

- [SE](#)
- [SP](#)
- [TO](#)

- [Concursos Federais](#)
- [Concursos Abertos](#)
- [Concursos 2023](#)

[Home](#) » [Blog](#) » Python para concursos públicos – visão geral e principais fundamentos

Buscar no blog



Artigo

Python para concursos públicos – visão geral e principais fundamentos

por [Lara Dourado Vasconcelos Nascimento](#)

Um ano atrás [0 comentários](#)

Compartilhe

-
-
-

Confira neste artigo uma visão geral sobre Python e uma apresentação dos principais conceitos cobrados em concursos públicos.

Olá, pessoal! Tudo bem?

A disciplina **Tecnologia da Informação** vem ganhando cada vez mais espaço, não só nos concursos específicos para profissionais com formação nessa área, mas também em outros que até pouco tempo não costumavam cobrá-la – pelo menos não com tanta ênfase.

Um exemplo desse novo cenário é o [concurso](#) para Auditor Federal de Controle Externo do [TCU](#), que pegou muita gente de surpresa ao cobrar um conteúdo bastante extenso de **Análise de Dados**, na parte de Conhecimentos Específicos do Edital.

Por isso, para estar bem preparado para as provas, é importante que os alunos estejam sempre atentos aos termos e conceitos das áreas de tecnologia, ao menos em um nível de *conhecimentos gerais*. Dessa forma, será mais fácil mergulhar no entendimento e nos detalhes, caso o assunto venha a ser cobrado na sua prova.

No artigo de hoje, falaremos sobre a linguagem de programação **Python**, abordando seu contexto de aplicação e os principais conceitos. Se você não é da área de tecnologia, não se preocupe. A ideia é justamente apresentar a linguagem sem complicações, para todos os públicos! Vamos lá?



Python para concursos públicos

O que é Python?

Inicialmente, é importante entender por que Python é a linguagem de programação mais popular para iniciantes. Em alguns países, como a Austrália, ela é ensinada até mesmo nas escolas, para crianças.

Essa linguagem foi desenvolvida em 1989, como um *hobby* pessoal, por Guido Van Rossum, um matemático holandês, cujo sonho era criar uma espécie de *esperanto* das linguagens de programação. Para quem não sabe, esperanto é uma língua planejada, criada com a pretensão de ser uma língua de comunicação internacional.

Assim, a ideia de seu criador era a de valorizar o conhecimento e a inteligência humana, de modo que uma linguagem intuitiva fosse capaz de abstrair funções mais complexas. Dessa forma, os usuários ficariam livres para focar na sua área de especialidade, que não necessariamente é a computação.

Esse projeto foi muito bem-sucedido, pois Python é bastante conhecida por sua capacidade de servir como uma **interface**. Tipicamente, é muito mais simples para os desenvolvedores aprenderem Python e, através dela, interconectar diferentes sistemas ou bases de dados complexas, concebidas em diversas outras linguagens de programação.

Um dos aspectos que facilitam o aprendizado é que ela é bastante próxima do idioma inglês. Diz-se que é uma **linguagem de programação de alto nível**, devido a essa proximidade com a linguagem humana.

Por ser largamente considerada como uma das linguagens de programação mais fáceis de entender, Python conquistou popularidade entre profissionais de tecnologia que não são programadores de formação, como engenheiros, estatísticos, matemáticos e cientistas de dados.

E será que uma linguagem que é fácil de aprender pode ser também poderosa? A resposta é SIM! Atualmente, Python é utilizada em aplicações de ponta, que requerem **análise e manipulação avançadas de dados**, principalmente aquelas ligadas à ciência de dados, inteligência artificial e *machine learning* – ou aprendizado de máquina.

Conceitos básicos

Tipos de dados – Python para concursos públicos

Ao falar sobre Python ou quaisquer outras linguagens de programação, normalmente os primeiros conceitos envolvidos são os de **representação e manipulação de valores**, ou seja, entender como a linguagem interpreta os dados.

Existem os **valores numéricos**, que representam números:

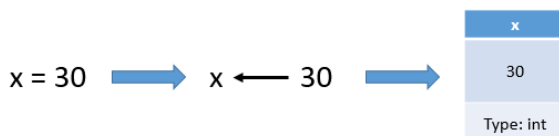
- **Inteiros**: números sem decimais, como 1 / 14 / 2 / 523.
- **Pontos flutuantes**: números com decimais, como 1.2 / 7.65 / 3.75.

E os **valores *string***, que representam textos:

- **Strings**: 'Coruja' / 'edital' / 'Projeto 100 questões por dia'.

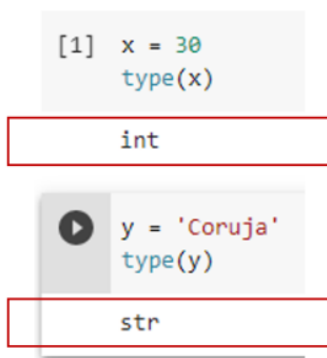
Uma das formas possíveis de manipular esses valores é salvá-los em **variáveis**. Ao fazer isso, o programa coloca o valor salvo em um espaço de memória, associando-o ao nome e ao **tipo** da variável, que representa uma **categoria de valores** à qual o dado pertence.

No exemplo abaixo, ao escrever `x = 30`, o tipo associado é **int**, já que 30 é um inteiro. Se `x` fosse um ponto flutuante, seu tipo seria **float**, e, se fosse uma *string*, seria **str**.



Variáveis em Python

O trecho de código a seguir ilustra uma característica bastante importante da linguagem Python, a **tipagem dinâmica**. Isso significa que a própria linguagem associa um tipo a cada variável criada.



Tipagem dinâmica

No bloco inicial, como `x` é igual a 30, quando o tipo da variável é solicitado através do método `type()`, o resultado é igual a `int`. Já a variável `y` é uma *string*, portanto, o resultado da chamada do método foi igual a `str`.

Em nenhum momento o programador que criou o código precisou indicar que `x` é um inteiro ou que `y` é um texto. Em outras linguagens, seria necessário deixar isso explícito no código. Mais uma vez, nota-se que Python foi construída para ser uma linguagem fácil.

Expressões e manipulações – Python para concursos públicos

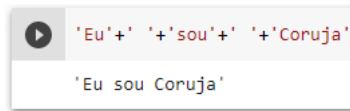
Uma **expressão** é uma combinação de **variáveis** e **operadores** que resulta em um único valor. Os **operadores aritméticos** são os mais conhecidos e alguns deles podem ser vistos na tabela a seguir:

Operador	Expressão, em que <code>a = 4</code> e <code>b = 8</code>
Adição: <code>+</code>	<code>a + b</code> <code>>>> 12</code>
Subtração: <code>-</code>	<code>a - b</code> <code>>>> -4</code>
Multiplicação: <code>*</code>	<code>a * b</code> <code>>>> 32</code>
Divisão: <code>/</code>	<code>b / a</code> <code>>>> 2</code>
Expoente: <code>**</code>	<code>a ** 2</code> <code>>>> 16</code>

Operadores aritméticos

A manipulação de variáveis numéricas via expressões aritméticas é de fácil compreensão, já que é bastante parecida com os exercícios de matemática da escola. E com relação às variáveis textuais, as *strings*? Quais são suas possibilidades de manipulação?

A primeira delas é a **concatenação**, que pode ser feita com o próprio operador `“+”`, como mostra o exemplo abaixo:

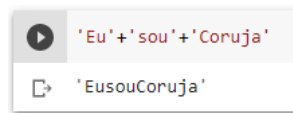


```
'Eu'+' '+'sou'+' '+'Coruja'
```

```
'Eu sou Coruja'
```

Strings em Python exemplo 1

O resultado da “soma” dos trechos ‘Eu’, ‘ ‘, ‘sou’ e ‘Coruja’ resultou na frase ‘Eu sou Coruja’. Importante notar que, se a *string* vazia não tivesse sido utilizada para intercalar as palavras, o resultado seria a frase ‘EusouCoruja’, conforme mostra o trecho abaixo.

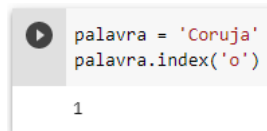


```
'Eu'+'sou'+'Coruja'
```

```
'EusouCoruja'
```

Strings em Python exemplo 2

Há diversas outras possibilidades de manipulação de *string*, a exemplo dos métodos *lower()* e *upper()*, que, respectivamente, tornam todas as letras da *string* minúsculas ou maiúsculas. O método *index* retorna a posição de uma determinada letra na *string*, como no exemplo a seguir:



```
palavra = 'Coruja'
```

```
palavra.index('o')
```

```
1
```

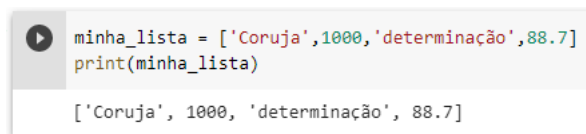
Indexação strings

Esse resultado significa que a letra “o” ocupa a posição 1 na *string* “Coruja”, já que a contagem das estruturas em Python (e diversas outras linguagens de programação) inicia em 0; ou seja, a letra C, que é a primeira letra da palavra, ocupa a posição 0.

Estruturas de dados compostas – Python para concursos públicos

Uma das **estruturas de dados** mais usadas e mais importantes em Python são as **listas**. Elas consistem em uma coleção de elementos que **podem ou não** ser do mesmo tipo. Sendo assim, nada impede que um dos elementos possa ser uma outra lista, o que dá origem a **listas aninhadas**.

Os exemplos a seguir mostram uma lista com elementos de diferentes tipos e outra com listas aninhadas.

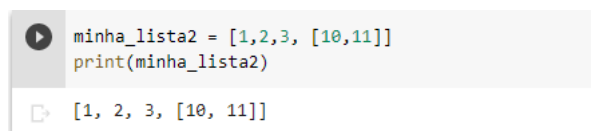


```
minha_lista = ['Coruja',1000,'determinação',88.7]
```

```
print(minha_lista)
```

```
['Coruja', 1000, 'determinação', 88.7]
```

Lista com diferentes tipos de elementos



```
minha_lista2 = [1,2,3, [10,11]]
```

```
print(minha_lista2)
```

```
[1, 2, 3, [10, 11]]
```

Lista aninhada

Nesses exemplos, nota-se que a criação de uma lista é feita com a utilização de colchetes e seus elementos são separados por vírgula.

Para acessar um elemento em uma lista basta usar o operador de indexação [], especificando o índice do elemento desejado dentro dos colchetes – lembrar que o índice do primeiro elemento é 0. O exemplo abaixo mostra como foi possível acessar a palavra “determinação” dentro da primeira lista.



```
minha_lista = ['Coruja',1000,'determinação',88.7]
```

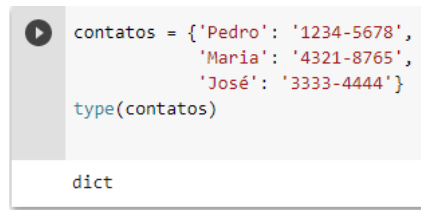
```
print(minha_lista[2])
```

```
determinação
```

Indexação lista

Outra estrutura importante em Python são os **dicionários**, que têm um conceito bastante parecido com os dicionários comuns de idiomas. Cada elemento dessa estrutura possui uma **chave**, que tem o papel análogo ao da palavra que se deseja buscar no dicionário; e a cada chave é associado um **valor**, que seria o significado da palavra buscada.

Outro exemplo intuitivo de uso dos dicionários são as agendas de contatos telefônicos. Cada nome salvo na lista é uma chave, que está associada a um número de telefone, que é o valor. A imagem abaixo mostra a estrutura de um dicionário simples e seu tipo, *dict*.



Dicionário em Python

A grande vantagem dos dicionários é que é possível acessar seus elementos pela chave, ao invés de índices numéricos. Dessa forma, se for necessário buscar o telefone de José, basta simplesmente utilizar `contatos['José']`. Não é necessário saber a posição que esse contato ocupa, como seria caso as mesmas informações estivessem em uma estrutura de lista.

Bibliotecas externas – Python para concursos públicos

Uma das melhores ferramentas que Python oferece são as **bibliotecas**, que aparecem e evoluem constantemente, sendo criadas e disponibilizadas de forma gratuita e colaborativa pela própria comunidade que utiliza a linguagem.

Algumas das bibliotecas mais conhecidas e utilizadas estão relacionadas às áreas que mais utilizam Python, como matemática, estatística, análise de dados e *machine learning*. Vejamos algumas dessas bibliotecas e o foco de cada uma:

- **Matplotlib:** destina-se à criação de **gráficos** e de **visualizações** de dados em geral em Python, sejam elas estáticas, animadas ou interativas;
- **Pandas:** é a mais utilizada em ciência de dados, pois oferece estruturas e ferramentas para o processo de **tratamento e análise de dados**. Suporta diferentes formatos, como CSV e JSON, o que permite trabalhar com várias fontes de dados ao mesmo tempo;
- **NumPy:** considerada como uma extensão matemática da linguagem Python, é utilizada para o processamento otimizado de **matrizes e vetores**. Possui funções e operações para cálculos numéricos, amplamente utilizados em modelos de *machine learning*, processamento de imagens e computação gráfica;
- **Scikit Learn:** desenvolvida especificamente para **aplicação prática de machine learning**, através dela é possível fazer desde o pré-processamento dos dados até a criação de modelos de classificação, regressão e clusterização.

Pessoal, por hoje chegamos ao final. Espero que esse artigo tenha ajudado a despertar o interesse por essa linguagem tão versátil e poderosa, que tem ganhado cada vez mais espaço também no mundo dos concursos. Um abraço e até a próxima.

Cursos e Assinaturas

Prepare-se com o melhor material e com quem mais aprova em Concursos Públicos em todo o país!

Assinatura de Concursos

Assinatura de 1 ano ou 2 anos

[ASSINE AGORA](#)

Sistema de Questões

Assinatura de 1 ano ou 2 anos

[ASSINE AGORA](#)