**PYTHON BÁSICO**

**TIPOS DE DADOS (PADRÃO DA LINGUAGEM):**

Texto – str;

Numéricos - int, float, complex;

Sequência – list, tuple, range;

Mapa – dict (chave: valor);

Coleção – set, fronzenset;

Booleano – bool;

Binário – bytes, bytearray, memoryview;

**Interpretador Python:** programar e ir vendo o resultado no terminal

Digitar “python” ou “python -i nome\_do\_programa.py” no terminal. Para sair digitar “exit()”

**Dir:** Sem argumento, retorna a lista de nomes no escopo local atual. Com um argumento, retorna uma lista de atributos válidos para o objeto. Exemplo

- dir()

- dir(100): lista de métodos dentro da classe int

**Help:** Invoca o sistema de ajuda integrado. É possível fazer buscas em modo interativo ou informar por parâmetro qual o nome do módulo, função, classe, método ou variável. Exemplo:

- help(): você consegue buscar o método e ele retorna todos os métodos e utilidades daquilo que foi buscado;

-help(100): aqui vai trazer as informações sobre os métodos da classe int

**VARIÁVEIS E CONSTANTES**

**Variáveis:** seu valor muda ao longo do programa. Definição: variável = valor

**Constantes:** seu valor não muda, uma vez atribuído um valor, ele sempre será. Exemplo: um programa sobre dados pessoais, o tipo sanguíneo será um valor constante. Definição: utiliza-se o nome da variável, por convenção, deve ser escrito todo em maiúsculo, VARIAVEL = valor.

**Boas práticas:**

- Padrão de nomes deve ser o snake case: substituir espaços pelo *underline*;

- Escolher nomes sugestivos;

- Nome de constantes todo em maiúsculo.

**CONVERSÃO DE TIPOS**

Guido Van Rossum – 1989: Linguagem intuitiva, código aberto

* Tipagem dinâmica e forte;
* Multiplataforma e multiparadigma(paradigma procedural, funcional, orientado a objetos...);
* Comunidade gigante e ativa;
* Curva de aprendizagem baixa

**FUNÇÕES DE ENTRADA E SAÍDA(input e output)**

**OPERADORES ARITMÉTICOS**

* +(adição)
* -(subtração)
* \*(multiplicação)
* /(divisão)
* %(módulo)
* \*\*(exponenciação)

-Precedência de operações segue as regras da matemática, logo:

* Parêntesis
* Expoentes
* Multiplicação e divisão(da esquerda para a direita)
* Soma e subtração(da esquerda para a direita)

**OPERADORES DE COMPARAÇÃO:** são operadores utilizados para compara valores.

* ==(igual)
* !=(diferente)
* >(maior)
* >=(maior ou igual)
* <(menor)
* <=(menor ou igual)

**OPERADOR DE ATRIBUIÇÃO:** iniciar o valor de variável ou sobrescrever

* valor = 0 (valor recebe 0)
* valor += 200 (valor recebe valor + 200)

**OPERADORES LÓGICOS:** usados com o de comparação cria-se sentenças

print("True and True: ", True and True)#True

print("False and False: ", False and False)#False

print("True and False: ", True and False)#False

print("True or False:", True or False)#True

print("True or True:", True or True)#True

print("False or False:", False or False)#False

**OPERADORES DE IDENTIDADE:** saber se as variáveis estão no mesmo espaço de memória

* Is (é)
* Is not (negação)

**OPERADORES DE ASSOCIAÇÃO:** são operadores utilizados para verificar se um objeto está presente em uma sequência.

//Esses operadores são case sensitive, ou seja se está escrito com maiúscula ou minúscula faz diferença

* in (está)
* not in (não está)

**ESTRUTURAS CONDICIONAIS E DE REPETIÇÃO:**

if / if else / elif: a estrutura condicional permite o desvio de fluxo de controle, quando determinadas expressões lógicas são atendidas.

* if testa a condição lógica e da um retorno.
* if-else torna possível a execução de dois desvios, ou seja, saber o que fazer em duas situações diferentes(contrárias).
* if-elif-else torna possível o teste de vários desvios, ou seja, testa várias condições diferentes.

**Comando for:** usado para percorrer um objeto iterável. Faz sentido usar ‘for’ quando sabemos o número exato de vezes que o bloco de código deve ser executado, ou quando queremos percorrer um objeto iterável.

**Função Range:** é uma função built-in(própria) do Pyhton, ela é usada para produzir uma sequência de números inteiros a partir de um início(inclusivo, ou seja, está incluso na contagem) para um fim(exclusivo, ou seja, não aparece na contagem. Se o start e stop for(0,10) ele vai contar de 0 a 9).

Se usarmos(i,j) será produzido:

I, i+1, i+2, i+3, ..., j-1.

Ela recebe 3 argumentos: stop(obrigatório), start(opcional) e step(opcional).

**Comando while:** usado para repetir um bloco de código várias vezes. Faz sentido usar o while quando não sabemos o número exato de vezes que o bloco de código deve ser executado. O fim da execução é determinado por uma condição.

**//break:** palavra reservada que interrompe o loop a qualquer momento, mesmo que ele não tenha chegado ao fim ou a condição de parada não tenha sido satisfeita. Não é muito recomendada...

**Comando continue:** comando utilizado para pular um ciclo de iteração dos comando for e while.