Curso de Computação 1
Introdução à Programação em Python

AULA 4

Variáveis e Atribuição, Strings e Tuplas

Já sabemos que algoritmos são sequências de ações a serem realizadas para atingir um objetivo. Os algoritmos que vimos até agora eram muito simples, se resumindo ao retorno do resultado da avaliação de uma expressão, eventualmente precedido pela realização de testes (avaliação de expressões booleanas) que decidem qual será a expressão a ser avaliada para cálculo do valor a ser retornado pela função. Os cálculos que fizemos até o momento foram todos bastante simples. Muitas vezes, os cálculos que queremos realizar em cada situação são complexos e expressá-los em uma só expressão pode prejudicar a clareza do código. Outras vezes, o mesmo cálculo pode ser necessário mais de uma vez na solução do problema. Para essas situações é importante poder anotar em algum lugar o resultado de um cálculo para uso posterior. Como fazemos quando resolvemos problemas no caderno. O recurso que as linguagens de programação oferecem para "anotar resultados" para uso posterior são as **variáveis**. Nesta aula veremos o conceito e a forma de utilização de variáveis, como são atribuídos valores a elas, e como as variáveis simplificam a construção de funções tornando-as mais legíveis. Além disso, conheceremos um pouco mais o tipo str (strings) do Python e veremos um novo tipo de dados: as tuplas.

1. Variáveis

Vamos começar então abordando o que são variáveis. As informações que geramos ao longo da construção de códigos, por vezes, precisam ser armazenadas para serem utilizadas posteriormente. Essas informações são armazenadas na memória do computador. A memória é dividida em diversos blocos, onde os dados de nossas soluções computacionais são armazenados e acessados (escritos e lidos).
Os blocos de memória são acessados por endereços. Para ter acesso a um bloco de memória é necessário conhecer este endereço, porém trabalhar diretamente com a memória do computador oferece várias dificuldades. Por isso, as linguagens de programação como o Python disponibilizam uma forma facilitada de armazenar e acessar informações na memória, o que conhecemos como variáveis. Variável é uma maneira simbólica de fazer referência a dados armazenados na memória do computador.

Curso de Computação 1
Introdução à Programação em Python

No vídeo a seguir, vamos ver o que são variáveis e como utilizá-las.

Introdução à criação e utilização de variáveis

Toda variável engloba os seguintes aspectos, semelhantes aos parâmetros de uma função:

- Nome (identificador): é a representação simbólica da variável, que será utilizada pelo programador para fazer referência aos dados que ela armazena
- Valor: o que de fato está armazenado
- Tipo: o "tipo de dado" do dado que está armazenado nela

Observe agora os códigos das funções *calcula_tinta* e *calcula_tinta2* a seguir. Estas funções foram ambas desenvolvidas para calcular a quantidade de latas de tinta necessária para pintar uma área, dados:

- a área a ser pintada, em metro quadrados,
- o rendimento da tinta (que área um litro de tinta pinta),
- a capacidade da lata (quantos litros tem em uma lata).

Elas fazem, de fato, a mesma coisa? Qual delas você acha mais fácil de entender?

```
import math

def calcula_tinta(area, rendimento_tinta, capacidade_lata):
    """ quantidade de latas de tinta necessária para pintar uma area, dado:
    a area a ser pintada, em metro quadrados,
    o rendimento da tinta (que área um litro de tinta pinta),
    a capacidade da lata (quantos litros tem em uma lata)
    float, float, float-> float"""

    litros = area/rendimento_tinta
    quant_latas = math.ceil(litros/capacidade_lata)

    return quant_latas

def calcula_tinta2(area, rendimento_tinta, capacidade_lata):
    """ quantidade de latas de tinta necessária para pintar uma area, dado:
    a area a ser pintada, em metro quadrados,
    o rendimento da tinta (que área um litro de tinta pinta),
    a capacidade da lata (quantos litros tem em uma lata)
    float, float, float-> float"""

    return math.ceil(area/rendimento_tinta/capacidade_lata)
```

Converse com seu professor sobre suas considerações na aula síncrona.

2. Comando de atribuição

Como vimos acima, variáveis em Python são criadas através de comandos de atribuição, cujo símbolo é o sinal de igual (=). É importante saber que à esquerda do operador de atribuição, deve existir somente o nome (identificador) da variável. À direita, deve haver uma expressão cujo valor será calculado e armazenado na variável. Uma expressão pode ser um conjunto de operações, aritméticas ou lógicas, utilizados para fazer "cálculos" e pode incluir chamadas de outras funções.

Exemplos de atribuições:

```
>>> X = 500/2
>>> Cor = "vermelho"
>>> Mensagem = Cor + " é uma cor viva." # onde o uso do identificador Cor no lado direito da atribuição representa o valor armazenado na variável (no caso, a string "vermelho")
>>> Y = f(X) # onde f é uma função definida como recebendo um argumento numérico para realizar sua função, seja ela qual for, e X representa o valor armazenado na variável (no caso, o float 250.0)
```

O processamento de uma atribuição em Python, então, é sempre o mesmo: avalia o resultado da expressão à direita do operador de atribuição e associa à variável. Na realidade, quase sempre, pois existe uma exceção, que é chamada de ***alias***. *Alias* significa **sinônimo**. Se a expressão à direita do operador de atribuição for apenas uma outra variável, como por exemplo, var2 = var1, não será realizada uma avaliação da expressão var1. Se var1 fosse avaliada e seu resultado armazenado na memória (em var2) isso significaria que uma cópia do valor de var1 seria associada à variável var2. Nesse caso particular Python faz com que var2 e var1 indiquem a mesma posição da memória, até que uma das duas variáveis apareça como lado esquerdo em outra atribuição, passando a indicar outra posição da memória.

Assista agora ao segundo vídeo para ver um pouco mais sobre o armazenamento e leitura de valores na memória através de variáveis.

Atribuição de Valor

3. Uso de variáveis

No próximo vídeo, vamos verificar e aprender, a partir de um exemplo, como a utilização de variáveis pode simplificar o código de um programa. Veremos também como obter a data e hora atual no Python utilizando o módulo datetime.

O uso de variáveis

4. Tipos e escopo de variáveis

O Python é uma linguagem de programação dinamicamente tipada. O tipo é atribuído de acordo com o valor atribuído à variável. Isto significa que não é necessário declarar previamente o tipo. O tipo de uma variável é determinado pelo tipo de dado do valor que ela está armazenando.
O **escopo** de uma variável está relacionado tanto ao seu "*tempo de vida*" quanto à sua "*visibilidade*". As variáveis definidas em uma função são destruídas quando a função termina, e criadas novamente a cada nova execução da função. Logo, o "tempo de vida" de uma variável é o momento em que a função está em execução. Já a sua "visibilidade" se refere às partes do código onde podemos fazer referência a ela, ou seja, utilizá-la para acessar valores. Isso corresponde ao corpo da função onde ela foi definida.

Resumindo: Uma variável existe apenas dentro da função onde foi definida e não pode ser acessada fora da função.

Observe o código da *produtoSomaDiferenca* a seguir:

```
def produtoSomaDiferenca(a,b):
    """ produto da soma pela diferenca
    float, float-> float"""
    x = a+b
    y = a-b
    return x*y
```

Neste exemplo, as variáveis "x" e "y" são locais, pois existem apenas dentro da função *produtoSomaDiferenca*. Depois que a função é executada, elas são destruídas. Assim, dizemos

que a função é o **escopo** de “x” e “y”. Tentar fazer uso destas variáveis fora da função ocasionaria um erro. No vídeo a seguir será abordado tipo e escopo de variáveis.

Tipo e escopo de variáveis

Atividade: Vamos agora treinar um pouco com exercícios de fixação. Responda as perguntas da atividade “Variáveis ”. Fique atento ao prazo de entrega dessa atividade!

5. Strings

String é o termo frequentemente usado em programação para se referir a uma sequência de caracteres. Em Python, o nome do tipo de dados correspondente é str. Já sabemos como realizar algumas operações com o tipo str: concatenação e replicação. Para concatenar strings usamos o operador de concatenação, que é representado pelo símbolo “+”. Já o operador de replicação é representado pelo símbolo “*”.

Nesta aula estudaremos um pouco mais sobre strings em Python. Por exemplo, é possível fazer algum tipo de verificação sobre qual é o primeiro caractere de uma string? Ou o segundo? Ou o último? Sim. Isso é possível. Para isso, Python provê um mecanismo de indexação de elementos do tipo str. O primeiro caractere tem índice 0, o segundo, índice 1, e assim por diante. A sintaxe para requisitar o valor atribuído a um índice i de uma sequência S é S[i]. Assim, você pode por exemplo escrever uma expressão booleana para verificar se o primeiro caractere de S é ‘A’, por exemplo: S[0] == ‘A’.

Para saber o tamanho (a quantidade de caracteres) de uma string usamos a função **len( )**. Esta função recebe uma string como argumento e retorna o número de caracteres que essa string contém.

Outro operador interessante que pode ser usado com strings é o operador **in**. Ele é um operador que lembra o operador de pertinência de conjuntos da matemática (∈), ou seja, serve para verificar se a string que está à esquerda aparece “dentro” da string que está à direita (se uma é substring da outra), devolvendo o booleano True, caso apareça, e False, caso contrário. O operador **not in** corresponde à negação do **in** devolvendo o booleano True, caso **não** apareça, e False, caso apareça. Aqui vamos exemplificar seu uso com strings.

Exemplo:

```
>>> 'a' in 'Maria'
True
```

```
>>> ‘j’ not in ‘Maria’
True
>>> 'a' not in 'Maria'
False
```

Todo caractere de uma string é indexado, começando do primeiro caractere (índice 0) à esquerda. Por exemplo, a string “Amanda” possui os seguintes índices:

Indice	0	1	2	3	4	5
Caractere	A	m	a	n	d	a

Logo para acessarmos o caracter “n” desta string, basta informar: ***“Amanda”[3]***, pois o valor contido nesta posição indexada é “n”. Existe uma segunda maneira de fazer acesso a uma posição que será abordada no vídeo a seguir.

Indexação de strings

6. Imutabilidade

Um detalhe importante é que não é possível usar o comando de atribuição botando à esquerda uma parte da string, ou seja, uma string indexada. Ou seja, assumindo que uma variável s tem o valor “pedro”, não é permitido escrever s[0] = “P”. Não podemos fazer uma atribuição a uma posição específica de uma string.

As strings, assim como os dados numéricos, são dados *imutáveis*: uma vez criados, não se modificam, ou seja, não podemos mudar um pedaço dela (assim como não podemos mudar um pedaço de um número). Porém, as strings, assim como os dados numéricos, podem ser manipulados a fim de gerar novos dados (através de operações e funções). Assim como ao somar 1+1 obtemos um novo dado (o inteiro 2), operações com strings geram novas strings.

7. Operação de fatiamento

Já vimos como acessar um caractere em uma posição específica de uma dada string através da indexação. Mas podemos também gerar *sub-strings* (trechos da string original) de uma string fornecida e fazemos isto por meio da operação de fatiamento. No exemplo a seguir, podemos observar a operação de fatiamento.

```
>>> nome = "Amanda Pereira"
>>> len(nome)
14
>>> sub_s1 = nome[0:6]
>>> sub_s1
'Amanda'
```

A variável sub_s1 está recebendo a sub-string “Amanda” a partir da string contida na variável nome. O vídeo a seguir explica com maiores detalhes esta operação que é bastante usada.

Veja no vídeo a seguir mais informações sobre fatiamento de strings e imutabilidade.

Fatiamento e Imutabilidade

8. Tuplas

Assim como as strings, uma tupla (tuple, em Python) é um tipo de dados do Python. Formalmente, uma tupla, assim como na matemática, é um elemento do produto cartesiano de dois ou mais conjuntos. Ou seja, uma sequência ordenada de dados. Os dados em uma tupla podem ser de diferentes tipos: inteiros, floats, strings e até mesmo outras tuplas. Tuplas têm as mesmas propriedades de imutabilidade, indexação e fatiamento das strings.

No Python, a sintaxe para representar a tupla é uma sequência de valores separados por vírgulas. Podem ou não estar entre parênteses.

No vídeo a seguir este novo tipo de dado será abordado.

Tuplas

Agora que você já sabe o que é uma tupla, assista o próximo vídeo que traz mais alguns exemplos de uso deste tipo de dado. Repare que as tuplas e as strings têm muito em comum quanto à sua manipulação, incluindo as operações de indexação e fatiamento. Outra coisa em comum é que ambas tuplas e strings são tipos de dados imutáveis.

Curso de Computação 1
Introdução à Programação em Python

Tuplas: mais exemplos

Muito bem. Nesta aula falamos sobre variáveis, atribuições, tipo e escopo de uma variável. Aprendemos a usar strings de maneira mais sofisticada e vimos um novo tipo de dados: Tuplas, cujas formas de manipulação tem muito em comum com as strings. Discutimos sobre a operação de fatiamento e o conceito de imutabilidade. A seguir é apresentado um vídeo sobre erros comuns relacionados a alguns temas aqui abordados.

Erros comuns

Atividade: Vamos agora treinar um pouco com exercícios de fixação. Procure se lembrar de todos os conceitos que você adquiriu ao longo desta aula e responda as perguntas da atividade “Manipulando strings e tuplas”. Fique atento ao prazo de entrega dessa atividade! Se tiver dúvidas, comente sobre elas na aula síncrona ou nos plantões de monitoria.

Não se esqueçam de tirar as dúvidas com o professor durante as aulas síncronas.
Até a próxima aula!