Relatório- Card N2> - Linguagem de programação python(I)

Jefferson korte junior

Descrição da atividade

Nesta aula aprendemos um pouco sobre a linguagem de programação python, essa linguagem é bem diferente das outras tanto pelo fato sobre as variáveis que não precisa ser declarada antes de serem usadas, como somar variáveis de diferentes tipos.

aprendemos nessa aula como gerar um comentário no código usando o #. Sobre importar pastas e arquivos também aprendemos usando o import(nome_da_pasta).(nome_do_arquivo) e outra maneira é usando o from (nome_da_pasta) inport (nome_do_arquivo).

Falando de **constantes** em python não existe, letras MAIUSCULAS são consideradas constantes, sobre pedir ao usuário para digitar algo na tela usamos o input, também podemos ver o tipo da variável usando o (type(nome_da_variavel), assim podemos ver se ela é do tipo strings, int, float, bool, list, tuple, dict(dicionário) e set (conjuntos).

Podemos mudar o tipo da variável também, EX: raio = input(“informe o raio”) assim ela estará no tipo string, para colocala no tipo float apenas adicione um float antes do input.EX: raio = float(input(“informe o raio”)).

Sobre listas e tuplas, listas aceitam repetições dentro delas e para usar nome_variavel = [elementos_da_lista], assim podendo ate mesmo adicionar ou alterar números ou ate mesmo apagar números de dentro dela. Já as tuplas uma vez que foi definido não pode ser mudado e para se usar nome_variavel = (elementos_da_tupla), e tanto em uma como a outra pode se verificar o numero de elementos que tem dentro, usando o Print(len(nome_variavel)).

Conjuntos a diferença entre lista e tuplas é que ele não é indexizado, assim não podemos acessar o índice de um conjunto e ele tem um característica que também não aceita elementos duplicados. Mais é igual a listas e tuplas podendo ser verificado a quantidade de elementos dentro. Para se usar nome_variavel = {elementos}.

Dicionario é uma estrutura que possui uma sintaxe semelhante aos conjuntos porém ao invés de sua indexação ser numérica, temos a indexação a partir de um valor textual nos permitindo atribuir valor as suas chaves e conseguimos acessar seus elementos e verificar a quantidade de elementos existentes em um dicionario. Sua sintaxe é: nome_variável = {‘chave’: ‘valor’}.

OPERADORES:

Operadores aritméticos: (x+y) (x-y) (x * y) (x / y) (x % y), esse ultimo operador é o modulo que seria o resto da divisão.

Operadores relacionais: em operadores relacionais verificamos a relação entre duas variáveis e retornamos no terminal ela como true ou false, (x > y) (x >= y) (x < y) (x <= y) (x==y) (x !=)

Operadores de atribuição: em atribuição, sempre que atribuímos um novo valor a uma variável o programa irá exibir a nova atribuição podemos também realizar operações

usando o valor anterior, ele não está completamente “excluído”. Os operadores de atribuição disponíveis em python são: +, -, *, / e %.

Operadores Lógicos: esses operadores irão comparar as variáveis com os operadores matemáticos, sendo eles AND, OR, NOT.

Operadores ternários: São os if e else, e em python é a identação que define um bloco, e caso quiser colocar mais condições se usa o elif.

Sobre as estruturas de controle: caso o valor de uma variável atenda as condições iniciais da estrutura de controle, ela vai entrar no bloco do if, caso contrario ela vai para o bloco da estrutura do else e caso não atenda as condições de nenhum dos blocos, o programa irá encerrar ou irá executar algo que esteja fora do bloco. O bom nessa estrutura podermos usar expressões relacionais para definir se é verdadeiro ou falso.

Laços de repetição: Aprendemos sobre o **while** que é uma estrutura de repetição que ira repetir ate que seja falso a condição dada, e sobre o **for** ele ira repetir sua execução ate que um determinado valor ou comando seja atingido, dai assim encerrara sua execução.

Funçoes: nessa segunda parte da aula começamos a aprender sobre funções algo importante a falar sobre funções é que ela pode ter um valor padrão para determinado parâmetro, podendo chamá-la de duas formas diferentes que pode ser passando ou não um parâmetro. Em python não podemos atribuir valores diferentes para uma mesma função em vez disso ele apenas sobrescreve a função, o que temos disponível são parâmetros opcionais que faz com que conseguimos chamar a mesma função de diferentes formas. Podemos executar uma função retornando um valor mas ela só será impressa na tela se ela for armazenada em uma variável. Outro ponto importante é que, ao definir uma função, é mais recomendável fazer com q ela retorne um valor, ao invés de imprimir diretamente dela pois isso possibilita usar a função de diversas maneiras.

Funçoes lambda: São funções anônimas apenas usando o nome lambda já as chama.

pep – proposta de melhoria do python é um guia de estilo de codificação do python que fornece informações sobre melhorias. Conseguimos ter um bom exemplo de pep nas funções do tipo args, onde definimos uma tupla ou um dicionario com vários argumentos contidos que são “empacotados” na função definida. Podemos também passar uma função como parâmetro para outra o que nos ajuda a reduzir o tempo de processamento de um código. Algo que também ajuda muito a reduzir tempo de processamento é a função map reduce que percorre todos os elementos de uma lista e chama uma função em que o primeiro parâmetro dela é um acumulador e o segundo é o elemento atual.

O ****kwargs** permite que você pode ter uma quantidade indefinida de argumentos nomeados para uma função. Isso oferece um a grande flexibilidade, pois você pode adicionar quantos argumentos precisar sem alterar a definição da função. Dessa forma, é possível adaptar a função a diferentes necessidades e contextos, tornando o código mais fácil.

***args** também permite que a função receba um numero indefinido de argumentos so que posicionais. E esses argumentos são armazenados em uma tupla , o que facilita a manipulação dentro da função.

Diferença entre argumentos posicionais e argumentos nomeados como citei acima, a diferença maior é como eles são passados oara uma função e como ela os interpreta, porem os argumentos nomeados podem ser passados em qual quer ordem assim ate podendo evitar erros na hora de colocar em ordem.

MAP: Ele aplica uma função a todos os itens itinerários como uma lista por exemplo e a transforma em um novo iterador com outros resultados.

FILTER: dependendo o caso e oque quer fazer ele pode ate mesmo substituir o laço de repetição, so que apenas sobre uma condição especifica, assim varrendo o itinerário e so selecionando os com critério dado.

REDUCE: ele já é aplicado quando necessário uma operação cumultativa, assim fazendo a agregação e reduzindo ao um único valor.

ENUMERATE: serve quando queremos enumerar alguma tupla ou alguma lista, sendo muito útil quando precisamos de um índice sobre cada itinerário.

Agora vamos falar um pouco sobre as **CLASS**, São maneiras de agrupar dados e funcionalidades dentro de uma única estrutura. Assim podendo se criar instancias (objetos) e definir métodos (funções) dentro de uma classe. Para definir uma classe, apenas usando a palavra-chave class seguida pelo nome da classe e dois pontos**.**e dentro da classe podemos definir um método especial chamado __init__ para inicializar os atributos. Em classes também temos **heranças** que permite criar uma nova classe a partir de uma classe existente, herdando seus atributos e métodos.

O __**init**__ é frequentemente usado em conjunto com classes, embora seu uso não seja obrigatório, ele é extremamente útil para inicializar os atributos de um objeto quando ele é criado. Ele é chamado automaticamente quando uma nova instancia de uma classe é criada, permitindo que você configure o estado inicial do objeto.

@property ele é usado para transformar métodos de uma classe em propriedades, permitindo que você acesse esses métodos como se fossem atributos, e isso facilita a criação de getters e setters também.

Setter é um método usado para definir ou modificar o valor de um atributo de uma classe.etter são frequentemente usados em conjuntos com o decorador @propety para controlar o acesso e a modificação de atributos.

@classmethod é usado par definir métodos que estão ligados a classe e não as instancias da classe. Isso significa que, invés de receber uma instancia da classe como primeiro argumento (geralmente chamado de self), o método recebe a própria classe como primeiro argumento, que é convencionalmente chamado de cls.

@staticmethod: ele já define métodos que pertencem a classe, mas que não dependem de nenhuma instancia da classe. Isso significa que você pode chamar esses

métodos diretamente pela classe, sem precisar criar um objeto da classe. E ele também na pode acessar e nem modificar nenhum elemento da classe ou instancia.

Conclusões

nesta aula foram introduzidos alguns conceitos básicos de python que serão essenciais para aplicar em aprendizado de maquina e também podemos observar que a sintaxe simplificada e de fácil entendimento faz com que essa linguagem de programação possar ser uma das principais para o uso em aprendizado de maquinas.

Referencias

https://www.youtube.com/watch?v=oUrBHiT-lzo

https://www.youtube.com/watch?v=iq7JLIH-sV0&t=2705s