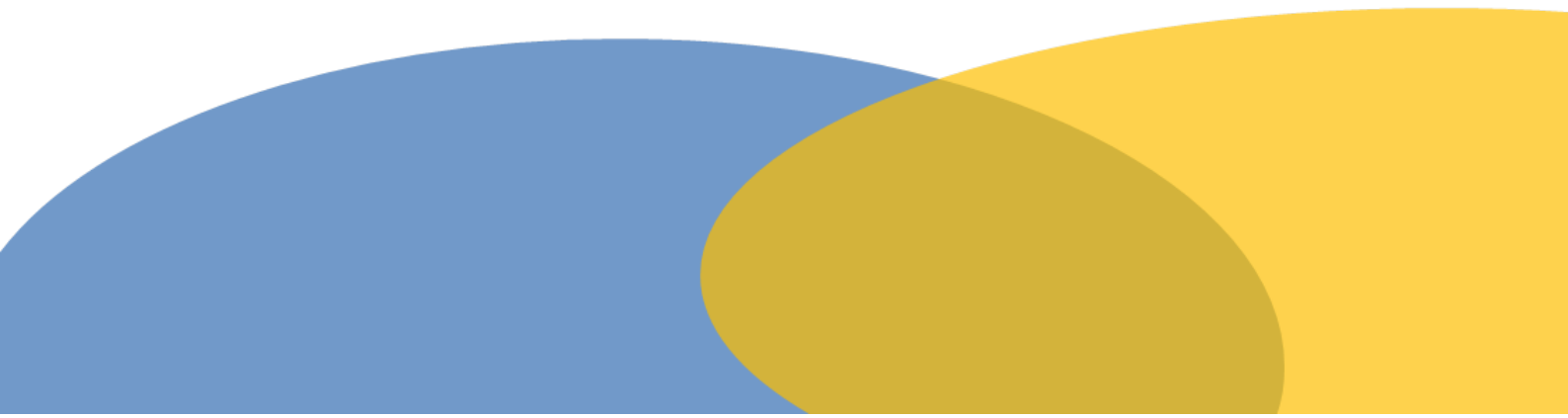
Introdução à Linguagem Python

Paradigmas de Linguagens de Programação

Rômulo Souza Fernandes
Ausberto S. Castro Vera

26 de setembro de 2022



UENF - UNIVERSIDADE ESTADUAL DO NORTE FLUMINENSE DARCY RIBEIRO

CCT - CENTRO DE CIÊNCIA E TECNOLOGIA
LCMAT - LABORATÓRIO DE MATEMÁTICAS
CC - CURSO DE CIÊNCIA DA COMPUTAÇÃO

Primeira edição, Setembro 2022

Sumário

1. Introdução

O Python é uma linguagem orientada a objetos de alto nível, que possui uma sintaxe simples e objetiva, assim colaborando para a fácil compreensão do código-fonte e permitindo que a linguagem seja produtiva. O Python contém várias estruturas de alto nível, como hora, data, dicionários, listas, complexos, entre outras estruturas. Contém um amplo conjunto de módulos disponíveis para utilização, frameworks que podem ser acrescentados, ferramentas de outras linguagens atuais, como persistência, unidades de teste, geradores, introspecção e metaclasse, além de ter disponíveis diversas bibliotecas, como IPython, Matplotlib, mIPy, NumPy, Pandas, SciPy, ScraPy, entre outras bibliotecas conhecidas.

O Python é uma linguagem multiparadigma, suportando a programação orientada a objetos, modular e funcional. A linguagem Python foi criada na Holanda, no ano de 1990, por Guido van Rossum, no Instituto Nacional de Pesquisa para Matemática e Ciência da Computação. [Bor14]

A linguagem Python é de código aberto, porém o criador Guido van Rossum possui a função central de decidir a evolução da linguagem. O Python se popularizou e se tornou a linguagem de desenvolvimento de aplicações mais indicada para iniciantes, assim sendo aconselhada como primeira linguagem de programação. [Per16]

1.1 História da linguagem Python

O intuito de Guido van Rossum era criar uma linguagem que pudesse suprir suas exigências, assim criando o Python, com base na linguagem ABC, mas solucionando os problemas encontrados por ele na linguagem. O Python tinha como usuários principais os engenheiros e físicos.

A seguir um pouco da história da linguagem Python, baseados em [Per16] e [Bor14] :

- O Holandês Guido van Rossum foi o autor principal da linguagem Python. O autor trabalhava no CWI (Centrum Wiskunde & Informatica), localizada em Amsterdã na Holanda.
- O nome Python não veio da espécie de serpente e sim do seriado de comédia preferido do autor da linguagem, chamado Monty Python's Flying Circus.
- A versão 0.9.0 do Python foi lançado em 1991, incluindo manipulação de exceções, classes, listas e strings. Incluía também alguns aspectos de programação funcional como lambda, maps, filter e reduce.

- No ano de 1995, o autor da linguagem continuou seu trabalho sobre Python na Corporation for National Research Initiatives (CNRI) em Reston, Virginia, USA.
- Em Maio de 2000, Guido van Rossum e o grupo de desenvolvimento do Python se mudaram para BeOpen.com, assim formando a equipe BeOpen PythonLabs.
- A versão 1.6 do Python foi lançada em 5 de setembro de 2000.
- A versão 2.0 do Python foi lançada em 16 de outubro de 2000.
- A versão 3.0 do Python foi lançada em 3 dezembro de 2008.

1.2 Áreas de Aplicação da Linguagem

O Python está entre as linguagens de programação mais utilizadas no mundo, é muito utilizado por usuários individuais, mas sua aplicação se estende para empresas reais. A natureza do Python é de propósito geral, assim tornando a linguagem aplicável em quase todas as áreas. Como a IBM, Seagate e Hewlett-Packard, que utilizam o Python para testes de hardware, o Yahoo! e Google usam a linguagem em serviços de Internet, já a empresa Industrial Light and Magic e outras empresas de filmes utilizam o Python na produção de animações. Entre todas as aplicações para o Python atualmente, o ponto em comum é que a linguagem é usada em todo o espectro, em questão de domínios de aplicação. [Lut07]

1.2.1 Big Data

Atualmente o Python é uma das melhores linguagens de programação para trabalhar com Big Data. Um dos motivos dessa preferência de uso é o suporte avançado de inúmeras bibliotecas e frameworks, muitas das bibliotecas são voltadas para lidar com Big Data, dando suporte e auxiliando na implementação de algoritmos de Machine Learning e Data Analytics. Abaixo algumas das bibliotecas de software livre:

- SciPy: Utilizada para computação técnica e computação científica, possibilita a interpolação, otimização, integração e modificação de dados utilizando funções especiais, álgebra linear, etc.
- NumPy: Utilizada para computação numérica para dados com formas de grandes matrizes multidimensionais e Arrays. A biblioteca também oferece diversas funções matemáticas de alto nível, para manipular os dados com transformada de Fourier, álgebra linear, processamento de números aleatórios, etc.
- Scikit-learn: Utilizada para Machine Learning, relacionada a vários algoritmos de regressão, clustering e classificação. Pode ser utilizada também em conjunto com outras bibliotecas, como a NumPy e SciPy.
- Pandas: Utilizada para análise e manipulação de dados, oferece diversas estruturas de dados e operações para manipulação de dados, no formato de séries temporais e tabelas numéricas. A biblioteca também dispõe de diferentes ferramentas para gravar e ler dados, entre estruturas de dados na memória e diferentes formatos de arquivo.

O Python possui uma sintaxe simples, possibilitando uma fácil leitura do código, assim tanto os iniciantes quanto os desenvolvedores experientes, podem se concentrar melhor no objetivo, ao invés de se desgastar se concentrando nas nuances técnicas da linguagem que está utilizando. Sendo assim, o Python é a linguagem preferida dos Cientistas de dados e Engenheiros de Big Data.

A linguagem Python é extremamente flexível, permitindo finalizar mais trabalhos com menor número de linhas de código. O Python também é escalável na manipulação de dados em grandes quantidades, sendo um ponto muito importante quando se trata de Big Data. Comparando o Python com outras linguagens de programação utilizadas em Big Data Analytics, como R e Java, elas não são tão escaláveis e flexíveis como o Python, onde havendo um aumento no volume de dados, o Python sem dificuldades pode aumentar a velocidade de processamento dos dados, sendo uma

tarefa complicada para fazer em R ou Java. [McK19]

1.2.2 Orientação a objetos

O Python é uma linguagem multiparadigma, suportando diferentes abordagens de programação, um jeito de solucionar problemas de programação é criar objetos, esse método é conhecido como Programação Orientada a Objetos(POO) e a orientação a objetos é um dos paradigmas da linguagem Python, com isso, a criação de objetos e classes é mais simples.

No Python os dados são guardados em objetos, em outras linguagens, determinados tipos são guardados na memória, não em entidades abstratas como objetos. A programação orientada a objetos é um importante método de organizar e desenvolver códigos, focando em criar códigos reutilizáveis. [Lut07]

No Python, a programação orientada a objetos possui alguns pontos importantes, baseados no mesmo autor citado no parágrafo anterior:

- Encapsulamento: Restrição ao acesso de métodos e atributos de uma classe, evitando que os dados sejam alterados diretamente, na linguagem Python existe apenas o private e public.
- Métodos: São funções definidas no corpo da classe. Utilizados para definir o comportamento do objeto.
- Objeto: É a instância de uma classe. Somente a definição do objeto é definida no momento em que a classe é estabelecida. Consequentemente nenhum espaço de memória é alocado.
- Classe: Juntam as funcionalidades de um certo objeto, servindo como um modelo.
- Polimorfismo: É a capacidade de utilizar uma interface para vários tipos de dados. Possibilitando que o objeto possua o poder de assumir diversas formas.
- Herança: Cria uma classe nova que será descendente e irá utilizar particularidades de outra classe já existente, sem realizar modificações na mesma.

1.2.3 Pentest

O Python está entre as melhores linguagens para pentest e segurança da informação, isso é devido a grande quantidade de bibliotecas, ferramentas, frameworks, por ter versatilidade e ser multiplataforma, entre outros pontos que tornam o Python uma das melhores linguagens para essa aplicação, como ter um código de fácil leitura e sintaxe simples. A linguagem é usada para diversas finalidades dentro do processo, possibilitando que as soluções sejam vinculadas e automatizadas sem dificuldades, como decodificar e enviar pacotes, varrer redes e portas, analisar malwares, acessar servidores, etc, com base em [Mor18] e [Sei15].

Como citado, a linguagem Python possui uma grande diversidade de ferramentas, de acordo com os mesmos autores citados no parágrafo anterior, essas são algumas das ferramentas disponíveis:

- Análise de malware:
  - Exefilter: Utilizada para filtrar tipos de arquivos para páginas web e e-mails, podendo detectar diversos tipos de arquivos e apagar o conteúdo ativo.
  - PyClamAV: Acrescenta a detecção de vírus maliciosos nas ferramentas do Python.
  - Pyew: Utilizada geralmente para analisar malwares, o pyew desmonta e edita hexadecimais de linha de comando.
- Utilitários de rede:
  - Dpkt: Utilizado para gerar e analisar pacotes de dados utilizando definições do protocolo TCP/IP.
  - Spoodle: Usado para verificar subdomínios e vulnerabilidade.
  - Knock Subdomain Scan: utilizado para retornar a lista de subdomínios do domínio de destino através da técnica de lista de palavras.
- Explorar bibliotecas:
  - Scapy: É uma biblioteca e ferramenta de processamento de pacotes, podendo decodificar

diversos protocolos, enviar pela rede, capturar e combinar respostas e solicitações. Oferece funções como as oferecidas pelo Tcpdump, Wireshark e Nmap, também oferece acesso programático.
  - Python Nmap: Usado para analisar os resultados da varredura do Nmap e lançar ataques particularizados contra hosts específicos.
  - Monda: O Monda é um depurador de imunidade, que auxilia no desenvolvimento de programas de exploração.
- Forense
  - Rekall: O Rekall é um framework criado pelo Google para realizar varreduras e análises de memória.
  - LibForensics: É uma biblioteca para desenvolvimento de aplicativos Forenses digitais
  - Aft: É um pacote de ferramentas forenses voltado para Android.

2. Conceitos básicos da Linguagem Python

Neste capítulo é apresentado alguns conceitos básicos da linguagem de programação Python, como Variáveis, constantes e tipos de dados básicos aceitos pela linguagem, que são inteiro, ponto flutuante, booleano, string e lista. Alguns dos livros indicados para iniciar o estudo sobre a linguagem de programação Python são: [Lut07], [Per16], que foram os mesmos autores usamos como base para escrever grande parte desse e outros capítulos.

2.1 Variáveis e constantes

De acordo com [Sev16] o Python possui um ótimo recurso que é a manipulação de variáveis, essas variáveis são nomes atribuídos a valores, que possuem como propósito armazenar valores de forma que mais tarde possam ser recuperados. Para criar variáveis é necessário fazer uma declaração por atribuição e atribuir valores a essas novas variáveis.

```
>>> texto = 'Hello, world!'
>>> pi = 3,14159265
>>> numero = 1500
```

No exemplo acima podemos observar a declaração por atribuição de 3 variáveis diferentes. A primeira está atribui uma string para uma variável chamada texto, a segunda variável atribui o valor 3,14 que é conhecido como Pi e a variável possui o nome pi. Já a terceira variável numero está recebendo o valor 1500, também por atribuição. Os usuários tem uma grande liberdade na hora de escolher os nomes das variáveis, apenas não sendo possível usar palavras reservadas da linguagem como nome de uma variável.

As constantes na linguagem Python, diferente de outras linguagens de programação, não podem ser criadas de forma que seu valor não seja alterado. Na documentação existem algumas orientações caso o usuário queira criar uma constante com sintaxe de variável, uma delas é que todas as letras da variável que será utilizada como constante, deveram ser maiúsculas, em casos do nome desejado possuir espaço, deverá ser utilizado underline.

```
>>> PRECO = 5
>>> PRECO_PRODUTO = 2
```

O exemplo acima é o padrão recomendado pela documentação do Python.

2.2 Tipos de Dados Básicos

A linguagem de programação Python é de tipagem dinâmica, com isso não é preciso declarar o tipo de variável, o tipo será definido através do valor que a variável receber, isso possibilita que o tipo mude no decorrer da execução do programa. Vamos falar sobre esses tipos de dados a seguir, com base no autor [Sev16].

2.2.1 String

Resumidamente, uma string é uma sequência de caracteres, sendo classificado como um item de dado simples. Para o Python, uma string é um array de caracteres ou qualquer grupo de caracteres escritos entre doble aspas ou aspas simples, por exemplo:

```
>>> #Aspas simples
>>> menssagem1 = 'Hello, World'
>>> print (menssagem1)
Hello, World

>>> #Aspas duplas
>>> mensagem2 = "O dia esta chuvoso"
>>> print (menssagem2)
O dia esta chuvoso
```

- *Concatenação de strings*
  A união de strings é chamado de concatenação, isso pode ser feito utilizando o operador +, que possui essa função de concatenar quando usado com operandos do tipo string. O comprimento de uma string pode ser calculado utilizado o operador `len(string)`.

```
>>> # concatenando 2 strings
>>> moto = "Titan " + "150 ESD"
>>> print (moto)
Titan 150 ESD

>>> print (len(moto))
13
```

- *Operador de indexação*
  Utilizando o operador de indexação é possível acessar os caracteres um por um, utilizando o operador colchetes, o número dentro do colchetes é denominado index, usado para indicar a posição do caractere da variável e atribuir esse caractere à uma variável. Existem duas formas de indexar os caracteres de um string em Python:

  Index com inteiros positivos indexando a partir da esquerda, começando com 0, sendo o 0 o index do primeiro caractere da sequência.

  Index com inteiros negativos indexando a partir da direita, começando com -1, sendo -1 o último elemento da sequência, -2 sendo o penúltimo elemento da sequência, e assim sucessivamente.

```
>>> #Acessar caracteres
>>> moto = 'titan'
>>> letra = moto[0]
```

```
>>> print(letra)
t

>>> # Indexando strings
>>> pyStr = "Programando"
>>> print (len(pyStr))
11
>>> print (pyStr)
Brasil verde amarelo
```

- *Operador de Fatias*
  O operador de acesso a itens (caracteres individuais) também pode ser utilizado como operador de fatias, para extrair uma fatia inteira (subsequência) de caracteres de um string. O operador de Fatias possui três sintaxes:
  `seq[ inicio ]`
  `seq[ inicio : fim ]`
  `seq[ início : fim : step ]`
  onde `início`, `fim` e `step` são números inteiros.

```
>>> # Indexando strings
>>> pyStr = "Programando Python"
>>> print (len(pyStr))
11
>>> print (pyStr)
Brasil verde amarelo
```

2.2.2 Lista

Em muitas situações, organizamos os dados em uma lista: uma lista de compras, uma lista de cursos, uma lista de contatos no seu telefone celular, uma lista de canções no seu player de áudio e assim por diante. Em Python, as listas normalmente são armazenadas em um tipo de objeto denominado lista. Uma lista é uma sequência de objetos. Os objetos podem ser de qualquer tipo: números, strings e até mesmo outras listas. Por exemplo, veja como atribuiríamos a variável animais à lista de strings que representa diversos animais:

```
>>> animais = ['peixe ', 'gato', 'cao']
```

A variável animais é avaliada como a lista:

```
>>> animais
['peixe', 'gato', 'cao']
```

Em Python, uma lista é representada como uma sequência de objetos separados por vírgulas, dentro de colchetes. Uma lista vazia é representada como []. As listas podem conter itens de diferentes tipos. Por exemplo, a lista chamada coisas em »> coisas = ['um', 2, [3, 4]] tem três itens: o primeiro é a string 'um', o segundo é o inteiro 2 e o terceiro item é a lista [3, 4]. Operadores de Lista A maioria dos operadores de string que vimos na seção anterior pode ser usada em listas de formas semelhantes. Por exemplo, os itens na lista podem ser acessados individualmente usando o operador de indexação, assim como os caracteres individuais podem ser acessados em uma string: »> animais[0] 'peixe' »> animais[2] 'cão' Figura 2.3 Uma lista de objetos de string. A lista animais é uma sequência de objetos. O primeiro objeto, no índice 0, é a string 'peixe'. Índices positivos e negativos podem ser usados, assim como para as strings. A Figura 2.3 ilustra a lista animais junto com a indexação dos itens da lista. Índices negativos também podem ser usados: »> animais[-1]

’cão’ O comprimento de uma lista (ou seja, o número de itens nela) é calculado usando a função len(): »> len(animais) 3 Assim como as strings, as listas podem ser “adicionadas”, significando que podem ser concatenadas. Elas também podem ser “multiplicadas” por um inteiro k, que significa que k cópias da lista são concatenadas: »> animais + animais [’peixe ’, ’gato’, ’cão’, ’peixe ’, ’gato’, ’cão’] »> animais * 2 [’peixe ’, ’gato’, ’cão’, ’peixe ’, ’gato’, ’cão’] Se você quiser verificar se a string ’coelho’ está na lista, pode usar o operador in em uma expressão Booleana que é avaliada como True se a string ’coelho’ aparecer na lista animais: »> ’coelho’ in animais False »> ’cão’ in animais True Na Tabela 2.2, resumimos o uso de alguns dos operadores de string. Incluímos na tabela as funções min(), max() e sum(), que podem apanhar uma lista como entrada e retornar, respectivamente, o menor item, o maior item, a soma dos itens da lista: »> lst = [23.99, 19.99, 34.50, 120.99] »> min(lst) 19.99 »> max(lst) 120.99 »> sum(lst) 199.46999999999997 Tabela 2.2 Operadores de lista e funções. Somente alguns dos operadores de lista comumente usados aparecem aqui. Para obter a lista completa no seu shell interativo, use a função de documentação help(): »>help(list) Uso Explicação x in lst Verdadeiro se o objeto x estiver na lista lst; caso contrário, falso x not in lst Falso se o objeto x estiver na lista lst; caso contrário, verdadeiro lstA + lstB Concatenação das listas lstA e lstB lst * n,n * lst Concatenação de n cópias da lista lst lst[i] Item no índice i da lista lst len(lst) Comprimento da lista lst min(lst) Menor item na lista lst max(lst) Maior item na lista lst sum(lst) Soma dos itens na lista lst

2.2.3 Inteiro

2.2.4 Ponto Flutuante

2.2.5 Booleano

2.3 Tipos de Dados de Coleção

2.3.1 Tipos Sequenciais

2.3.2 Tipos Conjunto

2.3.3 Tipos Mapeamento

2.4 Estrutura de Controle e Funções

2.4.1 O comando IF

2.4.2 Laço FOR

2.4.3 Laço WHILE

2.5 Módulos e pacotes

2.5.1 Módulos

2.5.2 Pacotes

Código fonte para a linguagem Python:

```
number_1 = int(input('Ingresse o primeiro numero: '))
number_2 = int(input('Ingresse o segundo numero: '))

# Soma
print('{} + {} = '.format(number_1, number_2))
print(number_1 + number_2)

# Substra\c{c}\~{a}o
print('{} - {} = '.format(number_1, number_2))
print(number_1 - number_2)

# Multiplica\c{c}\~{a}o
```

```
    print('{} * {} = '.format(number_1, number_2))
    print(number_1 * number_2)

    # Divis\~{a}o
    print('{} / {} = '.format(number_1, number_2))
    print(number_1 / number_2)
```

Referências Bibliográficas

[Bor14] Luiz Eduardo Borges. *Python para desenvolvedores: aborda Python 3.3*. Novatec Editora, Sao Paulo, SP, 2014. Citado na página 5.

[Lut07] M. Lutz. *Aprendendo Python*. Bookman, Porto Alegre, RS, 2007. Citado 3 vezes nas páginas 6, 7 e 9.

[McK19] Wes McKinney. *Python para analise de dados*. Novatec Editora, Sao Paulo, SP, May 2019. Citado na página 7.

[Mor18] Daniel Moreno. *Python para pentest*. Novatec Editora, Sao Paulo, SP, 2018. Citado na página 7.

[Per16] Ljubomir Perkovic. *Introducao a computacao usando Python: um foco no desenvolvimento de aplicacoes*. Rio de Janeiro, RJ, 2016. Citado 2 vezes nas páginas 5 e 9.

[Sei15] Justin Seitz. *Black Hat Python: Python Programming for Hackers and Pentesters*. No Starch Press, Sao Paulo, SP, 2015. Citado na página 7.

[Sev16] Dr. Charles Russell Severance. *Python for Everybody: Exploring Data in Python 3*. CreateSpace Independent Publishing Platform, Ann Arbor, MI, 1 edition, 2016. Citado 2 vezes nas páginas 9 e 10.

Disciplina: *Paradigmas de Linguagens de Programação 1970*
Linguagem: *LinguagemXYZabcd*
Aluno: *Rômulo Souza Fernandes*

Ficha de avaliação:

Aspectos de avaliação (requisitos mínimos)	**Pontos**
Introdução (Máximo: 01 pontos) • Aspectos históricos • Áreas de Aplicação da linguagem	
Elementos básicos da linguagem (Máximo: 01 pontos) • Sintaxe (variáveis, constantes, comandos, operações, etc.) • Cada elemento com exemplos (código e execução)	
Aspectos Avançados da linguagem (Máximo: 2,0 pontos) • Sintaxe (variáveis, constantes, comandos, operações, etc.) • Cada elemento com exemplos (código e execução) • Exemplos com fonte diferenciada (listing)	
Mínimo 5 Aplicações completas - Aplicações (Máximo : 2,0 pontos) • Uso de rotinas-funções-procedimentos, E/S formatadas • Uma Calculadora • Gráficos • Algoritmo QuickSort • Outra aplicação • Outras aplicações ...	
Ferramentas (compiladores, interpretadores, etc.) (Máximo : 1,0 pontos) • Ferramentas utilizadas nos exemplos: pelo menos DUAS • Descrição de Ferramentas existentes: máximo 5 • Mostrar as telas dos exemplos junto ao compilador-interpretador • Mostrar as telas dos resultados com o uso das ferramentas • Descrição das ferramentas (autor, versão, homepage, tipo, etc.)	
Organização do trabalho (Máximo: 01 ponto) • Conteúdo, Historia, Seções, gráficos, exemplos, conclusões, bibliografia • Cada elemento com exemplos (código e execução, ferramenta, nome do aluno)	
Uso de Bibliografia (Máximo: 01 ponto) • Livros: pelo menos 3 • Artigos científicos: pelo menos 3 (IEEE Xplore, ACM Library) • Todas as Referências dentro do texto, tipo [ABC 04] • Evite Referências da Internet	
Conceito do Professor (Opcional: 01 ponto)	
Nota Final do trabalho:	

Observação: Requisitos mínimos significa a *metade* dos pontos