Revisão para a prova!!11!!

Instalação e utilização do Github

- O GitHub é uma plataforma de desenvolvimento colaborativo que permite armazenar, compartilhar e colaborar em projetos de software

- Alguns passos são fundamentais para instalar o Github em nossas máquinas. Isso pode ser feito através de uma forma simples, e o download pode ser feito pelo próprio site do Github.

- É importante nos atentarmos a alguns detalhes, por exemplo, devemos nos certificar que o computador usado cumpre os requisitos mínimos para que o programa possa ser usado sem maiores problemas.

- Usamos o Github por meio de um terminal de comandos, dentro dele, há códigos como “git add”, para iniciar o envio de determinado projeto, “git commit” para salvar mudanças.

Tuplas, dicionários, listas e matrizes

- Tuplas são uma sequência de elementos separados por vírgulas, representados ou não entre parênteses, pode-se misturar elementos diferentes. Ao contrário das listas, as tuplas são **IMUTÁVEIS.**

- Dicionários, um tipo de tupla, são estruturas de dados que associam uma chave com um valor. Os valores podem ser um dado de qualquer tipo, mas as chaves só podem ser dados de tipos imutáveis.

- Listas: Uma lista em Python é uma estrutura que armazena vários dados, que podem ser de um mesmo tipo ou não. Podendo inclusive conter listas dentro de listas. São mutáveis. Uma lista é criada como a construção: [dado1,dado2,...]. Não permite o acesso por chaves, como os dicionários.

- Um elemento de uma lista em uma posição específica tem a mesma funcionalidade de uma variável normal. Por fim, os índices negativos referem-se à lista da direita para a esquerda.

Tuplas, dicionários, listas e matrizes

- Matrizes: São similares a listas, porém possuem mais de uma dimensão, e, portanto, armazenam uma quantidade maior de dados de forma mais organizada. Seguem a sintaxe:

  matriz = [[9.5, 9, 8, 0], [5, 6, 7, 8.5], [9, 8, 8, 0],

  [3.6, 7.0, 9.1, 8.7], [5.0, 4.5, 7.0, 5.2],

  [2.1, 6.5, 8.0, 7.0], ...]

- Os dados são acessados através de dois indicadores, que referem-se a linhas e colunas. Da seguinte forma: matriz[0][2]. Nesse exemplo, estaremos acessando o dado que se encontra na primeira coluna e segunda linha.

Strings

- As strings em Python possuem diversas funções. Strings são, a nível básico, conjuntos de caracteres, ou seja, são dados em formato de texto.

- Possuem vários métodos e funções específicas. A função *slice()*, por exemplo, serve para fatiar uma string em uma porção menor. Pode-se fatiar (slice) strings usando a seguinte sintaxe: [inicio:fim-1:passo].

- O operador in verifica se uma substring é parte de uma outra string. Por exemplo, o comando *resultado = "10" in "tirei 10 na prova"* vai verificar se os caracteres "10" estão presentes na frase "tirei 10 na prova"

- O método find retorna onde a substring começa na string, ou seja, além de verificar se algo é parte de uma string, como o exemplo acima, também retorna a posição em que determinada substring está. Dessa forma:

Strings

```
a = “tirei 10 na prova!”
print(a.find(“10”))
```

- Esse comando irá retornar o valor 6, pois a string “10” encontra-se na sexta posição da frase “tirei 10 na prova!”.

- O método *find()* retorna -1 quando a substring não aparece na frase analisada.

- O método *lower()* converte todos os componentes de uma string para caracteres minúsculos.

- E tem vários outros métodos aí
- Muita coisa mesmo
- Vários

Tipos de variáveis

- É um conceito bem simples, existem variáveis indicadoras e contadoras. As duas geralmente são usadas dentro de laços de repetição.

- Variáveis contadoras funcionam como um valor numérico, que vai aumentando até que certa condição seja satisfeita. Por exemplo:

```
while (numero <= 3):
.
.

numero = numero + 1
```

- Nesse exemplo, o laço de repetição *while* continua sendo executado enquanto a variável “numero” for menor que 3. Dentro do laço, soma-se a variável numero um por um, de forma gradual, garantindo que o laço se repita o número de vezes que foi estabelecido.

Tipos de variáveis

- Variáveis indicadoras possuem um valor booleano, ou seja, terão sempre um de dois valores possíveis: “*True*” ou “*False*”, ou seja, “Verdadeiro” ou “Falso”.

```
while (numero <= n-1):
if (condições):
VariavelIndicadora = True
```

- Nesse exemplo, há uma variável indicadora dentro de um laço de repetição. Dentro do laço, há um teste. Se as condições dele forem satisfeitas, a variável indicadora irá assumir um valor “True”, ou seja, “Verdadeiro”.

Bibliotecas

- Bibliotecas são ferramentas poderosas dentro do Python. De forma similar a bibliotecas físicas, em que emprestamos livros, nas bibliotecas do Python, podemos “emprestar” códigos e funções específicas. Facilitam muito nossa vida :)

- Ao utilizarmos bibliotecas, o primeiro passo sempre é importá-las em nosso código. Isso é feito com a seguinte sintaxe:

  import NomedaBiblioteca <- nesse exmplo, usei um nome genérico, claro.

Bibliotecas

- Uma das bibliotecas mais usadas é a “time”, nativa do Python e que tem diversas funções para entregar informações acerca de horários e datas atuais.

- a Função “*gmtime()*” dentro dessa biblioteca retorna informações sobre data e hora de forma detalhada, armazenando cada informação em uma variável separada. Por exemplo, “*tm_month*” armazena o mês, “*tm_date”* armazena a data e etc.

- A função “*sleep()”* faz com que o Python aguarde uma quantidade de tempo em segundos para executar o próximo código.

- Entre outras funções, também existem inúmeras outras bibliotecas.
- Muitas
  Diversas
- Muitas mesmo

Funções

- Funções referem-se a uma forma de programar, mais do que uma estrutura específica.
- Servem para separar o programa em partes menores, facilitando a compreensão.
- Seguem a seguinte sintaxe:

```
def NomeDaFunção(parâmetro1, parâmetro2…):
    comandos
    da
    função

    return ValorDaFunção
```

- O comando "*def*" vem de "definir", é o comando inicial de qualquer função.
- Os parâmetros são variáveis que serão processadas dentro da função
- O comando *Return* serve para que a função entregue alguma informação após processar os parâmetros. Quando usamos o comando *return*, a função é encerrada automaticamente.

Funções

- Para que as funções sejam utilizadas, precisamos chamá-las dentro do programa. Da seguinte forma:

```
def NomeDaFunção(parâmetro1, parâmetro2…):
    comandos
    da
    função

    return ValorDaFunção
a = NomeDaFunção(1, 2)
```

- Nesse exemplo, Chamamos a função usando o nome dela e atribuindo o valor de retorno a uma variável “a”, e inserindo os valores para os parâmetros entre os parênteses, nesse caso, os valores são 1 e 2.

Funções

- Algumas observações são importantes. Por exemplo, no geral, é impossível chamar uma função e depois defini-la. O python entende isso como se a função ainda não existisse, tornando sua utilização impossível. Segue um exemplo:

```
a = NomeDaFunção(1, 2)

        def NomeDaFunção(parâmetro1, parâmetro2…):
        comandos
        da
        função


        return ValorDaFunção
```

- Aqui, invertemos a ordem do exemplo anterior. Isso causaria um erro se tentássemos rodar, já que a função é definida somente após ser chamada.

Funções

- Por fim, dentro das funções existem dois tipos de variáveis, chamadas de globais e locais.

- Uma variável é chamada global se ela for criada fora de qualquer função. Também quer dizer que ela pode ser alterada e editada de qualquer lugar do programa.

- Uma variável é chamada local se ela é criada ou alterada dentro de uma função. Logo, só pode ser alterada e editada dentro da função em que foi criada, por isso é chamada de local :)

E é mais ou menos isso, podem se basear por esses slides de revisão, mas seria indicado que dessem uma olhada no conteúdo completo pelo AVA também.

Boa sorte meus bacanos

meu deus era um PNG falso que tristeza