1.3.1. Declarando objetos - Javascript

```
episode = {
    title: "Inside",
    duration: 87,
    hasBeenWatched: true
}

console.log(episode.title);

episode["title"] = "What";

console.log(episode["title"]);
```

- A sintaxe de objetos em Javascript é feita por meio de JSON (JavaScript Object Notation) e é muito similar aos dicionários de Python.
- O acesso de objetos em Javascript é parecida com a sintaxe para acessar colunas no Pandas ou dicionários padrões do Python

1.3.2. Comparando estruturas de dados - Tabela

Data Structure	**Python**	**Java**	**JavaScript**
Lists	`list`	`ArrayList`	`Array`
Sets	`set`	`HashSet`	`Set`
Maps/Dicts	`dict`	`HashMap`	`Object` (literal)
Queues	`collections.deque`	`LinkedList`	Not common*
Stacks	`list`	`Stack`	Not common*

1.4.1. Declarando classes - Javascript

```
class Episode {
    constructor(title, duration, hasBeenWatched) {
        this.title = title;
        this.duration = duration;
        this.hasBeenWatched = hasBeenWatched;
    }
}
let firstMovie = new Episode('Inside', 87, true);
console.log(firstMovie)
```

- A sintaxe de criação de classes de JS é muito similar à Java

1.4.2. Declarando classes - Java

`Episode.java`

```
public class Episode{
    private String title;
    private int duration;
    private boolean hasBeenWatched;

    public Episode(String epTitle, int epDuration, boolean
epWatched){
        this.title = epTitle;
        this.duration = epDuration;
        this.hasBeenWatched = epWatched;
    }
}
```

`Main.java`

```
public class Main{
    public static void main(String[] args){
        Episode episode = newEpisode("Inside", 87, true);
```

```
    }
}
```

- A estrutura de criação de classes em java é verbosa, mas simples:

1. Você inicializa as variáveis da classes, junto com sua visibilidade

Visibilidade	Definição
`public`	A variável é acessível de qualquer lugar, dentro ou fora da classe.
`protected`	A variável é acessível dentro da mesma classe, de suas subclasses e dentro do mesmo pacote.
`default`	(Sem modificador) A variável é acessível somente dentro do mesmo pacote.
`private`	A variável é acessível apenas dentro da mesma classe.

- 99% das vezes que você não estiver lidando com herança, você coloca a visibilidade como `private`

2. Você cria um construtor com o mesmo nome da classe. No caso acima, a classe `public class Episode` deve ter o construtor `public Episode`
3. Você coloca os valores que você deseja acessar dentro do contexto da classe

> **Separação de arquivos**
>
> Em Java, apesar de ser possível ter mais de uma classe dentro de um arquivo, isso não é recomendado. De maneira geral, você precisa ter um arquivo por classe (ambos com o mesmo nome), e um arquivo principal main que incorpora as várias classes.

1.4.3. Declarando classes - Python

```
class Episode: # ou class Episode():
    def __init__(self, title:str, duration:int,
hasBeenWatched:bool):
        self.title = title
        self.duration = duration
        self.hasBeenWatched = hasBeenWatched

newEpisode = Episode("Inside", 87, True)
```

- Python possui a sintaxe mais diferentona, mas os princípios são os mesmos.
- O construtor é feito com o método `__init__()`.
- A palavra-chave para referir-se aos métodos e atributos do objeto é `self` por convenção.
- Diferentemente de JS e Java, o `self` precisa ser passado como argumento para permitir que os métodos da classe tenha acessado a esses dados.

1.4.4 Comparando classes - Tabela

Sintaxe	Python	Java	JavaScript
Declaração de Classe	`class NomeDaClasse:`	`public class NomeDaClasse {`	`class NomeD {`
Construtor	`def __init__(self, args):`	`public NomeDaClasse(args) {`	`constructor {`
Métodos	`def nome_do_metodo(self, args):`	`public returnType nomeDoMetodo(args) {`	`nomeDoMetod {`

Sintaxe	Python	Java	JavaScript
Atributos	Dentro do construtor ou métodos	Dentro da classe como membros	Dentro da c como memb
Visibilidade de Membros	Públicos por padrão, privados com `__` e 'protegidos' com `_`	`public`, `protected`, `private`	Não há modificador padrão (pod `_`)
Herança	`class SubClasse(Pai):`	`class SubClasse extends Pai {`	`class SubCl extends Pai`
Polimorfismo	Suportado	Suportado	Suportado

1.5.1. Declarando e manipulando arrays - Javascript

Declaração

```
let guests = ['Sarah Kate', 'Audrey Simon', 'Will
Alexander'];

console.log(guests[0])
```

```
'Sarah Kate'
```

- Arrays em Javascript funcionam exatamente da mesma forma que listas em Python
- Índices começam em 0
- Slicing funciona da mesma maneira que o `Substring` em Java (`início : final(não-incluso)`)

```
let numbers = [0,1,2,3,4,5,6,7,8,9];

console.log(numbers.slice(2,5))
```

```
> [2,3,4]
```

Manipulação

```
let guests = ['Sarah Kate', 'Audrey Simon', 'Will
Alexander'];

guests.push('Tau Perkington'); // adds 'Tau Perkington' to
the end
guests.unshift('Tau Perkington'); // 'Tau Perkington' is
added at the beginning
guests.pop(); // removes the last element from the array

console.log(guests)
```

1.5.2. Declarando e manipulando arrays - Java

```
import java.util.ArrayList;
import java.util.List;

public class TestArray{
    public static void main(String[] args) {
        List myList = new ArrayList<>();
    }
}
```

- Detalhe que é necessário importar o `java.util.ArrayList` e, caso você precise de operações da superclasse List, também `java.util.List`

Manipulação

```
import java.util.ArrayList;
import java.util.List;

public class TestArray {
    public static void main(String[] args) {
        List<String> myList = new ArrayList<>();
```

```
        myList.add("one");
        myList.add("two");
        myList.add("three"); // add at the end

        myList.add(0, "zero"); // add at the beginning

        myList.remove(0); // by index
        myList.remove("three"); // by value

        System.out.println(myList);

    }
}
```

```
[one, two]
```

Um detalhe importante é que, na maioria das vezes, listas contém variáveis de um mesmo tipo em Java. Nas raras exceções que você deseja criar uma lista com tipos misturados, você pode usar `ArrayList<Object> arr = new ArrayList<Object>();`

1.5.3. Declarando e manipulando arrays - Python

```
my_list = ["one","two","three","four"]

my_list.append("five") # coloca no final
my_list.insert(0, "zero") # coloca no início

my_list.pop(5) # remove por índice
my_list.pop(0) # remove por índice

my_list.remove("three") # remove por valor

print(my_list)
```

```
['one', 'two', 'four']
```

1.5.4. Comparando arrays - Tabela

Operação	Python	Java	JavaS
Declaração de Lista	`lista = [item1, item2, ...]`	`List<Type> lista = new ArrayList<>();`	`let li`
Acesso por Índice	`item = lista[indice]`	`item = lista.get(indice);`	`item =`
Slicing	lista[início:final(não incluso)]	lista.substring(int início, int final(não incluso))	lista.sl
Inserção no Final	`lista.append(item)`	`lista.add(item);`	`lista.`
Inserção em Posição	`lista.insert(indice, item)`	`lista.add(indice, item);`	`lista.`
Remoção por Valor	`lista.remove(item)`	`lista.remove(item);`	`lista. 1);`
Remoção por Índice	`item = lista.pop(indice)`	`item = lista.remove(indice);`	`item = [0];`
Tamanho da Lista	`tamanho = len(lista)`	`int tamanho = lista.size();`	`let ta`
Iteração	`for item in lista:`	`for (Type item : lista)`	`for (l`
Ordenação	`lista.sort()`	`Collections.sort(lista);`	`lista.`
Verificar Existência	`existe = item in lista`	`existe = lista.contains(item);`	`existe`

1.6.1. Declarando sets e maps - Javascript

Sets → conjuntos não ordenados que não permitem repetições
Maps → conjunto ordenados de chave-valor, que permite a

procura rápida de um elemento (complexidade O(1) em vez de O(n))

```
const mySet = new Set();

mySet.add(1)
mySet.add(2)
mySet.add(3)

console.log(mySet)

const myMap = new Map();

myMap.set('a', 1);
myMap.set('b', 2);
myMap.set('c', 3);

console.log(myMap)
```

```
> Set(3) { 1, 2, 3 }
> Map(3) { 'a' => 1, 'b' => 2, 'c' => 3 }
```

1.6.2. Declarando sets e maps - Java

```
import java.util.HashMap;
import java.util.Map;

import java.util.HashSet;
import java.util.Set;
import java.util.Arrays;

public class Main {
    public static void main(String[] args) {
        // Sets
        Set<Integer> mySet = new HashSet<>
(Arrays.asList(1, 2, 3, 4));
```

```
        // Maps (Dictionaries)
        Map<String, String> myMap = new HashMap<>();
        myMap.put("key1", "value1");
        myMap.put("key2", "value2");
    }
}
```

1.6.3. Declarando sets e maps - Python

- Sets são criados usando a função `set` sobre outra estrutura de dados, ou utilizando chaves `{}`

```
# Creating a set using set() constructor
my_set1 = set([1, 2, 3, 4])

# Creating a set using curly braces
my_set2 = {4, 5, 6, 7}

my_set1.add(5) # Adds 5 to my_set1
my_set2.remove(7) # Removes 7 from my_set2
```

- Maps em Python são chamados de dicionários. Sua criação é feita com chaves + par de chave-valor `my_dict = {key: value}`. O acesso é feito por meio de colchetes `my_dict["nome_da_chave"]`

```
# Creating a Python dictionary
person = {
    "name": "John",
    "age": 30,
    "city": "New York"
}
```

```python
# Accessing values using keys
print(person["name"])  # Output: John
print(person.get("age"))  # Output: 30

# Modifying values
person["age"] = 31
person["city"] = "San Francisco"

# Adding new key-value pairs
person["gender"] = "Male"
```

- A sintaxe é praticamente idêntica aos objetos de JS

E.1.4.4. Criação de classes

Exercício: Criar uma Classe Livro

Crie uma classe chamada `Livro` com as seguintes propriedades:

1. `titulo` (String)
2. `autor` (String)
3. `ano` (int)

Inclua um construtor para inicializar essas propriedades e um método para
exibir os detalhes do livro.

Em seguida, crie um array ou um ArrayList de objetos `Livro` e realize as seguintes tarefas:

1. Crie e adicione vários livros à lista.
2. Percorra a lista e exiba os detalhes de cada livro.

E.1.4.4.1. Criação de classes - Javascript

```
class Livro{
    constructor(titulo, autor, ano){
        this.titulo = titulo;
        this.autor = autor;
        this.ano = ano;
    }

    mostrarDetalhes() {
        return `${this.titulo} ${this.autor} ${this.ano}`;
    }
}

const livros = [];

// Create and add books to the array
livros.push(new Livro("The Great Gatsby", "F. Scott
Fitzgerald", 1925));
livros.push(new Livro("To Kill a Mockingbird", "Harper
Lee", 1960));
// Add more books...

// Display details of each book
for (const livro of livros) {
    console.log(livro.mostrarDetalhes());
}
```

```
The Great Gatsby F. Scott Fitzgerald 1925
To Kill a Mockingbird Harper Lee 1960
```

Detalhes importantes

- JS é hilário, se você escrever um termo errado como `constuctor`, em vez de `constructor`, ele roda normalmente e dá o resultado como `undefined`

- O sistema de backticks em JS `` `Ano: ${this.ano}` `` é igual à f-string em Python `f"Ano: {self.ano}"`
- O JS não possui um sistema intrínseco de classes para conversão de representação de strings, como o Python tem `__str__` e o Java `toString`

E.1.4.4.2. Criação de classes - Java

```
import java.util.List;
import java.util.ArrayList;

public class Livro{
    private String titulo;
    private String autor;
    private int ano;

    public Livro(String livroTitulo, String livroAutor,
int livroAno){
        this.titulo = livroTitulo;
        this.autor = livroAutor;
        this.ano = livroAno;
    }

    @Override
    public String toString(){
        return this.titulo + " " + this.autor + " " +
this.ano;
    }

    public static void main(String[] args){
        List<Livro> livros = new ArrayList<>();

        livros.add(new Livro("The Great Gatsby", "F. Scott
Fitzgerald", 1925));
        livros.add(new Livro("To Kill a Mockingbird",
"Harper Lee", 1960));
```

```
        for (Livro livro:livros){
            System.out.println(livro);
        }
    }
}
```

```
> The Great Gatsby F. Scott Fitzgerald 1925
> To Kill a Mockingbird Harper Lee 1960
```

E.1.4.4.3. Criação de classes - Python

```
class Livro:
    def __init__(self, titulo, autor, ano):
        self.titulo = titulo
        self.autor = autor
        self.ano = ano

    def __str__(self):
        return f"{self.titulo} {self.autor} {self.ano}"

livros = []
livros.append(Livro("The Great Gatsby", "F. Scott
Fitzgerald", 1925))
livros.append(Livro("To Kill a Mockingbird", "Harper Lee",
1960))

for livro in livros:
    print(livro)
```

```
The Great Gatsby F. Scott Fitzgerald 1925
To Kill a Mockingbird Harper Lee 1960
```