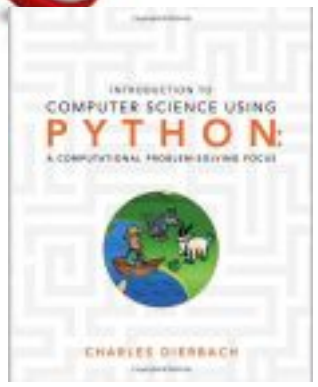
Linguagem de Programação

03: OPERADORES, EXPRESSÕES LÓGICAS E ESTRUTURAS SEQUENCIAIS

Nossos **objetivos** nesta aula são:

- Formalizar o conceito de operador, conhecer os operadores aritméticos e expressões aritméticas.
- Identificar e classificar os diversos tipos de dados que serão utilizados em seus programas.
- Formalizar a estrutura sequencial e escrever programas que recebem dados, efetuem cálculos e produzem resultados.



A referência para esta aula são as **seções 2.3** (Operadores) **e 2.4** (Expressões e Tipos de Dados) **do Capítulo 2** (Data and Expressions) do livro:

DIERBACH, C. *Introduction to Computer Science Using Python: A Computational Problem Solving Focus.* 1st Edition, New York: Wiley, 2012.

Agora que conhecemos os tipos **numéricos** e **string** em Python, vamos ver como fazer operações com esses tipos.

Um **operador** é um símbolo que representa a operação que pode ser realizada em um ou mais operandos. Operadores que atuam sobre um operando são chamados **operadores unários**, e operadores que atuam sobre dois operandos são chamados de **operadores binários**.

Operadores Aritméticos

Operadores aritméticos são aqueles cujos operandos são dados de tipo numérico (inteiros ou em ponto flutuante. A tabela a seguir lista os operadores básicos de acordo com a sintaxe do Python, e de maneira geral, a maioria das linguagens de programação tem operadores equivalentes (ou funções que os substituem).

Operadores	**Significado**	**Exemplo**	**Resultado**
`-x`	Negação	`-(10)`	`-10`
`x + y`	Soma	`2 + 4`	`6`
`x - y`	Subtração	`2 - 4`	`-2`
`x * y`	Multiplicação ou produto	`2 * 4`	`8`
`x / y`	Divisão	`25 / 10`	`2.5`
`x // y`	Divisão truncada	`25 // 10`	`2`
`x % y`	Módulo ou resto da divisão	`25 % 10`	`5`
`x ** y`	Exponenciação	`2 ** 4`	`16`

Observações:

- O operador (-) pode ser um operador unário (negação) ou um operador binário (subtração). Todos os outros operadores são sempre binários.
- A exponenciação pode ser utilizada tanto para números inteiros quanto para números em ponto flutuante na base e no expoente.
- Exemplos:

```
>>> 2 ** 4
16
>>> 2.5 ** 1.2
3.00281108
```

- O operador módulo ou resto de divisão (%), resulta em um ciclo de valores:

Modulo 7		Modulo 10		Modulo 100	
0 % 7	**0**	0 % 10	**0**	0 % 100	**0**
1 % 7	**1**	1 % 10	**1**	1 % 100	**1**
2 % 7	**2**	2 % 10	**2**	2 % 100	**2**
3 % 7	**3**	3 % 10	**3**	3 % 100	**3**
4 % 7	**4**	4 % 10	**4**	.	.
5 % 7	**5**	5 % 10	**5**	.	.
6 % 7	**6**	6 % 10	**6**	96 % 100	**96**
7 % 7	0	7 % 10	**7**	97 % 100	**97**
8 % 7	1	8 % 10	**8**	98 % 100	**98**
9 % 7	2	9 % 10	**9**	99 % 100	**99**
10 % 7	3	10 % 10	0	100 % 100	0
11 % 7	4	11 % 10	1	101 % 100	1
12 % 7	5	12 % 10	2	102 % 100	2

Para praticar: **seguintes:**

Qual o resultado das operações abaixo?

```
a) 1/3
b) 1/3 + 1/3
c) 3 * (1/3)
d) 3*1/3
e) 10 + 35
f) -10 + 35
```

```
g) 4 * 2
h) 4 ** 2
i) 45 / 10
j) 45 // 10 k) 2025 % 10 l)
2025 % 1000
```

Dê o resultado para cada um dos

(a) 22 * 3 (b) 15 % 4 (c) 3 ** 2

Dê o resultado exato de cada uma das divisões seguintes:

(a) 5 / 4 (b) 5 // 4 (c) 5.0 // 4

Quantos operandos existem na expressão aritmética seguinte? 2 * 24 + 60 - 10 (a) 4 (b) 3 (c) 7

Quantos operadores binários existem na expressão aritmética seguinte? - 10 + 25 / (16 + 12) (a) 2 (b) 3 (c) 4

EXPRESSÕES E TIPOS DE DADOS

Uma expressão é uma combinação de símbolos (operandos e operadores) que serão avaliados. Assim como na matemática, em programação uma expressão pode conter variáveis, como por exemplo: `4 + 3*k`

As expressões que avaliam valores do tipo numérico são chamadas de **expressões aritméticas**.

Para praticar:

Qual o resultado das expressões abaixo?

```
                                3*(4-1)
a) (2+3) * 4 c) 2 + (3*4)
b) 2 + ((3*4) - 8) d) 2 +
```

Para avaliarmos corretamente as expressões acima, é importante sabermos em que ordem devemos realizar as operações, ou seja, é necessário conhecer a **precedência dos operadores.** O natural seria considerar a precedência da matemática, porém cada linguagem de programação tem sua própria regra, que pode ser igual ou não à da matemática.

Além da precedência dos operadores, devemos conhecer também como cada linguagem trata os operadores que tenham o **mesmo nível de prioridade.** À ordem em que isso ocorre damos o nome de **associatividade** dos operadores.

A tabela a seguir traz a prioridade e associatividade adotada pelo Python.

Prioridade	Operador	Associatividade
1	**	Direita para esquerda ←
2	- (negação)	→ Esquerda para direita
3	* / // %	→ Esquerda para direita
4	+ e – (subtração)	→ Esquerda para direita

Para praticar:

Desenvolva as expressões abaixo usando a tabela de precedência e calcule seu resultado: (a) 4 + 2**5 // 10 (b) 5 + 42%10

(c) -3*2 + 2/5*10 + 3 (d) 2**4 / 2 + 2%5*3

TIPO DE DADO

Um **tipo de dado** é um conjunto de valores e um conjunto de operadores que podem ser aplicados a esses valores.

Exemplo:

- O tipo **`int`** define o conjunto de números inteiros e as operações (exponenciação, negação, multiplicação, divisão, divisão truncada, resto de divisão, adição e subtração) que podem ser aplicadas no conjunto dos inteiros.

Tem-se também o tipo lógico ou tipo booleano, chamado **`bool`**, que pode assumir os valores **`True`** para verdadeiro e **`False`** para falso.

Exemplo:

- `resultado = True`
- `aprovado = False`

Observe que o valor booleano é diferente da string `'True'` ou `'False'`, por exemplo.

Tipagem de dados

Podemos citar três conceitos relativos a tipagem de dados de uma linguagem de programação:

- **Tipagem estática/dinâmica:** Diz respeito a capacidade de uma linguagem em mudar ou não o tipo de uma variável durante a execução do programa.
- **Tipagem forte/fraca:** Diz respeito à verificação de tipo que é feita ou não durante uma operação.
- **Inferência de tipo:** Diz respeito capacidade da linguagem em inferir o tipo de uma variável em função do seu valor ou não.

No caso da linguagem Python, temos que ela é de tipagem **dinâmica** (as variáveis podem mudar de tipo durante a execução), **forte** (é feita uma verificação de tipo no momento de realizar uma operação e se os tipos forem incompatíveis com a operação é levantado um erro) e **com inferência de tipo** (não é necessário dizer explicitamente o tipo das variáveis criadas, pois o Python "advinha" o tipo da variável em função do dado que ela está recebendo).

FUNÇÕES MATEMÁTICAS

Podemos usar um conjunto variado de funções como as trigonométricas (ex: sin, cos, tan, etc) e logarítmicas (ex: log) e também constantes matemáticas (ex: pi). Esse conjunto de funções pode variar dependendo da linguagem de programação utilizada. No nosso curso utilizaremos as funções disponíveis na biblioteca de funções matemáticas da linguagem Python

Veja a seguir a tabela com algumas funções:

Sintaxe Descrição

`acos(x)` retorna o arco cosseno de `x`

`asin(x)` retorna o arco seno de `x`

`atan(x)` retorna o arco tangente de `x`

`ceil(x)` retorna o menor inteiro maior ou igual a `x` como `int` `cos(x)` retorna o cosseno de `x`

`degrees(r)` converte radianos para graus

`exp(x)` retorna o exponencial de `x` (e^x)

`fabs(x)` retorne o valor absoluto de `x`

`factorial(x)` retorna `x!`

`floor(x)` retorna o maior inteiro menor ou igual a `x` como um `int` `log(x,b)` retorna $\log_b x$ (se `b` for omitido, retorna `log x` na base e) `log10(x)` retorna $\log_{10} x$

`modf(x)` retorna a parte fracionária e a parte inteira como dois `floats` `pow(x,y)` retorna x^y
`radians(g)` converte graus para radianos
`sin(x)` retorna o seno de `x`
`sqrt(x)` retorna a raiz quadrada de `x`
`tan(x)` retorna a tangente de `x`
`trunc(x)` retorna a parte inteira de `x` como um `int`; igual a `int(x)`

Para praticar:

Escreva as seguintes expressões matemáticas utilizando os operadores e funções vistos até agora:

a) $y = \frac{a + \sqrt[2]{3b}}{5x^3}$

b) $x = y + \sqrt{\frac{2b}{a+b}}$

ESTRUTURA SEQUENCIAL

Nos exercícios construídos anteriormente sempre foi utilizada a mesma sequência de instruções: receber dados iniciais, realizar operações matemáticas e apresentar resultados finais.

Esta estrutura de controle, caracterizada pela execução das instruções na ordem em que foram escritas é denominada de Estrutura Sequencial.

Podemos observar esta estrutura no exemplo já trabalhado nas aulas anteriores em que construímos um programa que exibe a soma de dois números informados pelo usuário.

```
n1 = int(input('Digite um número: '))
n2 = int(input('Digite outro: '))
soma = n1 + n2
print('A soma é:', soma)
```

Para praticar:

Escreva um programa que:

1) Leia dois números inteiros e exiba o quadrado da diferença do primeiro valor pelo segundo.

2) Receba o preço da conta de IPTU de uma casa, calcule e mostre:

a) o valor a pagar se for pago atrasado, com multa de 5% sobre o valor digitado; b) o valor a pagar se for pago adiantado, com desconto de 10% sobre o valor digitado.

3) Leia um número inteiro de 3 dígitos, determine e apresente o número invertido usando apenas os operadores aritméticos. (exemplo: número informado = 345, número apresentado = 543).

4) Leia as coordenadas (x,y) de dois pontos no plano cartesiano, calcule e mostre a distância entre os dois pontos.

$$d = \sqrt{(x\,x\,)\,(y\,y\,)}_{2-1}{}^{2}+{}_{2-1}{}^{2}$$

5) Receba a quantidade de dinheiro, em reais, que uma pessoa tem para fazer uma viagem ao exterior. Receba, também, o valor da cotação do dólar do dia. Calcule e apresente o valor convertido em dólares.

6) Faça um programa que peça a temperatura em graus Fahrenheit, transforme e mostre a temperatura em graus Celsius, sabendo que:

$$c = (f - 32) \bullet 5/9$$

COMO TESTAR UM PROGRAMA?

Existem formas de testar a corretude de programas. Uma das formas é o teste de **caixa-preta**, cujos passos resumidamente são:

a) Escolhe-se uma entrada cuja saída correta correspondente seja conhecida, sem precisar usar o programa que será testado;

b) Simula-se a execução do programa usando a entrada escolhida (TESTE DE MESA); c) Compara-se a saída simulada com aquela teórica inicialmente esperada; d) Caso haja diferença entre a saída teórica e a saída real, muito provavelmente o programa está incorreto e necessita de correções. As devidas modificações são feitas e retorna-se ao passo (b);

e) Caso não existam divergências entre as saídas, opta-se por: (I) encerrar os testes ou (II) submeter o programa a outros casos de teste.

Note que **testes de caixa-preta não consideram a estrutura interna** do programa, ou seja, o algoritmo usado para construir o programa, assim como a linguagem de programação em que foi implementado, não importam. Veja a ilustração simplificada do procedimento de teste de caixa-preta na Figura 1.

Figura 1 – Simplificação do funcionamento do teste de caixa-preta.

Lembre-se, **testes não garantem**, necessariamente, **que um programa está correto**. Um programa que seja aprovado em todos os casos de teste aos quais foi submetido não está obrigatoriamente correto, pois pode haver um caso de teste ausente que geraria uma falha.

Para garantir a corretude de um programa por meio de testes é necessário fazer um **teste exaustivo**. Testes exaustivos submetem o programa à todas as entradas esperadas possíveis, o que muitas vezes é impraticável. Imagine quantas possíveis entradas esperadas existem para um programa que soma dois números inteiros e exibe o resultado. Infinitas!

Uma forma de garantir a corretude de um programa sem precisar construir testes é por meio de **prova formal**, porém esse método exige maiores conhecimentos matemáticos, técnicas avançadas de análise de algoritmos e criatividade. Provas formais costumeiramente demandam mais recursos financeiros e tempo para serem satisfatoriamente concluídas em programas mais extensos.

Geralmente, para **sistemas não críticos** (aqueles que não gerenciam laboratórios com doenças altamente contagiosas; não administram usinas nucleares; não controlam produção de foguetes; não automatizam cirurgias, aviões, trens, etc.) são montadas sequências de testes com boa abrangência, testando principalmente as extremidades das possíveis entradas esperadas. Isso amplia o nível de confiança nas soluções propostas. Em nossas aulas optamos por essa abordagem simplificada para o teste de programas.

O uso de **testes de mesa** pode auxiliar na execução de testes de caixa-branca, pois permite que toda instrução que implique em mudança nas variáveis seja representada em uma linha-coluna da tabela. No caso de testes de mesa com estruturas condicionais, é facilitador acrescentar uma coluna com a expressão da condição de seleção, sendo útil para perceber erros na definição da condição (frequentes com iniciantes).

EXEMPLO

Vamos aplicar a técnica da caixa preta ao exercício da soma de dois números, cujo código é o seguinte:

```
n1 = int(input('Digite um número: '))
n2 = int(input('Digite outro: '))
soma = n1 + n2
print('A soma é', soma)
```

Vamos pensar em dois testes:

ID do teste	Entradas	Saída prevista
1	5 e 6	A soma é 11
2	-10 e 5	A soma é -5

Para a elaboração do teste de mesa criamos uma tabela com as variáveis usadas e com um coluna (Tela) que mostra o que está sendo exibido na tela:

n1	n2	soma	Tela
5	6	11	A soma é 11

Em seguida, a cada comando executado no programa, registramos o valor que as variáveis envolvidas vão recebendo, conforme mostrado na tabela a seguir, na qual executamos o teste 1:

Comando 1 Teste de Mesa após a execução do comando 1 `n1 = int(input('Digite um número: '))` **n1 n2 soma Tela** 5

Comando 2 Teste de Mesa após a execução do comando 2 `n2 = int(input('Digite outro: '))` **n1 n2 soma Tela** 5 6

Comando 3 Teste de Mesa após a execução do comando 3 `soma = n1 + n2` **n1 n2 soma Tela** 5 6 11

Comando 4 Teste de Mesa após a execução do comando 4 `print('A soma é', soma)` **n1 n2 soma Tela** 5 6 11 A soma é 11

1º Teste de mesa - resultado

n1	n2	soma	Tela
			5 6 11 A soma é 11

2º Teste de mesa - resultado

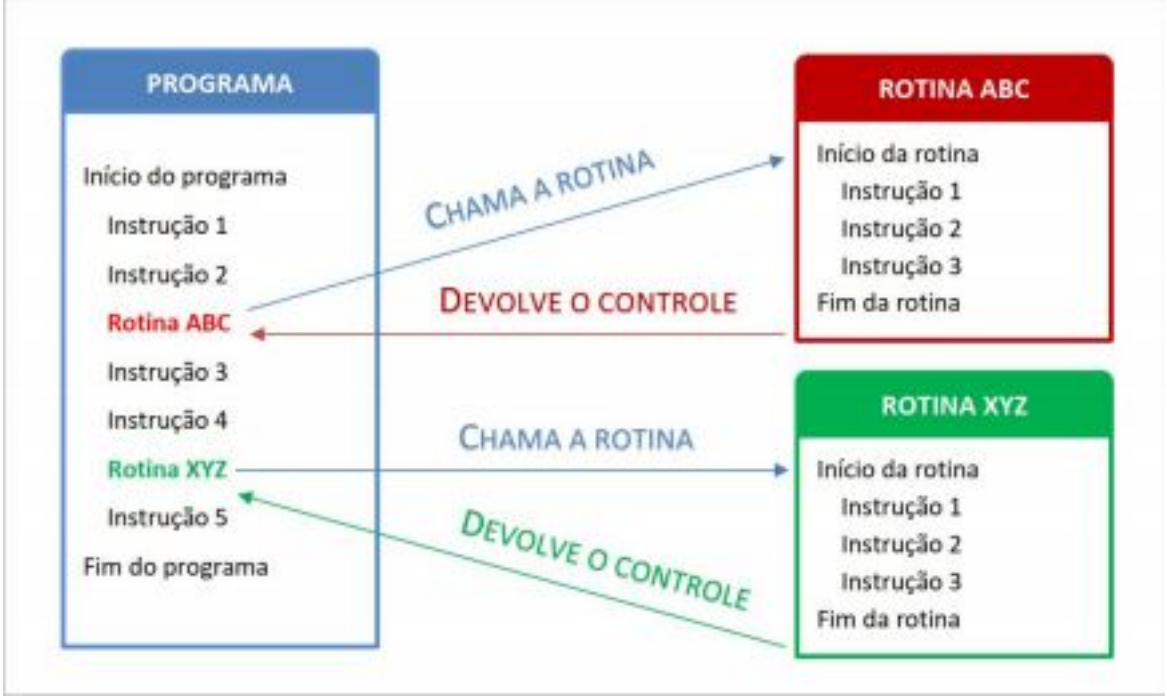
n1	n2	soma	Tela
			-10 5 -5 A soma é -5

Após a condução dos testes de mesa, podemos verificar os resultados:

ID do teste	Entradas	Saída prevista	Saída prevista	Resultado
1	5 e 6	A soma é 11	A soma é 11	ok
2	-10 e 5	A soma é -5	A soma é -5	ok

ROTINAS E FUNÇÕES

Uma **rotina** é definida como um grupo de instruções que executa alguma tarefa bem definida e identificada por um nome. Algoritmos maiores, por exemplo programas completos, podem ser compostos por diversas rotinas que são independentes do programa principal (quanto maior a independência melhor).



Uma rotina pode ser invocada ou chamada quantas vezes for necessário em um dado programa. No momento em que é chamada (a execução do programa chega na linha do código onde está o nome da rotina), o controle de execução é redirecionado para o código da rotina.

Enquanto a rotina está em execução, o código posterior à linha com o nome da rotina fica em estado de espera, pois não é possível seguir em frente enquanto a rotina não terminar sua tarefa.

Quando uma rotina termina de ser executada, o controle da execução retorna automaticamente para o ponto onde foi chamada e o processamento continua com as instruções sucessoras.

As rotinas podem ser pré-definidas da própria linguagem de programação ou construídas pelo programador quando necessário. Existem diversas rotinas prontas em Python como as vistas acima, por exemplo. Dentre as vantagens do desenvolvimento de Rotinas e/ou Funções, podemos citar:

1) **Reutilização do código:** não é necessário reescrever o código do algoritmo toda vez que precisarmos utilizá-lo;
2) **Redução dos testes**: como o algoritmo já foi testado no momento de sua implementação, não é necessário efetuar o teste novamente a cada chamada que fizermos. 3) **Divisão do trabalho:** Através da divisão do trabalho em funções, pode-se dividir o trabalho de construção de um software, o que ajuda também na sua organização.

EXEMPLO

No exercicio resolvido acima, caso tenhamos que efetuar a soma de dois números duas vezes, temos que construir um programa repetindo a entrada o processamento e a saída duas vezes, fazendo com que o nosso programa fique desnecessariamente longo.

Para evitar isso, podemos fazer uso de funções. A criação de uma função é feita com o comando

def, enquanto a chamada (utilização) da função é feita colocando parênteses após o nome da função, da mesma forma que para as funções do Python que já utilizamos (*print*, *input*, etc.).

```
# criação de uma função
def nome_da_funcao():
  ...

# Execução (chamada) dessa função
nome_da_funcao()
```

À seguir temos um exemplo de código criado para realizar a mesma operação (receber dois números, somá-los e exibir o resultado) duas vezes, e em seguida, temos o mesmo trecho de código reescrito com o uso de uma função.

Estrutura sequencial sem o uso de funções:

```
n1 = int(input('Digite um número: '))
n2 = int(input('Digite outro: '))
soma = n1 + n2
print('A soma é:', soma)

n1 = int(input('Digite um número: '))
n2 = int(input('Digite outro: '))
soma = n1 + n2
print('A soma é:', soma)
```

Para transformar o código acima em um que faça uso de funções, devemos:

1) Criar a função com o mesmo código que havia no programa anterior; e
2) Trocar cada repetição do código pela chamada da função criada.

```
# definição da função
def somar():
 n1 = int(input('Digite um número:
')) n2 = int(input('Digite outro:
')) soma = n1 + n2
 print('A soma é:', soma)

# chamadas da função
somar()
somar()
```


No exemplo à esquerda, inicialmente é criada uma função com o comando *def*, em seguida essa função é chamada duas vezes, e para cada vez, o código da função é executado.

EXERCÍCIOS PROPOSTOS

1) Escreva um programa que permite ao usuário digitar dois números inteiros e exibir o resultado

para cada uma das seguintes operações: +, -, *, /, //, %, **. Por exemplo: se o usuário digitar 7 e 5, a saída do seu programa deverá ser:

a) 7 + 5 = 12 b) 7 - 5 = 2 c) 7 * 5 = 35 d) 7 / 5 = 1.40 e) 7 // 5 = 1 f) 7 % 5 = 2 g) 7 ** 5 = 16807

2) Escreva as seguintes expressões matemáticas utilizando operadores e funções vistas nessa aula:

a) $3x\ y\ xy\ \frac{-b \pm \sqrt{b} - 4ac}{2}$

b) $x = \frac{}{2a}$

c) $y = log_3 x\ e\ \pi$

3) Modifique o programa de conversão de temperatura, construído anteriormente, para converter de Celsius para Fahrenheit. A fórmula de conversão é: $f = c \bullet 9/5 + 32$

4) Escreva um programa que receba um tempo em segundos (ex: 32084700 s), calcule e exiba a esse valor convertido em dias, horas e minutos.

5) Escreva um programa que peça as 4 notas de atividades contínuas e mostre a média. Lembrando que o total de atividades contínuas consideradas para o cálculo da média será um total de sete.

6) Escreva um programa que converta metros para milímetros.

7) Escreva um programa que peça o raio de um círculo, calcule e mostre sua área.

8) Escreva um programa que calcule a área de um quadrado, em seguida mostre o dobro desta área para o usuário.

9) Escreva um programa que pergunte quanto você ganha por hora e o número de horas trabalhadas no mês. Calcule e mostre o total do seu salário no referido mês.

10) Escreva um programa que lê dois valores inteiros e exibe o resultado do primeiro número dividido pelo segundo.

11) Escreva um programa que lê dois valores em ponto flutuante e exibe o resultado do primeiro dividido pelo segundo.

12) Escreva um programa que calcule e apresente o volume de uma lata de óleo.

$$v = \pi . r^2 . altura$$

13) Escreva um programa que receba três números quaisquer e apresente: a) a soma dos quadrados dos três números;
b) o quadrado da soma dos três números.

14) Escreva um programa que receba o custo de um espetáculo teatral e o preço do convite desse espetáculo. Esse programa deve calcular e mostrar:

a) A quantidade de convites que devem ser vendidos para que pelo menos o custo do espetáculo seja alcançado.

b) A quantidade de convites que devem ser vendidos para que se tenha um lucro de 23%.

15) Dado o preço de um produto (inteiro), elabore um programa que calcule e apresente a menor quantidade de notas (de cada valor) necessárias para efetuar o pagamento da compra desse produto. Considere como valores das notas atuais: 1, 2, 5, 10, 20, 50, 100.

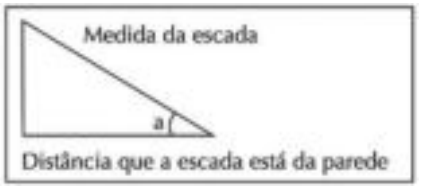
16) A fábrica de refrigerantes Meia-Cola vende seu produto em três formatos: lata de 350 ml, garrafa de 600 ml e garrafa de 2 litros. Se um comerciante compra uma determinada quantidade de cada formato, escreva um programa para calcular quantos litros de refrigerante ele comprou.

17) Um hotel deseja fazer uma promoção especial de final de semana, concedendo um desconto de 25% na diária. Sendo informados, através do teclado, o número de apartamentos do hotel e o valor da diária por apartamento para o final de semana completo, elabore um programa para calcular:

a) o valor promocional da diária;

b) o valor total a ser arrecadado caso a ocupação neste final de semana atinja 100%; c) o valor total a ser arrecadado caso a ocupação neste final de semana atinja 70%; d) o valor que o hotel deixará de arrecadar em virtude da promoção, caso a ocupação atinja 100%.

18) Escreva um programa que receba a medida do ângulo formado por uma escada apoiada no chão e a distância em que a escada está da parede, calcule e mostre a medida da escada.



19) Três amigos, Carlos, André e Felipe, decidiram rachar igualmente a conta em um bar. Escreva um programa para lei o valor total da conta e imprimir quanto cada um deve pagar, mas faça com que Carlos e André não paguem centavos. Por exemplo: uma conta de R$101,53 resulta em R$33,00 para Carlos, R$33,00 para André e R$ 35,53 para Felipe.

20) A padaria Hotpão vende uma quantidade de pães franceses e uma quantidade de broas a cada dia. Cada pãozinho custa R$ 0,12 e a broa custa R$ 1,50. Ao final do dia, o dono quer saber quanto arrecadou com a venda dos pães e broas (juntos), e quanto deve guardar numa conta de poupança (10% do total arrecadado). Você foi contratado para fazer os cálculos para o dono. Com base nestes fatos, escreva um programa para ler as quantidades de pães e de broas, e depois calcular os dados solicitados.

TAREFA MÍNIMA

EXERCÍCIO 1 - RESOLVIDO

Reescreva as seguintes expressões matemáticas utilizando operadores e funções vistas nessa aula: $x = \sqrt{a + p(p-a)(p-b)(p-c)_2}$

Solução

1) Vamos trocar √-por pela função sqrt () . Toda a expressão deve ficar dentro do parêntesis () da função sqrt(). OBS.: sqrt → **sq**uare **r**oo**t** → raiz quadrada

2) Na expressão (p-a)(p-b) devemos colocar o operador multiplicação dessa forma:

$$^{*} - b$$

$$(p-a)\ (p\ )$$

3) O exponencial $(p-c)$ pode ser trocado pelo operador ** ^{2}

$**$
$$(p-c)\ 2$$

A expressão final fica então:

$$^{*} - a^{*} - b^{*} - c^{**}$$
$$x = sqrt(\ a + p\ (p\ )\ (p\ )\ (p\ )\ 2\ )$$

EXERCÍCIO 2

Reescreva as seguintes expressões matemáticas utilizando os operadores e funções vistos nessa aula:

$$x = \sqrt{a + {}_{b_2}}$$
$$p-a$$

EXERCÍCIO 3 - RESOLVIDO

Escreva um programa que calcula e apresenta o dobro, o triplo e o quadrado de um número digitado pelo usuário.

Solução

Precisamos inicialmente entender o objetivo do exercício. Uma forma interessante de saber onde devemos chegar é identificar no enunciado do exercício o resultado final que está sendo solicitado (**saída**), os valores que devemos fornecer para conseguirmos fazer o cálculo (**entradas**) e

finalmente elaborar o cálculo necessário (**processamento**).

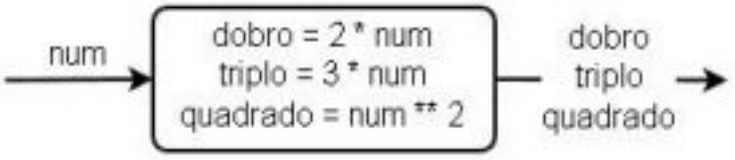
Podemos usar algumas palavras-chave para identificarmos a saída (exibir, apresentar, imprimir). Do texto podemos observar que temos 3 saídas (apresentar dobro, triplo e quadrado). Vamos guardar essas informações em variáveis e chamá-las de **dobro**, **triplo** e **quadrado**.

As entradas também podem ser identificadas por palavras-chave como (dado, digitado, informado, obtido). Nesse caso temos como entrada um único número digitado pelo usuário. Vamos atribuí-lo à variável **num**.

Agora precisamos identificar qual o cálculo que devemos fazer com a entrada (num) para obtermos as saídas pedidas (dobro, triplo e quadrado). Seguem os cálculos

- `dobro = 2*num` → dobro do número digitado pelo usuário
- `triplo = 3*num` → triplo do número digitado pelo usuário
- `quadrado = num**2` → quadrado do número digitado pelo usuário

Em seguida, podemos desenhar o seguinte diagrama de entrada/processamento/saída:



Escrevendo um código em uma estrutura sequencial para resolver o problemas, devemos primeiramente receber o dado de entrada, em seguida fazer os cálculos e finalmente exibir a saída:

```
# receber o dado de entrada
num = int(input('Digite um número: '))

# processamento dos dados, cálculos
dobro = 2 * num
triplo = 3 * num
quadrado = num ** 2

# Exibição do resultado
print('O dobro é:', dobro)
print('O triplo é:', triplo)
print('O quadrado é:', quadrado)
```

EXERCÍCIO 4

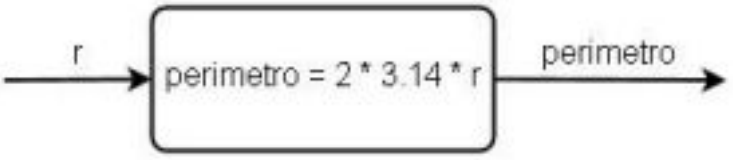
Escreva um programa em que o usuário digita um número e é apresentado o número consecutivo ao digitado (número seguinte).

EXERCÍCIO 5 - RESOLVIDO

Escreva um programa que calcule e apresente o perímetro de um aro de raio r. $perimetro = 2\pi r$

Solução

Podemos novamente montando uma estratégia de solução, utilizando como apoio o diagrama de entrada/processamento/saída:



Obs.: Perceba que o valor de π foi substituído por 3.14 (uma aproximação do valor real).

Com isso o código em Python fica:

```
# receber o dado de entrada
r = int(input('Digite o raio: '))

# processamento dos dados, cálculos
perimetro = 2 * 3.14 * r

# Exibição do resultado
print('O perimetro é:', perimetro)
```

EXERCÍCIO 6

Escreva um programa que calcule e apresente o volume de uma bola de futebol de raio r, digitado pelo usuário.

$$v = \frac{4}{3}\pi r^3$$