Lição 1

Entrada, impressão e números

1. Como ler e escrever em Python

Todo programa é, eventualmente, um processador de dados, então devemos saber como inserir e produzir dados dentro dele. Existe uma função, **print()**, para mostrar dados de qualquer programa Python. Para usá-la, passe uma lista separada por vírgulas de argumentos que você deseja imprimir para a função **print()**. Vamos ver um exemplo. Copie e cole o programa a seguir no IDLE do Python e o execute para ver como o programa está sendo executado linha por linha:

```
print(5+10)
print(3*7, (17-2)*8)
print(2**16) # dois asteriscos são usados para exponenciação (2 elevado a 16)
print(37/3)  # barra simples é uma divisão
print(37//3) # barra dupla é uma divisão inteira
             # retorna apenas o quociente da divisão (ou seja, nenhum resto)
print(37%3)  # sinal de porcentagem é um operador de módulo
             # resulta no resto do valor esquerdo dividido pelo valor direito
```

No programa anterior, utiliza-se comentários. Um comentário é uma informação adicional acrescentada ao programa para melhorar seu entendimento ou leitura. Comentários iniciam com o caractere "**#**" e vão até o final da linha – comentários não interferem nem participam da execução do programa.

O programa anterior não requer nenhum dado adicional para a sua execução. Para inserir dados em um programa, usamos a função **input()**. Esta função lê uma única linha de texto, como uma **string** (cadeia de caracteres).

O programa a seguir lê o nome do usuário e o cumprimenta. Teste-o:

```
print("Qual é o seu nome? ")
nome = input() # lê uma única linha e armazena-a na variável "nome"
print("Olá, "+nome+"!")
```

2. Soma de números e strings

Vamos tentar escrever um programa que introduza dois números e imprima sua soma. Lemos os dois números e os armazenamos nas variáveis **a** e **b** usando o operador de atribuição "**=**". No lado esquerdo de um operador de atribuição, colocamos o nome de uma variável.

Variável é um pedaço de memória no qual podemos armazenar um valor. Para referenciar uma variável fazemos uso de um **nome**, um **identificador**. O nome de uma varável é uma cadeia de caracteres latinos (**A-Z**, **a-z**, **0-9**, **_**), mas deve começar com uma letra no intervalo "A-Z", "a-z", ou com "_" (caractere sublinha), nunca com um dígito.

No lado direito de um operador de atribuição, colocamos qualquer expressão que o Python possa avaliar. O nome passa a apontar para o resultado da avaliação. Copie e cole este exemplo, execute-o, informando os valores 5 e 7, e veja a saída:

```
a = input()
b = input()
s = a + b
print(s)
```

Depois de executar o exemplo, podemos ver que ele imprime 57. Como aprendemos na escola, 5 + 7 dá 12. Então, o programa está errado e é importante entender o porquê. O fato é que, na terceira linha, s = a + b, Python "somou" duas strings, em vez de dois números. A soma de duas strings em Python funciona da seguinte maneira: elas são apenas coladas uma após a outra. À essa operação chamamos de "concatenação de strings". Strings e números são representados em Python de maneira diferente.

Todos os valores em Python são chamados de "objetos". Cada objeto tem um certo tipo. O número 2 corresponde a um objeto "número 2" do tipo "**int**" (ou seja, um número inteiro). A string "Olá" corresponde a um objeto "string 'Olá'" do tipo "**str**". Cada número real de ponto flutuante é representado como um objeto do tipo "**float**". O tipo de um objeto especifica que tipos de operações podem ser aplicadas a ele. Por exemplo, se as duas variáveis "primeiro" e "segundo" estão apontando para os objetos do tipo **int**, o Python pode multiplicá-los. No entanto, se eles estiverem apontando para os objetos do tipo **str**, o Python não pode fazer isso:

```
primeiro = 5
segundo = 7
imprimir (primeiro*segundo)
# você pode usar aspas simples ou duplas para definir uma string
primeiro = "5"
segundo = "7"
print(primeiro*segundo)
```

Para converter (converter) a sequência de dígitos em um número inteiro, podemos usar a função **int()**. Por exemplo, int("23") fornece um objeto **int** com valor 23.

Dadas as informações acima, agora podemos corrigir a saída incorreta e gerar a soma dos dois números corretamente:

```
a = int(input())
b = int(input())
s = a+b
print(s)
```