Introdução à Python

Origem:

- Criada por Guido van Rossum em 1989
- Projetada com a filosofia de priorizar a **importância do esforço do programador sobre o esforço computacional** e a **legibilidade do código sobre a velocidade**.

Características:

- Conceitos fundamentais simples e poderosos;
- **Sintaxe clara e legível** (programas mais curtos);
- Tipos de variáveis predefinidos fortes e poderosos;
- Portável e independente de plataforma;
- Licença Open Source;
- **Orientada a objetos** mas com suporte a outros estilos ou paradigmas de programação: funcional, etc;
- **Interpretada:** compilação implícita e automática;
- **Extensível:** permite usar módulos escritos em C, C++ e Java;
- Vasto repertório de **bibliotecas**;
- Estruturas de dados poderosas, nativas, e que normalmente não estão disponíveis em outras linguagens de programação mais tradicionais: listas, dicionários;
- Verificação de tipos é dinâmica: **não há declaração de variáveis**, mesma variável pode ser usada ara objetos de tipos diferentes em momentos diferentes da execução.

Programa básico em Python

- Em Python, não há uma declaração explícita para a criação das variáveis:

    - A sua primeira utilização é que será a sua declaração *(implícita)* - tipicamente um comando de atribuição;
    - O tipo de dados usado na criação será o tipo do valor usado neste primeiro comando.
- Alguns tipos básicos de Python: **int, long, float, string, list, tuple, dictionary, file.**
- Estrutura de Python que não existe em outras linguagens. Sua sintaxe genérica é:

```
var1, var2, ... = expr1, expr2, ...
# Exemplos:
v1, v2 = v2, v1 # Troca os valores de v1 e v2.
v1, v2, v3 = v2, v3, v1 # Semelhante com 3 variáveis.
```

    - Execução simultânea das atribuições das expressões às variáveis correspondentes.

Comando input

- Principal comando de leitura;
- Tudo é lido sempre como string;

```
# Exemplo:
var = input(“Mensagem qualquer: ”)
var = int(input(“Mensagem qualquer: ”))
```

Comando print

- Escreve na saída o conteúdo que segue.
- Conteúdos múltiplos são separados por vírgulas e são normalmente impressos com um espaço de separação.
- Cada print começa uma linha exceto se o último print tiver uma

```
a, b = 10, 20
#Exemplos de comando print:
```

vírgula no final.

- Constantes String precisam ser delimitadas por aspas ou apóstrofos. Dentro delas pode-se usar também:
  - – \n para mudar de linha na impressão.
  - – \t para dar um <tab> na impressão.
  - – \", \' e \\ para escrever o símbolo após a \

```
print(a) #10
print("Resultado é", a) #Resu
print("A =", a, end=' ') #A =
print("e B =", b) #e B = 20
print("A =", a, "\t B =", b) #
    #B = 20
print("A =", a, "\n B =", b) #
#B = 20
```

```
# Este programa repete a idade digitada.
idade = 0 # Declara a variável idade...
idade = input("Digite a sua idade: ")
print("A idade digitada foi", idade)
```

- A função **type** pode ser usada em variáveis e retorna o tipo atual do valor armazenado.

```
# Exemplo:
type(nome)
```

Algumas funções nativas de Python

```
int (x) #transforma x em inteiro (se possível).
float (x) #transforma x em tipo float.
str (x) #transforma x em string.
pow (x,y) #mesmo que x**y.
round (x, n) #arredonda x para n casas decimais
#se n for 0, resultado será valor inteiro mas tipo será float.
```

Biblioteca Math

```
import math # Dá acesso às funções matemáticas.

math.sqrt (x) # significa raiz quadrada de x.
math.pi # Retorna a constante Pi.
math.trunc (x) # float -> int desprezando a parte decimal.
math.floor (x) # Mesmo resultado mas com tipo float.
math.ceil (x) # Semelhante mas valor vai para cima.
math.sin (x) # Retorna o seno de x
```

Operadores Aritméticos com Atribuição

```
var1 += var2 #significa var1 = var1 + var2
var1 -= var2 #significa var1 = var1 - var2
var1 *= var2 #significa var1 = var1 * var2
```