**Python Básico**

**(lógica de programação)**

**1 – comentários de código em python**

Para criar um comentário em python é só colocar a # no início da linha, mas esse é um comentário de uma linha só. Para um comentário de múltiplas linhas basta você colocar três aspas (simples ou duplas) no início da parte que deseja comentar e no final da parte que deseja comentar. (na verdade não é um comentário, sim uma docstring (pesquisar isso depois)).

E se abrir o comentário com aspas simples, deve-se usar o mesmo para fechar. E isso vale também para as aspas duplas.

Texto

Descrição gerada automaticamente

Para duplicar uma linha é só apertar ctrl + d.

Python é uma linguagem interpretada mas, assim como Java, passa por um processo de compilação. Um código fonte Java é primeiramente compilado para um bytecode e depois interpretado por uma máquina virtual.

**2 – comando Print**

A função print serve para mostrar alguma coisa em tela. E funções podem receber argumentos. No caso abaixo, os números são um argumento para a função print.

Interface gráfica do usuário, Texto

Descrição gerada automaticamente

Assim como em Js os argumentos são separados por uma vírgula.

Em python a gente tem argumentos nomeados, e para a função print por exemplo nós temos o argumento **sep=’’**. O valor que ele receber entre as aspas vai ser o que ele vai usar para separar os valores que foram passados para print imprimir.

Interface gráfica do usuário

Descrição gerada automaticamenteTexto

Descrição gerada automaticamente

Como foi visto acima, para a impressão dos prints a linha foi quebrada, e esse é o padrão. Para que tenha os prints exibidos na mesma linha basta passar o argumento end. Com o end a gente pode mandar qualquer coisa que a gente queira que esteja no final da linha, e por padrão ele manda uma quebra de linha. Mas se a gente passar um espaço vazio para o end ele não vai ter a quebra de linha separando os prints na hora da impressão.

Interface gráfica do usuário, Texto

Descrição gerada automaticamente



(alguma palavra seguida de parênteses ou é uma função, ou é uma classe).

**3 – strings e aspas em python**

String é um dado primitivo em python. E em resumo strings são textos que estão dentro de aspas.

O python é uma língua de tipagem dinâmica. Isso significa que se a gente colocar um texto entre aspas ele automaticamente reconhece que se trata de uma string. Não é necessário dizer se o valor em questão é um float, int, boolean, string , e etc... Esse comportamento é igual ao do Javascript em relação à tipagem.

Assim como no Js, se a gente colocar o número 5 dessa forma ele será reconhecido como int, um number. Mas se a gente colocar ele assim ‘5’ ele será reconhecido como uma string, e isso vai afetar na hora de fazer cálculos ou concatenações.

Também à semelhança do Javascript, caso a gente abra algo com ‘ ‘ , dentro disso nós temos que usar “ ” caso desejarmos usar aspas novamente. E vice versa.

Caractere de escape = \. Não é uma boa prática usar isso, usar se for extremamente necessário.

\n cria uma quebra de linha no meio da string. Caso a gente queira imprimir o \n é só colocar um **r** na frente das aspas que ele vai entender que os comandos ali dentro não devem ser executados e vai ignorá-los.

\*\*Tudo em python é um objeto, até os tipos primitivos.

**4 - Tipos de Dados “primitivos”**

Tipos de dado:

**Str – string |Int – inteiro| Float – número real/ponto flutuante | Bool -booleano/ lógico**

Tela de computador com texto preto sobre fundo branco

Descrição gerada automaticamente

Se a gente mandar algum valor vazio, como array, objeto, e etc o valor booleano desta coisa vazia é false. O número 0 também é avaliado como falso.

**Type casting**

(conversões de um tipo para outro)

Para fazer as conversões a gente usas as mesmas siglas dos nomes dos tipos (str, int, float, bool)

Se a gente tiver qualquer valor dentro de alguma coisa e a gente converter para booleano, a coisa em questão vai ser convertida para true.

Interface gráfica do usuário, Texto

Descrição gerada automaticamente

Ao final dessa linha 14, a gente converteu o 10 str para int.

Devemos ficar atentos também ao que a gente está mandando converter. Por exemplo, se a gente manda converter a string ‘Davi’ para int ou float vai dar errado. Pois letras não são números e não logo não podem ser convertidas para esses tipos de dado. (lembrando que a conversão de float ou int para string acontece, o que não acontece é esse caminho inverso). Também não temos como converter um float para int. Mas um int em float é possível.

Obs é possível concatenar duas str, ex ‘10’ + ‘10’ resulta em ‘1010’. Mas concatenar uma str com float ou int não funciona também.

**Aula 5 – Operadores aritméticos**

Texto, Site

Descrição gerada automaticamente

O operador de \* também serve como operador de repetição. Tendo isso em mente imagine o seguinte exemplo: Vamos supor que você faça ‘10’ \* ‘10’ o python vai dar um erro pois não se multiplica str com str. Mas se a gente fizer ‘10’ \* 10, o python vai entender que é pra repetir a str ‘10’ 10 vezes, então vai exibir ela 10x na tela.

 

Outro comportamento é a concatenação. Se a gente “somar” duas str, elas vão ser concatenadas. Print(‘10’ + ‘10’) = 1010. Porém a gente não pode juntar int com str.

Porém diferente do Js que a gente consegue concatenar int com str aqui, para realizar essa concatenação, a gente tem que transformar esse valor em str. Ex print(‘meu nome é ’ + ‘’ + ‘Davi’ + ‘’ + ‘ e tenho’ + str(21) + ‘’ + ‘anos’). Nesse caso a gente poderia estar mudando a tipagem de alguma variável (o que faz mais sentido). Obs esse exemplo de string concatenada não é o normal do dia a dia da programação em python.