Programação de Computadores I – BCC701

Conteúdo Teórico em Linguagem Python

Módulo 2 – Variáveis e Expressões

2020/1

Elaborado pela Comissão de Unificação da Disciplina BCC701,
com a colaboração de professores e estagiários docentes
http://www.decom.ufop.br/bcc701/

DECOM – Departamento de Computação
ICEB – Instituto de Ciências Exatas e Biológicas
UFOP – Universidade Federal de Ouro Preto

Sumário

MÓDULO 2

Variáveis e Expressões

Em programação de computadores, uma **variável** é um contêiner que serve para armazenar dados na memória do computador. O dado que é armazenado pode ser referenciado usando o nome da variável. O programador pode escolher qualquer nome para uma variável, exceto as palavras-chave do **Python** (palavras reservadas para finalidades específicas), e é uma boa prática escolher nomes significativos que reflitam o conteúdo da variável.

Variáveis não são os próprios valores, você pode pensar nelas como pequenas caixas nas quais você pode guardar coisas, conforme ilustrado na Figura 1.

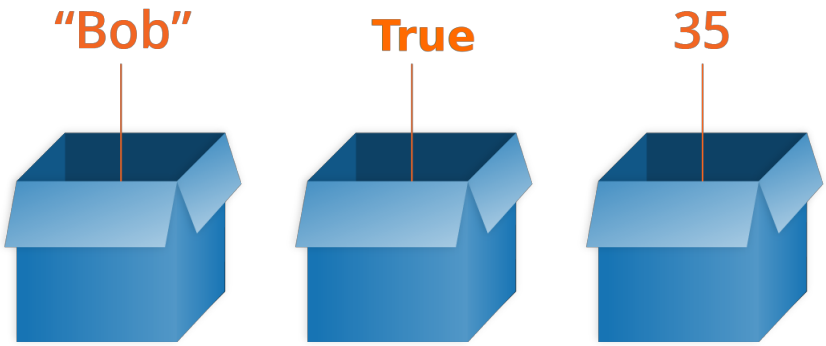

Figura 1: Representação de três variáveis.

2.1 Para que serve uma variável?

Variáveis são criadas para reservar um espaço na memória do computador para armazenar valores. Quando você cria uma variável, você está reservando um espaço na memória. Dependendo do tipo de variável que você criar, você pode armazenar valores inteiros, caracteres, valores decimais, etc.

2.2 Regras para o nome de uma variável

Para dar nomes às variáveis em **Python**, algumas regras devem ser observadas:

- Não podem conter espaços;
- Não podem iniciar com números;
- Além de caracteres alfanuméricos, pode conter somente o caracter: "_";
- Apesar de permitido em **Python**, caracteres acentuados em nomes de variáveis devem ser evitados, pois a maioria das linguagens de programação não permite acentos em nomes de variáveis.
- **Python** diferencia letras maiúsculas e minúsculas, ou seja: `Nome` $\neq$ `nome` $\neq$ `NOME`;
- Além disso, é recomendado que as variáveis tenham nomes significativos. A escolha de nomes significativos para as variáveis ajuda o programador a entender o que o programa faz e a prevenir erros.

Alguns exemplos de nomes **válidos** em **Python**:

- var
- nome1
- total_de_alunos
- José (**dica**: *apesar de válidos em Python, evite nomes de variáveis com acentos!*)

Alguns exemplos de nomes **inválidos** em **Python**:

- 1Aluno (o primeiro caractere é um algarismo)
- total de alunos (tem espaços)
- #funcionarios (o uso do caracter inválido "#")
- %valor (o uso do caracter inválido "%")

2.3 Atribuição de Valores para uma Variável

A Atribuição de Valores é a passagem de informação a determinada variável. Toda variável, por sua definição, pode receber valores ou então, pode ter seu valor alterado.

A linguagem **Python** tem definido que o sinal que conhecemos pelo nome de igual ($=$) será o **operador de atribuição**, conforme ilustrado na Figura 2.

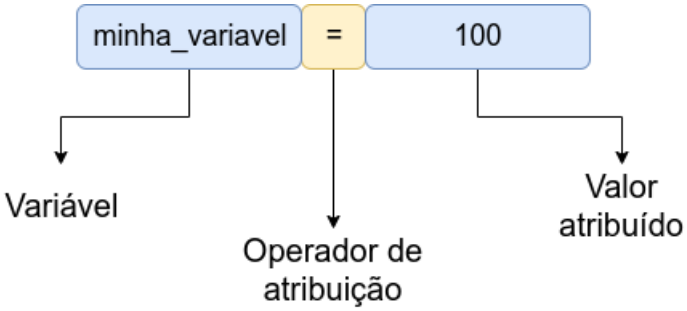

Figura 2: Atribuição de valor para uma variável.

A parte ao lado esquerdo do operador de atribuição sempre receberá o valor definido no lado direito do operador. Sendo assim, a única estrutura de linguagem aceita do lado esquerdo do operador é um nome de variável válido. Por outro lado, qualquer expressão válida, que resulte em um valor da linguagem de programação, poderá ser usado no lado direito do operador de atribuição.

Diferente de outras linguagens de programação, em **Python** não é necessário especificar o tipo de dado na declaração de uma variável, pois a linguagem infere o tipo de uma variável no momento da primeira atribuição.

Um valor a ser atribudo à variável pode ser, por exemplo:

- Um valor inteiro (int): 2, 4, 10, etc.;
- Um valor real (float): 2.7698, 0.0023, etc.;
- Um valor textual (string): `"Um texto qualquer"`;
- Um valor lógico (boolean): **True** ou **False**;
- Outros valores disponibilizados pela linguagem;

A seguir alguns exemplos de uso de variáveis em programas. Recomendamos que você crie os programas de exemplo no **Thonny** e execute alterando os valores para observar os diferentes resultados.

Exemplo 1:

Criar um programa contendo uma variável para guardar a informação da idade de uma pessoa, em seguida apresentar no monitor do computador o valor que está guardado na variável.

Solução:

```
1 idade = 19
2 print(idade)
```

Na primeira linha, o operando à esquerda do sinal de igual é o nome da variável (idade), enquanto o operando à direita do sinal de igual é o valor que será guardado na variável (19).

Na segunda linha, temos o comando **print** que apresenta na tela do computador o valor que está armazenado na variável idade. Este comando é utilizado nos exemplos a seguir de forma semelhante, apenas para imprimir o valor das variáveis, ele será detalhado no próximo módulo.

Exemplo 2:

Criar um programa contendo uma variável para armazenar o nome do cliente de uma loja que, em seguida, apresenta o valor desta variável na tela do computador.

Solução:

```
1 cliente = "João da Silva"
2 print(cliente)
```

Exemplo 3:

Criar um programa que armazene o valor 6 em uma variável e em seguida imprime esse valor na tela do computador.

Solução:

```
1 resultado = 6
2 print(resultado)
```

Exemplo 4:

Criar um programa que armazene o valor 27 em uma variável e em seguida divida esse valor por 3 e armazene em outra variável e por fim, imprima o resultado obtido na tela do computador.

Solução:

```
1 a = 27
2 x = a/3
3 print(x)
```

Exemplo 5:

Criar um programa que armazene o texto '*Um texto qualquer.*' em uma variável e em seguida imprima esse texto na tela do computador

Solução:

```
1 texto = 'Um texto qualquer.'
2 print(texto)
```

Exemplo 6 (Contraexemplo):

Suponha que você deseja resolver o seguinte problema matemático: **Calcule o valor de x a partir do valor de y, dado que o dobro de x equivale ao triplo de y.**

Você naturalmente formulará a seguinte expressão para resolver o problema: $2 * x = 3 * y$. Mas, caso você queira resolver o problema para $y = 10$ em um programa de computador, você não pode criar o programa usando diretamente esta expressão, veja:

Solução INCORRETA:

```
1 y = 10
2 2 * x = 3 * y
3 print(x)
```

A solução anterior está incorreta porque você está cometendo um **erro de sintaxe**, pois a atribuição é definida como:

$$nome\_variavel = valor$$

e não temos no lado esquerdo do símbolo $=$ o nome de uma variável, temos a expressão $2 * x$.

Mesmo na Matemática, para resolver o problema seria recomendado realizar operações para adequar a equação para o cálculo de x, obtendo a expressão: $x = (3 * y)/2$.

E, a partir daí, já podemos representar o cálculo em um programa que respeite a sintaxe da atribuição de um valor para a variável x:

Solução CORRETA:

```
1 y = 10
2 x = (3 * y) / 2
3 print(x)
```

Obs.: Sempre existem diferentes formas de se implementar um programa, este é apenas um exemplo de solução correta.

2.4 Tipos de Dados

Quando você cria uma variável, os dados armazenados nela podem ter muitos tipos diferentes. Por exemplo, você pode armazenar a velocidade de um carro com um valor numérico e, em seguida, armazenar o nome do carro, usando um valor alfanumérico. O **Python** possui vários tipos de dados diferentes que você pode usar para armazenar tipos diferentes de valores, cada um com seu próprio conjunto de operações exclusivas que você pode usar para manipular e trabalhar com as variáveis.

O **Python** possui os variados tipos de variáveis, por exemplo: *Numéricos*, *String*, *Lógico*, *Listas*, *Tuplas*, *Dicionários*. Inicialmente, apresentaremos apenas os dois primeiros tipos.

Se você não está seguro de qual é o tipo de uma variável específica em seu programa, você pode usar a função **type** que retorna o tipo de uma variável.

Exemplo 7:

```
1 n = 14
2 print(type(n))
```

Na primeira linha, a variável **n** recebe o valor 14. É importante observar que, no momento em que fazemos uma atribuição de um valor para uma variável, também estamos definindo o tipo de dado da variável. Nesse caso, a variável **n** está sendo definida como **int**, pois ele recebeu um valor inteiro.

Na segunda linha, o comando **print** apresenta na tela do computador o tipo de dado da variável **n**. Para realizarmos a identificação do tipo, utilizamos a função **type**. Aqui temos como parâmetro da função **print** a função **type**, nesse caso, a função mais interna é executada primeiro, e o seu resultado será o parâmetro para a função mais externa. A função **type**, por sua vez, recebe como parâmetro a variável **n**. Desta forma, na execução do programa, para gerar o resultado esperado do comando da linha 2, primeiro é obtido o resultado de **type(n)**, que corresponde ao tipo do valor atribuído à variável, e em seguida este resultado é repassado para a função **print**, que realizará sua exibição na tela do computador.

2.4.1 Numéricos

Os tipos de dados **numéricos** são criados quando você cria uma variável usando um valor numérico reconhecido pela linguagem **Python**.

Exemplo 8:

Criar um programa capaz de armazenar o preço de um produto e em seguida, o programa deve apresentar o valor dessa variável na tela do computador.

Solução:

```
1 preco_produto = 4
2 print('O valor armazenado na variável é:')
3 print(preco_produto)
4 print('A variável é do tipo:')
5 print(type(preco_produto))
```

Aqui, atribuímos à variável **preco_produto** um valor numérico 4. Em seguida, imprimimos o tipo da variável. O **Python** aloca automaticamente a memória como um valor numérico.

O **Python** suporta 4 tipos de valores numéricos:

1. **int**: esse é o tipo de dado mais usado para manipular números inteiros. Seu valor máximo é 2.147.483.647 e seu valor mínimo é - 2.147.483.648.

2. **long**: esse é o tipo de dado mais usado para manipular números inteiros com valor muito alto.

3. **float**: esse tipo é usado para manipular números reais.

4. **complex**: esse tipo é usado para manipular números complexos.

2.4.2 String

Uma **string** é identificada como um conjunto de caracteres que deve estar entre aspas. O **Python** permite que os valores **string** sejam declarados com aspas simples (') ou duplas (").

Exemplo 9:

Criar um programa capaz de armazenar o nome do cliente e em seguida, o programa deve apresentar o valor dessa variável na tela do computador.

Solução:

```
1 nome_cliente = "João da Silva"
2 print('O valor armazenado na variável é:')
3 print(nome_cliente)
4 print('A variável é do tipo:')
5 print(type(nome_cliente))
```

Na linha 1 do código acima, atribuímos o conjunto de caracteres `"João da Silva"` para a variável **nome_cliente**. Utilizamos o comando **print** para exibir o conteúdo da variável na linha 3 e o tipo da variável na linha 5.

Exemplo 10:

Criar um programa capaz de guardar na memória dois valores e em seguida exibir o conteúdo armazenado em cada variável.

Solução:

```
1 a = 8
2 b = 4.5
3 print('Valores atribuídos: ')
4 print('a =')
5 print(a)
6 print('b =')
7 print(b)
```

2.5 Operadores aritméticos

Operadores aritméticos são símbolos especiais que representam cálculos matemáticos como adição, multiplicação e divisão. Os operadores aritméticos mais usados em **Python** são listados na Tabela 1. O seu funcionamento é como conhecemos na Matemática.

Exemplo 11:

Criar um programa capaz de **somar** dois números e em seguida apresentar o resultado na tela do computador.

Operador	Operação
$+$	Adição
$-$	Subtração
$*$	Multiplicação
$/$	Divisão
$**$	Potência

Tabela 1: Operadores aritméticos.

Solução:

```
1 a = 3
2 b = 6
3 c = a + b
4 print(c)
```

Obs.: Perceba que no exemplo acima foi criada uma variável (**c**) adicional para armazenar o resultado da soma. Ela é dispensável, pois poderia ser feito **print(a + b)**, assim, primeiro o resultado da expressão é calculado para depois o **print** fazer a exibição do resultado. A criação de variáveis adicionais é recomendado para facilitar a organização e entendimento do código, não é essencial neste caso, mas poderá ser em várias outras situações daqui para frente.

Exemplo 12:

Criar um programa para calcular a **diferença** entre dois números e em seguida, o programa deve apresentar o resultado na tela do computador.

Solução:

```
1 a = 3
2 b = 6
3 c = a - b
4 print(c)
```

Exemplo 13:

Criar um programa para calcular a **multiplicação** entre dois números e em seguida, o programa deve apresentar o resultado na tela do computador.

Solução:

```
1 a = 3
2 b = 6
3 c = a * b
4 print(c)
```

Exemplo 14:

Criar um programa para calcular a **divisão** entre dois números e em seguida, o programa deve apresentar o resultado na tela do computador.

Solução:

```
1 a = 6
2 b = 3
3 c = a / b
4 print(c)
```

Exemplo 15:

Criar um programa para calcular 9 elevado ao cubo e em seguida o programa deve apresentar o resultado na tela do computador.

Solução:

```
1 a = 9
2 b = 3
3 c = a ** b
4 print(c)
```

Exemplo 16:

Criar um programa para calcular a soma do primeiro número com o dobro do segundo número e ao final apresentar o resultado na tela.

Solução CORRETA:

```
1 a = 3
2 b = 6
3 c = a + b * 2
4 print(c)
```

Deve-se tomar cuidado, no entanto, ao agrupar expressões com mais de um operador. O resultado será diferente para se você calcular o dobro da soma dos dois números, que estará incorreto em relação ao que o exercício solicita, conforme solução **incorreta** apresentada a seguir.

Solução INCORRETA:

```
1 a = 3
2 b = 6
3 c = (a + b) * 2
4 print(c)
```

Exemplo 17:

Criar um programa capaz de inicializar duas variáveis atribuindo numéricos e em seguida somar essas duas variáveis e por fim exibir o resultado da soma no monitor do computador.

Solução :

```
1 a = 7
2 b = 11
3 c = a + b
4 print('Soma =')
5 print(c)
```

2.5.1 Precedência e associatividade de operadores

A **precedência** de operadores indica qual operador deverá ser executado primeiro:

- Assim, na expressão aritmética $2 + 3 * 6$, a sub-expressão $3 * 6$ é executada primeiro;
- Portanto, tem-se como resultado para a expressão o valor 20.

Exemplo 18:

Criar um programa para calcular e imprimir o resultado da seguinte expressão matemática: $2+3*6$.

Solução :

```
1 resultado = 2 + 3 * 6
2 print(resultado)
```

A **associatividade** define a regra usada quando os operadores possuem a mesma precedência:

- Define-se que a ordem de avaliação será da esquerda para a direita ou o contrário, da direita para a esquerda;
- Para a expressão $A - B + C + D$, o termo $A - B$ é avaliado primeiro;
- Para a expressão $A * *B * *C * *D$, o termo $C * *D$ é avaliado primeiro.

Exemplo 19:

Criar um programa para calcular e imprimir o resultado das seguintes operações matemáticas: $5 - 4 + 2$ e $5 * *4 * *2$.

Solução :

```
1 e = 5 - 4 + 2
2 print(e)
3 e = 5 ** 4 ** 2
4 print(e)
```

Qual a ordem de execução das operações? Utilize a depuração do **Thonny** para observar como o Python executa estas expressões.

2.6 Funções elementares

A linguagem **Python** possui diferentes módulos (ou bibliotecas) que fornecem diversas funções úteis para o programador. Um desses módulos é o **math** que fornece várias funções matemáticas prontas que são úteis para realizar cálculos matemáticos.

Para usar qualquer módulo em **Python** é necessário importá-lo antes. Por exemplo, para usar o módulo **math** temos que acrescentar o seguinte comando no início do programa:

```
1 import math
```

Após o comando de importação do módulo, é possível usar as funções definidas por ele. A importação é necessária apenas uma vez para cada programa.

A seguir exploraremos várias funções elementares, apenas uma não pertence ao módulo **math**, por isso iniciaremos por ela. A Tabela 2 sumariza todas estas funções.

Exemplo 20: função abs(x):

Função para calcular o valor absoluto do argumento x, ou seja, a distância (positiva) entre x e zero.

```
1 resultado = abs(-12)
2 print(resultado)
```

Na primeira linha, a variável **resultado** recebe o valor absoluto de -12, que é calculada a partir da função **abs**. Na segunda linha, temos o comando **print** que apresenta na tela do computador o valor que está armazenado na variável **resultado**.

Observe que neste exemplo não foi necessário importar o módulo **math**, pois ela pertence ao módulo padrão do **Python**, assim como função **print** usada em todos os exemplos.

Outra observação importante é em relação ao "**argumento x**". Uma função que necessita receber algum valor (um ou mais) para realizar sua tarefa, recebe estes valores a partir de **argumentos**, aqui o nome **x** é usado apenas de forma ilustrativa, tanto que ele não apareceu no programa de exemplo. Ao utilizar a função **abs**, que recebe apenas um argumento, nós passamos o valor que será utilizado pela função, neste caso o valor -12. Qualquer valor numérico ou expressão que resulte em valor numérico poderá ser utilizado, por exemplo, o resultado de **abs(1 - 13)** será o mesmo de **abs(-12)**, já que $1 - 13 = -12$.

Agora exploraremos algumas funções do módulo **math**.

Exemplo 21: função sqrt(x):

É uma função para calcular a raiz quadrada de do argumento x.

```
1 import math
2 resultado = math.sqrt(9)
3 print(resultado)
```

A primeira linha importa o módulo **math**, permitindo assim o de todas as suas funções dentro do programa. Na segunda linha, a variável **resultado** recebe a raiz quadrada de 64, que é calculada a partir da função **sqrt**. Na terceira linha, temos o comando **print** que apresenta na tela do computador o valor que está armazenado na variável **resultado**.

Exemplo 22: função cos(x):

Função para calcular o cosseno do argumento x, em radianos.

```
1 import math
2 resultado = math.cos(30)
3 print(resultado)
```

A primeira linha importa o módulo **math**. Na segunda linha, a variável **resultado** recebe o cosseno de 30 radianos, que é calculada a partir da função **cos**. Na terceira linha, temos o comando **print** que apresenta na tela do computador o valor que está armazenado na variável **resultado**.

Exemplo 23: função tan(x):

Função para calcular a tangente do argumento x, em radianos.

```
1 import math
2 resultado = math.tan(3.1417)
3 print(resultado)
```

A primeira linha importa o módulo **math**. Na segunda linha, a variável resultado recebe a tangente de 3.1417, que é calculada a partir da função **tan**.

Na terceira linha, temos o comando **print** que apresenta na tela do computador o valor que está armazenado na variável resultado.

Exemplo 24: função sin(x):

Função para calcular o seno de do argumento x.

```
1 import math
2 resultado = math.sin(3.1417)
3 print(resultado)
```

Exemplo 25: função log(x):

Calcula o **logaritmo natural** do argumento x.

```
1 import math
2 resultado = math.log(10)
3 print(resultado)
```

Exemplo 26: função log10(x):

Calcula o logaritmo de x na base 10.

```
1 import math
2 resultado = math.log10(1000)
3 print(resultado)
```

Exemplo 27: função log2(x):

Calcula o logaritmo de x na base 2.

```
1 import math
2 resultado = math.log2(256)
3 print(resultado)
```

Exemplo 28: função log(x, b):

Calcula o logaritmo de x na base b.

```
1 import math
2 resultado = math.log(81, 3)
3 print(resultado)
```

Calcular o logaritmo na base 2 ou base 10 usando **log2(x)** e **log10(x)**, resulta em valores mais precisos do que **log(x, 2)** e **log(x, 10)**, já que são funções específicas para estas duas bases e utilizam algoritmos mais precisos para realizar os cálculos necessários.

2.6.1 Resumo das funções elementares

A Tabela 2 apresenta um resumo das funções apresentadas na seção anterior. Lembre-se de que todas são fornecidas pelo módulo **math**, à exceção apenas da função **abs**.

2.7 Constantes

A linguagem **Python** possui algumas constantes, como por exemplo: **pi**, **número de Euler** e **Tau**. Constantes úteis em diversos cálculos matemáticos são fornecidas no módulo **math**. A Tabela 3 apresenta um resumo de algumas das constantes fornecidas.

Função	Nome	Exemplo	Resultado
Valor absoluto	**abs**	abs(-2)	2
Raiz quadrada	math.**sqrt**	sqrt(9)	3
Cosseno	math.**cos**	cos(30)	0.1542514
Tangente	math.**tan**	tan(7.3456)	1.7945721
Seno	math.**sin**	sin(pi)	1.2246e-16
Parte inteira	math.**trunc**	trunc(5.2)	5
Menor inteiro maior que	math.**ceil**	ceil(5.2)	6
Maior inteiro menor que	math.**floor**	floor(5.2)	5
Potência	math.**pow**	pow(2,3)	8
Logaritmo natural	math.**log**	log(148.41315910257657)	5
Logaritmo base 10	math.**log10**	log10(1000)	3
Logaritmo base 2	math.**log2**	log2(256)	8
Logaritmo qualquer base	math.**log**	log(81,3)	4

Tabela 2: Funções elementares.

Constante	Nome	Valor
Pi (π)	math.**pi**	3.141592..., até a precisão disponível
Número neperiano (e)	math.**e**	2.718281..., até a precisão disponível
Tau (τ)	math.**tau**	6.283185..., até a precisão disponível
Infinito (∞)	math.**inf**	Uma representação do infinito positivo

Tabela 3: Algumas constantes do módulo **math**.

A utilização das constantes é semelhante ao das funções elementares apresentadas anteriormente, à exceção de que para constantes **não é necessário fazer a passagem de argumentos**, conforme exemplos a seguir.

Exemplo 29: constante pi (π):

A constante matemática π é utilizada em diversas situações. Uma delas é para se calcular o comprimento de uma circunferência, através da equação:

$$C = 2 * \pi * r,$$

onde C é o comprimento e r o raio da circunferência.

O programa a seguir pode ser usado para calcular o comprimento de uma circunferência de raio igual a 100 metros.

```
1 import math
2 r = 100
3 C = 2 * math.pi * r
4 print(C)
```

Observe que, para usar a constante π nós importamos o módulo **math** e utilizamos **math.pi**, que representa o valor numérico de π com a maior precisão possível na linguagem **Python**.

Exemplo 30: número de Euler (ou neperiano) (e):

O número neperiano também é muito útil em diversas situações, como em logaritmos naturais por exemplo. Uma outra utilização é para o cálculo de juros compostos, onde o *montante* após um tempo

t, M_t, a uma taxa de juros composto r é definido como:

$$M_t = M_i * e^{r*t},$$

onde M_i é o *montante* inicial.

O programa a seguir pode ser utilizado para calcular o valor a ser pago considerando um montante inicial de R$ 1.000,00, com juros composto a uma taxa mensal de 1,5%, durante 12 meses.

```
1 import math
2 Mi = 1000
3 r = 0.015
4 t = 12
5 Mt = Mi * math.e ** (r*t)
6 print(Mt)
```

2.8 Exercícios Propostos

Exercício 1

Qual é o valor impresso ao final da seguinte sequência de comandos?

```
1 dia = "quinta-feira"
2 dia = 32.5
3 dia = 19
4 print(dia)
```

Exercício 2

O seguinte nome é válido para uma variável em **Python**? Justifique sua resposta.

```
1 Uma_boa_nota_é_A+ = 9
```

Exercício 3

Um prêmio em dinheiro será dividido entre três ganhadores de um concurso.

- O primeiro ganhador receberá 46% do valor total do prêmio
- O segundo ganhador receberá 32% do valor total do prêmio
- O terceiro receberá o restante

Escreva um programa que defina uma variável com o valor **780000** para o prêmio e, em seguida, calcule e imprima a quantia que deve ser distribuída para cada um dos ganhadores.

Exemplos de execução:

Exemplo 1:

```
O primeiro ganhador deve receber R$:
358800.0
O segundo ganhador deve receber R$:
249600.0
O terceiro ganhador deve receber R$:
171600.0
```

Exercício 4

Faça um programa que defina a largura e a altura de uma parede em 2.8 e 5.2 metros em duas variáveis. Posteriormente, calcule a sua área e a quantidade de tinta necessária para pintá-la, sabendo que cada litro de tinta pinta uma área de 2 metros quadrados.

Exemplos de execução:

Exemplo 1:

```
Área da parede em metros quadrados:
14.559999999999999
Litros de tinta necessários para pintar essa parede:
7.279999999999999
```

Exercício 5

Escreva um programa que defina, em variáveis, a quantidade de Km percorridos por um carro alugado (com valor 879) e a quantidade de dias pelos quais ele foi alugado (com valor 6). Calcule o preço a pagar, sabendo que o carro custa R$ 60,00 por dia e R$ 0,15 por Km rodado.

Exemplos de execução:

Exemplo 1:

```
O cliente deve pagar R$:
491.85
```

Exercício 6

Faça um programa que defina uma variável com área de um quadrado no valor de 256 e calcule o valor de cada lado do quadrado.

Exemplos de execução:

Exemplo 1:

```
Cada lado tem comprimento igual à:
16.0
```

Exercício 7

Faça um programa que defina uma variável com o raio de uma circunferência no valor de 1.2 e calcule sua área. Utilize a fórmula:

$$\text{area} = \pi * (\text{raio}^2)$$

Utilize a constante π definida na linguagem.

Ao final, o seu programa deve apresentar o valor da área da circunferência calculada.

Exemplos de execução:

Exemplo 1:

```
A área dessa circunferência é:
4.523893421169302
```

2.9 Solução dos Exercícios Propostos

Exercício 1

Qual é o valor impresso ao final da seguinte sequência de comandos?

```
1 dia = "quinta-feira"
2 dia = 32.5
3 dia = 19
4 print(dia)
```

Resposta: 19, pois este corresponde ao último valor atribuído à variável durante a execução do programa até o comando **print**.

Exercício 2

O seguinte nome é válido para uma variável em **Python**? Justifique sua resposta.

```
1 Uma_boa_nota_é_A+ = 9
```

Resposta: Não, pois a linguagem não permite o uso do símbolo "+" na declaração do nome da variável.

Exercício 3

Um prêmio em dinheiro será dividido entre três ganhadores de um concurso.

- O primeiro ganhador receberá 46% do valor total do prêmio
- O segundo ganhador receberá 32% do valor total do prêmio
- O terceiro receberá o restante

Escreva um programa que defina uma variável com o valor **780000** para o prêmio e, em seguida, calcule e imprima a quantia que deve ser distribuída para cada um dos ganhadores.

Exemplos de execução:

Exemplo 1:

```
O primeiro ganhador deve receber R$:
358800.0
O segundo ganhador deve receber R$:
249600.0
O terceiro ganhador deve receber R$:
171600.0
```

Solução:

```
1 premio = 780000
2 primeiro_ganhador = premio * 46 / 100
3 segundo_ganhador = premio * 32 / 100
4 terceiro_ganhador = premio - primeiro_ganhador - segundo_ganhador
5 print('O primeiro ganhador deve receber R$')
6 print(primeiro_ganhador)
```

```
7  print('O segundo ganhador deve receber R$')
8  print(segundo_ganhador)
9  print('O terceiro ganhador deve receber R$')
10 print(terceiro_ganhador)
```

Exercício 4

Faça um programa que defina a largura e a altura de uma parede em 2.8 e 5.2 metros em duas variáveis. Posteriormente, calcule a sua área e a quantidade de tinta necessária para pintá-la, sabendo que cada litro de tinta pinta uma área de 2 metros quadrados.

Exemplos de execução:

Exemplo 1:

```
Área da parede em metros quadrados:
14.559999999999999
Litros de tinta necessários para pintar essa parede:
7.279999999999999
```

Solução:

```
1 altura = 2.8
2 largura = 5.2
3 area = altura * largura
4 quantidade_tinta = area / 2
5 print('Área da parede em metros quadrados')
6 print(area)
7 print('Litros de tinta necessários para pintar essa parede')
8 print(quantidade_tinta)
```

Exercício 5

Escreva um programa que defina, em variáveis, a quantidade de Km percorridos por um carro alugado (com valor 879) e a quantidade de dias pelos quais ele foi alugado (com valor 6). Calcule o preço a pagar, sabendo que o carro custa R$ 60,00 por dia e R$ 0,15 por Km rodado.

Exemplos de execução:

Exemplo 1:

```
O cliente deve pagar R$:
491.85
```

Solução:

```
km_percorridos = 879
numero_diarias = 6
custo_km_percorridos = km_percorridos * 0.15
custo_diarias = numero_diarias * 60
custo_total = custo_km_percorridos + custo_diarias
print('O cliente deve pagar R$')
print(custo_total)
```

Exercício 6

Faça um programa que defina uma variável com área de um quadrado no valor de 256 e calcule o valor de cada lado do quadrado.

Exemplos de execução:

Exemplo 1:

```
Cada lado tem comprimento igual à:
16.0
```

Solução:

```
import math
area_quadrado = 256
comprimento_quadrado = math.sqrt(area_quadrado)
print('Cada lado tem comprimento igual à:')
print(comprimento_quadrado)
```

Exercício 7

Faça um programa que defina uma variável com o raio de uma circunferência no valor de 1.2 e calcule sua área. Utilize a fórmula:

$$\text{area} = \pi * (\text{raio}^2)$$

Utilize a constante π definida na linguagem.

Ao final, o seu programa deve apresentar o valor da área da circunferência calculada.

Exemplos de execução:

Exemplo 1:

```
A área dessa circunferência é:
4.523893421169302
```

Solução:

```
import math
raio_circunferencia = 1.2
area_circunferencia = math.pi * raio_circunferencia ** 2
print('A área dessa circunferência é:')
print(area_circunferencia)
```