*Aula 3 - Introdução à programação com o Python I*

Definição do programa. Linguagem interpretada vs. compilada. O Python como uma linguagem

O que é programação? O que é um programa?

A programação como um conceito deriva de outro conceito mais antigo, que é o do algoritmo. Um algoritmo é uma sequência de passos lógicos para se chegar a um resultado. Um código de programação é um algoritmo. Uma receita de culinária também é um algoritmo. Se o algoritmo for seguido taxativamente, o resultado desejado é alcançado, seja um software para um propósito específico, um aplicativo para celular, ou uma pizza.

A programação surgiu como um complemento necessário para o computador. O computador foi originalmente concebido como um dispositivo para acelerar determinados tipos de cálculos. Os cálculos realizados pelo computador foram crescendo e se tornando cada vez mais complexos, juntamente com os desenvolvimentos que fizeram com que os computadores funcionassem cada vez mais rápido, em um espaço mais reduzido e consumindo menos recursos.

Um programa é, portanto, uma forma de o computador entender o funcionamento de um algoritmo. Em sua forma mais básica, um computador só entende as sequências de zeros e uns. Portanto, é necessário encontrar uma forma de transmitir os algoritmos que nós humanos projetamos (em nossa forma de comunicação) para a linguagem que o computador entenda, que por ter os dois símbolos 0 e 1 é chamada de linguagem binária.

Programação e linguagens

Hoje em dia existem muitas linguagens de programação. Mas por que tem que haver tantas? A resposta é simples: "não existe um canivete suíço para a programação". Algumas linguagens têm melhor desempenho para sistemas empresariais; outras têm muito bom desempenho para o desenvolvimento de Web; os sistemas em tempo real têm linguagens que se adaptam melhor às suas necessidades; e, finalmente, há um punhado de linguagens que funcionam muito bem para o Data Science. Entre eles podemos mencionar o Python, R, Julia e o Scala. Embora tenham diferenças mais ou menos sutis, todos eles cumprem com o seu propósito de ajudar o Data Scientist. Além disso, essas quatro linguagens estão entre as mais populares nas buscas de emprego no LinkedIn. Em nossa aula vamos trabalhar com o Python, pois o consideramos uma linguagem que não pode faltar na preparação de um bom CV / Portfólio, e também tem uma curva de aprendizagem não tão acentuada se o que queremos é fazer o Data Science.

Linguagem interpretada vs. compilada.

Para entender como vamos utilizar o Python, vejamos antes algumas de suas qualidades. Antes de mais nada, o Python é uma linguagem interpretada. Isto significa que em todos os momentos temos um programa "intérprete" que traduz quase em tempo real cada um de nossos comandos, linha por linha, para o código binário do computador. Este nem sempre é assim. Outras linguagens (que são chamadas de compiladas) são traduzidas ao binário somente no final, assim que estejam totalmente escritas.

Vamos pensar um segundo sobre esta diferença: as linguagens interpretadas permitem o uso interativo. Isto significa que podemos testar o código "pouco a pouco", sem ter que verificar se ele está completo.

O Python como uma linguagem

Vejamos agora o Python com um pouco mais de detalhe. O Python é uma linguagem open source, o que significa que podemos trabalhar livremente com ela. Uma de suas principais características é sua simplicidade. Que uma linguagem seja simples significa que é fácil de compartilhar com outros programadores e, desta forma, conseguir trabalhar de uma forma colaborativa.

Para ter uma instalação completa do Python com a qual possamos trabalhar, precisamos de pelo menos três componentes:

* O intérprete do Python. Este é um programa que tem que estar funcionando para que possamos executar o código.
* Um editor de texto ou IDE, que é necessário para que possamos efetivamente escrever o código. Analisaremos isso na próxima seção.
* Pacotes para uso específico. Os pacotes são conjuntos de funções que vêm pré-montadas para solucionar problemas comuns.

A seguir veremos em detalhes a instalação do Python para trabalhar no Data Science.

Instalação do Python via Miniconda

De acordo com os componentes mencionados na seção anterior, há várias formas de instalar o Python. Mencionaremos aqui uma das mais simples utilizadas no Data Science: O Miniconda.

O Miniconda é um instalador que oferece o intérprete do Python, juntamente com uma ferramenta chamada Conda, muito útil para instalar pacotes adicionais. Pode ser baixado a partir [deste link](https://docs.conda.io/en/latest/miniconda.html). Tenha em mente que existem duas grandes versões do Python que coexistem: A versão 2 e a versão 3. Neste curso utilizaremos a versão 3.

Uma vez instalado, o Miniconda pode ser utilizado de várias maneiras. O mais simples é iniciá-lo como um programa, e ele abrirá um terminal. Ali digitamos python e entramos diretamente no intérprete, onde podemos trabalhar sem problemas.

Python interativo

Outra maneira de utilizar o Python é através de sua versão interativa, chamada IPython. A diferença com a forma "tradicional" é que este IPython é muito mais orientado para o uso de pequenos blocos de código, que permitem realizar pequenos testes e facilitam a colaboração entre programadores. Não é que uma forma seja melhor do que a outra, cada uma tem as suas vantagens. Além disso, a forma interativa é uma das mais populares para o Data Science. Neste curso, utilizaremos a forma interativa.

Para iniciar o IPython, abrimos o Miniconda e digitamos IPython. Aqui ainda estamos no terminal, mas cada linha de código é marcada com um número. Esta é uma das diferenças da forma interativa.

IPython e notebooks

Até agora ainda estamos trabalhando no terminal, que não é a forma mais confortável de programar. Vejamos isso: acabamos de instalar o intérprete, e até agora temos duas formas de acessá-lo e utilizá-lo. Vamos tentar uma terceira forma, que é a mais utilizada para o Data Science. Continua sendo o IPython, mas veremos que a forma de trabalhar com o código é muito melhor. Esta forma inclui a utilização de gráficos de uma forma muito prática e intuitiva.

Vamos abrir o Miniconda e executar o seguinte código:

| conda install -c conda-forge jupyterlab notebook |
| --- |

Veremos o progresso da instalação no terminal.

A seguir, digitamos no Miniconda

| conda install -c conda-forge jupyterlab notebook |
| --- |

Veremos que se abre uma aba em nosso navegador, mostrando-nos uma lista de arquivos.

Esta forma de trabalho é chamada de programação com "notebook". Novamente, continua sendo o IPython, mas aqui, em vez de trabalharmos através de um terminal, estamos utilizando o navegador. A partir do navegador podemos digitar o código, que passará pelo intérprete (que já temos instalado no computador graças ao Miniconda), e finalmente poderemos ver os resultados no mesmo navegador.

Cada conjunto de linhas de código é digitado em uma caixa de texto. Pode-se trabalhar com várias caixas de texto ao mesmo tempo. Estas caixas de texto podem ser executadas interativamente de uma ou de todas elas juntas. O conjunto completo de caixas de texto constitui um notebook.

Em resumo, o processo de instalação do Python funciona da seguinte forma:

* Instalação do Miniconda com o Python 3
* Instalação dos programas Jupyter e Notebook a partir do Miniconda
* Utilização do IPython no navegador através de notebooks

Esta é uma das configurações mais utilizadas e mais solicitadas nas buscas de emprego.

Mais uma opção com uma variante híbrida

Nos casos em que desejamos tentar algo mais rápido, ou apresentar uma ideia em ambientes mais didáticos do que empresariais, o Google oferece a interface do Colaboratory em <https://colab.research.google.com/>. Você só precisa ter uma conta do Google. Assim que acessamos a página, teremos à nossa disposição um notebook em questão de segundos. Por padrão, esta opção hospeda todas as informações na nuvem. Esta forma de trabalho é muito prática, mas pode ter limitações quando se trata de trabalhar profissionalmente.

Finalmente, com algumas configurações adicionais, pode ser utilizada a interface Colaboratory com o ambiente de execução local. Esta opção híbrida pode ser útil para aqueles que não estão acostumados com o ambiente de Jupyter notebooks.

Noções básicas de programação

Variável, atribuição, expressões

Nesta seção, veremos os blocos conceituais mais básicos da programação com o Python.

Vamos pensar na seguinte expressão matemática: x + y = z, sendo z: f(x,y)

Nesta expressão, queremos representar que a soma dos números x e y resultará no número z.

Agora vamos pensar nisto em termos de programação: se quiséssemos fazer um programa que realizasse uma adição, podemos propor isso da seguinte maneira:

* Inserir um número, que desempenhará a função de x.
* Inserir outro número, que desempenhará a função de y
* Instruir o computador para efetuar a soma dos dois números.
* Faremos com que o resultado desempenhe a função de z
* Mostrar ao usuário o valor de z, ou seja, o resultado da adição dos dois números inseridos x e y.

Todo esse processo é pensado detalhadamente para a programação.

Toda vez que tivermos um símbolo ao qual damos um valor concreto (por exemplo, x = 2), estaremos fazendo uma atribuição a uma variável. O símbolo x é uma variável porque pode assumir qualquer valor. Neste caso em particular, damos a ele (através do processo de atribuição) o valor 2.

No Python, podemos trabalhar da seguinte maneira:

| x = 2 y = 3 z = x + y z 5 |
| --- |

A primeira linha atribui à variável x o valor 2. A segunda linha atribui à variável y o valor 3. A terceira linha soma x e y. A quarta linha mostra o resultado.

Cada parte do código "composto" que representa um valor particular (por exemplo, x + y) é uma expressão. Em princípio, podemos fazer a analogia com expressões matemáticas sem nenhum problema.

Exercícios

Quais são as saídas dos seguintes blocos de código?



Objetos e ponteiros

No Python tudo é um objeto. Um objeto, em termos de programação, é uma entidade que possui dados e, também, possui metadados (atributos ou propriedades) e funcionalidade própria (métodos). Considere o seguinte caso:

| x=1.0 x.is\_integer()  x=1.4 x.is\_integer() |
| --- |

Nesses casos, is\_integer() é uma funcionalidade de x que diz se é inteiro ou não

| print(x.real,x.imag) |
| --- |

Aqui, real e imag são propriedades de x que mostram sua parte real e sua parte imaginária.

Por outro lado, o exemplo da seção anterior mostra a forma de trabalhar com ponteiros (pointers). Ao contrário de outras linguagens onde as variáveis se comportam como contêineres de dados, as variáveis no Python são ponteiros. Isto significa que elas não contêm um valor, mas apontam para um dado que se encontra na memória. Desta forma, se atribuirmos uma variável à outra (y = x), estamos simplesmente dizendo a ambas que apontem para o mesmo dado (a lista [1, 2, 3]). Em sua forma mais específica, a operação “=” realiza a conexão entre o ponteiro e os dados. Em contrapartida, a aplicação de métodos (por exemplo, append()) altera os dados de destino.

Esta diferença é sutil e, na prática, não afeta o trabalho do Data Science. No entanto, traz melhorias substanciais na performance e é importante levá-la em conta para não cometer erros na programação.

Operações

O Python define uma série de operações básicas. Estas operações são mostradas a seguir:

As operações aritméticas são as mais básicas. Prestar atenção a algumas específicas que podem ser muito úteis.

| **Aritméticas** |  |
| --- | --- |
| a + b | Adição |
| a - b | Subtração |
| a \* b | Multiplicação |
| a / b | Divisão |
| a // b | Divisão inteira (resultado sem decimal) |
| a % b | Módulo (restante da divisão inteira) |
| a \*\* b | Exponenciação |
| -a | Negativo |

Além do operador de atribuição “=”, há ainda os seguintes, que são equivalentes às operações da coluna da direita, e podem nos fazer poupar muito tempo.

| **Atribuições** |  |
| --- | --- |
| **a += b** | **a = a + b** |
| **a -= b** | **a = a - b** |
| **a \*= b** | **a = a \* b** |
| **a /= b** | **a = a / b** |
| **a //= b** | **a = a // b** |
| **a %= b** | **a = a % b** |
| **a \*\*= b** | **a = a \*\* b** |

Os comparadores servem para comparar objetos. Eles se comportam de maneira semelhante aos símbolos matemáticos que representam.

| **Comparadores** |  |
| --- | --- |
| **a == b** | **a igual a b** |
| **a != b** | **a diferente de b** |
| **a < b** | **a menor que b** |
| **a > b** | **a maior que b** |
| **a <= b** | **a menor ou igual a b** |
| **a >= b** | **a maior ou igual a b** |

Os operadores de identidade vão mais fundo nos detalhes dos objetos e ponteiros. Se dois objetos são o mesmo (operador “is”), significa que apontam para o mesmo dado. Por outro lado, se quisermos saber se um dado está contido em outro, não importa em que parte, o operador "in" nos será de grande utilidade. Este último é frequentemente usado em listas de dados.

| **Identidade e associação** |  |
| --- | --- |
| **a is b** | **a é o mesmo objeto que b** |
| **a is not b** | **a não é o mesmo objeto que b** |
| **a in b** | **a está contido em b** |
| **a not in b** | **a não está contido em b** |