GERAÇÃODEQUADROS
Desenvolvendo Competências

Manual do Curso de Lógica de Programação

Duração: 20h

Luanda, 2021

Objectivos

Geral:

Promover um sólido conhecimento aos futuros profissionais da área de desenvolvimento de sistemas por meio de tecnologias abordadas durante o curso. O curso, em sua elaboração, valoriza a cientificidade na parte teórica quanto na prática, integrando assim, ciência e tecnologia, buscando soluções inteligentes para os problemas abordados.

Específico:

O objetivo deste módulo é apresentar os conceitos elementares da lógica e sua aplicação no quotidiano, estabelecendo uma relação entre a lógica e os algoritmos por meio de exemplos e exercícios práticos.

Conteúdo Programático

1. Introdução a Lógica de Programação
   2H
   - Noções de Lógica
   - Conceito de algoritmo
   - Algoritmos não convencionais / Programa
   - Linguagens de programação
   - Formas de representação de algoritmos
   - Regras para construção de algoritmos
   - Exercícios práticos

2. Variáveis 2H
   - Conceito e utilidade de variáveis
   - Tipos de dados
   - Tipos de operadores
   - Variáveis de entrada e saída
   - Exercícios práticos

3. Estruturas de controlo 4H
   - Estruturas de decisão
   - Estruturas de repetição
   - Exercícios práticos

4. Matrizes 2H
   - Arrays unidimensionais (vectores)
   - Arrays multidimensionais
   - Exercícios práticos

5. Modularização(funções) 4H
   - Parâmetros
   - Retorno

6. Considerações finais
   4H
   - Dúvidas
   - Discussão de Resultados

1. Introdução a Lógica de Programação

A lógica pode ser relacionado com a correção, pois uma das suas preocupações é determinar quais operações são válidas e quais não, fazendo análise das formas e leis do pensamento.

Ela é a arte de bem pensar, que é a ciência das formas de pensamento. Visto que a forma mais complexa do pensamento é o **raciocínio**, a lógica estuda a correção do raciocínio.

A lógica tem em vista a ordem razão, isso dá a entender que a nossa razão pode funcionar desordernamente. Por isso, a lógica estuda e ensina a colocar ordem no pensamento.

Significa o uso correto das leis do pensamento, da ordem de razão e do processamento de raciocínio e simbolização formais na programação de computadores, com objectivo de racionalização e o desenvolvimento de técnicas que cooperem para a produção de soluções logicamente válidas e coerentes, que resolvam com qualidade os problemas que se deseja programar.

Para representar fielmente a lógica de programação utilizamos os algoritmos.

Conceitos de algoritmos

- Algoritmo é um conjunto finito de regras, bem definidas, para a solução de um problema em um tempo finito e com um número finito de passos.

- Serve como modelo para programas, pois sua linguagem é intermediaria a linguagem humana e as linguagens de programação, sendo então, uma boa ferramenta na validação da lógica de tarefas a serem automatizadas.

Algoritmos não computacional / Programa

Abaixo é representado um Algoritmo não computacional cujo objectivo é trocar uma lampada.

1. Pegar uma escada.
2. Posicionar embaixo da lâmpada.
3. Pegar a lâmpada nova.
4. Subir na escada.
5. Retirar a lâmpada velha.
6. Colocar a lâmpada nova.
7. Descer da escada.

Programa é um algoritmo escrito em uma linguagem computacional.

Linguagens de Programação

São software que permitem o desenvolvimento de programas. Possuem um poder ilimitado desde jogos, editores de texto, sistemas empresariais até sistemas operacionais.

Existem várias linguagens de programação, cada um com suas características próprias. Ou seja cada uma com sua sintaxe e sua semântica.

Sintaxe: define como os comandos devem ser escritos.

Semântica: é o significado de cada comendo.

Exemplos:

- JavasScript, PHP, Python;
- Delphi, Visual Basic, Assemble;
- C, C++, C#, ABC, R, Dart.

Técnicas Atuais de Programação

- Programação Sequencial ;
- Programação Estruturada;
- Programação Orientada a Objetos.

Formas de representar algoritmos

Voltando para os algoritmo há diversas formas de representação. Essas formas diferem entre si pela quantidade de detalhes de implementação que fornecem ou, pelo grau de abstração com relação a implementação do algoritmo em termos de linguagem de programação específica.

Dentre as principais formas de representação de algoritmos destacam-se:

a) Descrição narrativa
b) Fluxograma convencional;
c) Pseudocódigo (ou linguagem estruturada).

Descrição narrativa

Nesta forma de representação os algoritmos são expressos diretamente em linguagem natural.

A vantagem nesta forma é que existe uma certa liberdade ao especificar as instruções dos algoritmos, cumprindo apenas algumas regras verbais e de sequência.

A desvantagem é a ambiguidade quanto a interpretação dos algoritmos, dependendo dos idiomas em que foram escritos.

Exemplo 1: Algoritmo “Fazer um Bolo”
Exemplo 2: Algoritmo “Tomar banho”
Exemplo 3: Algoritmo “Calcular a média do aluno”

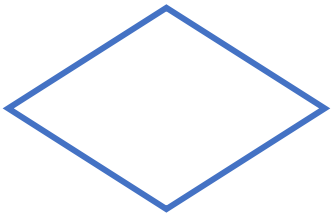
Fluxograma convencional

É uma representação gráfica de algoritmos onde formas geométricas diferentes implicam ações (instruções comandos) distintas.

Tal propriedade facilita o entendimento das ideias contidas nos algoritmos e justifica sua popularidade.

Esta forma é aproximadamente intermediária a descrição narrativa e ao pseudocódigo pois é menos imprecisa que a primeira e, no entanto, não se preocupa com detalhes de implementação do programa.

Símbolos

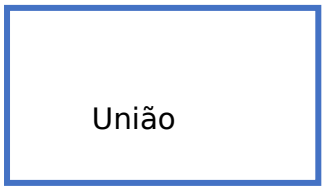


Inicio e Fim de programa

Decisão

Operação de Saída

Operação de atribuição



Fluxo

Operaçao de entrada de dados

Nota: Nesse curso não iremos usar os Pseudocodigo(linguagem estruturada). Mas sim os fluxograma convencional com uma linguagem de programação escolhida com pela nossa equipa.

Regras para construção de algoritmos

1- Usar somente um verbo por frase;

2- Imaginar que está a desenvolver um algoritmo para pessoas que não trabalham com informática;
3- Usar frases curtas e simples;
4- Ser objetivo;
5- Procurar usar palavras que não causam ambiguidade(confusão).

2.Variáveis

As variáveis são utilizadas para armazenar os dados que um programa deve manipular. Toda variável possui um nome (identificador).

Para acessar ou alterar o conteúdo de uma variável, é necessário utilizar o nome dessa variável.

Exemplo: Idade, altura, nome, peso etc...

Tipos de Dados

Existem 3 (três) tipos de **dados primitivos**, outros são tipos de **dados abstratos**, pois são resultado da combinação de tipos primitivos.

- Dados numéricos;
  - Inteiros;
  - Reais.
- Cadeia de caracteres ou literal;
  - String
- Dados lógicos/boolianos ;
  - Verdadeiro
  - Falso

Dados numéricos

Inteiros: são aqueles que não possuem componentes decimais ou fraccionários, podendo ser positivo ou negativo.

Exemplos

15 – número inteiro positivo.
-20 – número inteiro negativo.
0 – número inteiro neutro.

Reais: são aqueles que podem possuir componentes decimais ou fraccionários, podendo ser positivo ou negativo.

Exemplos**:**

15.75 – número real positivo com duas casa decimais.
- 20.4 – número real negativo com uma casa decimal.
0.0 – número real neutro com uma casa decimal.

Cadeia de caracteres

É constituído por uma sequência de caracteres contendo letras, dígitos e/ou símbolos especiais. Também chamado alfanumérico, do inglês **String**.

São delimtadas em seu início e fim com o caracter aspas (“”) ou (‘’) e também possui um comprimento dado pelo número de caracteres nele contido.

Exempos:

“QUAL ?” – cadeia de caracteres de comprimento 6.
“ ” – cadeia de caracteres de comprimento 1.
“qUal ?!$” – cadeia de caracteres de comprimento 8.
“1-2+4=4” – cadeia de caracteres de comprimento 8.
“0” – cadeia de caracteres de comprimento 1.

Tipo booleano

É usado para representar dois únicos valores possíveis: Verdadeiro(V) e Falso(F).

Nomenclatura de Variáveis

Para atribuímos um nome a uma variável, também podemos chamar de identificador da variável, devemos seguir as seguintes etapas:

1. O nome de uma variável deve realçar o valor que será armazenado nela;
2. O primeiro carácter do nome de uma variável deverá ser sempre um alfabeto(letra) ou Underscore(_);
3. O nome da variável não pode conter espaços;
4. O nome da variável não pode conter caracteres ou símbolos especiais, excepto Underscore(_), que representa **espaços**.

Exempos:

- SALARIO (correcto);
- Salario_Hora (correcto);
- 1 ANO (incorrecto);
- _Desconto (correcto);
- Sal/CASA (incorrecto).

Operadores

Operadores: são elementos operacionais que actuam sobre operandos e produzem um determinado resultado.

Tipos de Operadores

Operadores Aritméticos

+	Adição
-	Subtração
*	Multiplicação
/	Divisão
%	Resto da divisão
**	Expoente

Operadores Relacionais	
>	Maior que
<	Menor que
>=	Maior ou igual
<=	Menor ou igual
==	Igual
!=	Diferente

Operadores Lógicos		
And	&&	E
Or	||	Ou
Not	!	Não

Tabela da Verdade

A	B	A and B	A or B	Not A	Not B
V	V	V	V	F	F
V	F	F	V	F	V
F	V	F	V	V	F
F	F	F	F	V	V

Hierarquia das Operadores

1. () Parênteses

2. Operadores Aritméticos
   2.1. Multiplicação ou Divisão (o que vier primeiro)
   2.2. Soma ou Subtração (o que vier primeiro
3. Operadores Relacionais
4. Operadores Booleanos

Atribuição

Para atribuirmos um valor as variáveis ou constantes usamos o sinal de igualdade (=).

Exemplo:

```
nome = ‘Geração de Quadros’
idade = 10
```

Variáveis de Entrada e Saida

O computador não é uma máquina isolada, pois ele precisa se comunicar com o mundo exterior através do vídeo, impressora, teclado, discos, etc.

Para realizar esta comunicação existem comandos que permitem que informações sejam exibidas, como também existem comandos que permitem que informações sejam colocadas na memória (variáveis) do computador através do teclado do PC. Com esta finalidade, utilizaremos os comandos de entrada e saída.

Cada linguagem de programção possui os seus comandos de entrada e saída, em Python iremos usar os seguintes comandos:

print (para imprimir valor na tela)
input (para receber e guardar valores)

Nota: A partir desde ponto em diante todos os exemplos serão apenas feitos em Python.

Exemplo 1: Programa para ler um nome e imprimir na tela.

```
nome = input(‘Insira um nome: ’)
print(nome)
```

Exemplo 2: Programa para ler um número e imprimir na tela.

```
numero = input(‘Escreve um numero: ’)
print(‘Seu nome é ’, numero)
```

Constantes

São endereços de memória destinados a armazenar informações fixas, inalteráveis durante a execução do programa.

Exemplo:

```
PI = 3.1416
```

Comentários

São notas que podem ser incluídas no código fonte para descrever o que se quiser. Não modificam o programa executado e servem somente para ajudar o programador a melhor organizar os seus códigos.

Abaixo temos as formas de comentar um código em algumas linguagens.

	Comentários de linha	Comentário de blobo
Python	# Meu comentário 1	‘’’ Meu comentário 2 ‘’’
PHP, JS, C	// Meu comentário 3	/* Meu comentário 4 */

Conversão de tipos

Se as informações possuem tipos, logo, temos de ser capazes em converter um tipo de informação num outro tipo de dado. Essa ação de conversão é comummente chamada de Coerção de Tipos.

Para convertermos, por exemplo, um texto para o tipo numérico, devemos especificar o tipo a ser convertido e passarmos o valor através de parêntesis, como podemos ver a seguir:

tipo_a_ser_convertido(informação)

Em python nos temos pelo menos 3 tipos de dados para conversão:

- str() # Conversão para string(literal)
- float() # Conversão para número decimal
- int() # Conversão para número inteiro

Exemplo: Conversão do tipo string para inteiro.

```
texto = "10"
num = int(texto)
```

Em python para saber de que tipo é uma variável usados a função **type()**

Exemplo:

```
texto = "10"
print(type(texto)) # Saída:  <class ‘str’>

num = int(texto)
print(type(num)) # Saída: <class ‘int’>
```

Concatenação

Bla bla bla bla ... ...

```
print( texto + ‘10’ )
# Saída: 1010
# O sinal de + concatena duas informações

print( num + 10 )
#Saída: 20
# O sinal de + soma dois números
```

3. Estruturas de controlo

Como pode ser analisado no tópico anterior, todo programa possui uma estrutura sequencial determinada por um ÍNICIO e FIM. Dentres essas estruturas temos duas:

- Estruturas de decisão
- Estruturas de repetição

Estruturas de decisão

Executa uma seqüência de comandos de acordo com o resultado de um teste lógico. A estrutura de decisão pode ser Simples ou Composta. Abaixo apresentamos um exemplo em python.

Simples	**Composta 1**	**Composta 2:**

if condição:
...comando
	if condição:
	...comando
	else:
	...comando
		if condição:
		...comando
		elif condição:
		...comando
		else:
		...comando

```
if 1 == 1:
    print(‘Ola’)
```

```
if num > 12:
   print(‘Patrício’)
else:
   print(‘Geração´)
```

```
if media < 7:
    print(‘Reprovado)
elif media < 10:
    print(‘Recurso’)
else:
    print(‘Aprovado’)
```

Estruturas de repetição

É quando uma sequência de comandos deve ser executada repetidas vezes.

A estrutura de repetição, assim como a de decisão, envolve sempre a avaliação de uma condição.

Quando quando sabemos previamente o número de vezes a repetir usa-se a **estrutura de repetição determinada (for).**

```
Forma geral:

        for contador in range(1, 10):

            ...comando
```

```
for contador in range(1, 10):

        print(contador)

# Saída: 1 2 3 4 5 6 7 8 9
```

E quando não sabemos preveiamente o número de vezes a repetir usa-se a **estrutura de repetição indeterminada (while).**

```
Forma geral:

        contador = 1
        while contador < 10:
            ...comando
           contador = contador + 1
```

```
contador = 1
while contador < 10:
    print(contador)
    contador = contador + 1

#Saída: 1 2 3 4 5 6 7 8 9
```

4. Matrizes

São estruturas de dados muito simples que podem nos ajudar muito quando temos muitas variáveis do mesmo tipo em um algoritmo.

Imagine o seguinte problema: Você precisa criar um algoritmo que lê o nome e as 4 notas de 50 alunos, calcular a média de cada aluno e informar quais foram aprovados e quais foram reprovados.

Conseguiu imaginar quantas variáveis você vai precisar? Muitas né? Vamos fazer uma conta rápida: 50 variáveis para armazenar os nomes dos alunos, (4 * 50 =) 200 variáveis para armazenar as 4 notas de cada aluno e por fim, 50 variáveis para armazenar as médias de cada aluno. 300 variáveis no total, sem contar a quantidade de linhas de código que você vai precisar para ler todos os dados do usuário, calcular as médias e apresentar os resultados. Mas eu tenho uma boa notícia para você. Nós não precisamos criar 300 variáveis! Podemos utilizar Vetores e Matrizes (também conhecidos como ARRAYs).

Que podem ser de dois tipos:

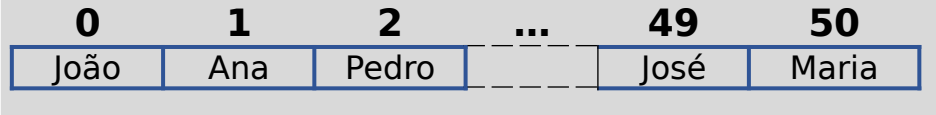
- Arrays unidimensionais
- Arrays multidimensional

Arrays unidimensionais (vectores)

É uma variável que armazena várias variáveis do mesmo tipo. No problema apresentado anteriormente, nós podemos utilizar um vector de 50 posições para armazenar os nomes dos 50 alunos.

Exemplo graficamente:

Vetor de nomes dos alunos



Exemplo em python:

nome_do_array = [elementros_separados_por_virgula]

```
alunos = [‘João’, ‘Ana’, ‘Pedro’, ‘José’, ‘Maria’]
  alunos = array(‘João’, ‘Ana’, ‘Pedro’, ‘José’, ‘Maria’)
print(alunos[0])  # Saída: João
```

Arrays multidimensional

É um vetor de vetores. No nosso problema, imagine uma matriz para armazenar as 4 notas de cada um dos 50 alunos. Ou seja, um vetor de 50 posições, e em cada posição do vetor, há outro vetor com 4 posições. Isso é uma matriz.

Cada item do array é acessado por um número chamado de índice.

Exemplo graficamente:

Matriz das notas de alunos

	0	1	2	3
0	10	14	16	12
1	8.5	12	7.6	13
2	18	12	15	19
49	11	14	15	16
50	12	5.5	8.5	10

Exemplo em python:

nome_do_array = [vector_1, vector_1]

```
notas_alunos = [
      [10, 14, 16, 12],
      [8.5, 12, 7.6, 13],
      [18, 12, 15, 19],
      [11, 14, 15, 16],
      [12, 5.5, 8.5, 10]
 ]

print(notas_alunos[0])       # Saída: [10, 14, 16, 12]
print(notas_alunos[1][3])    # Saída: 7.6
```

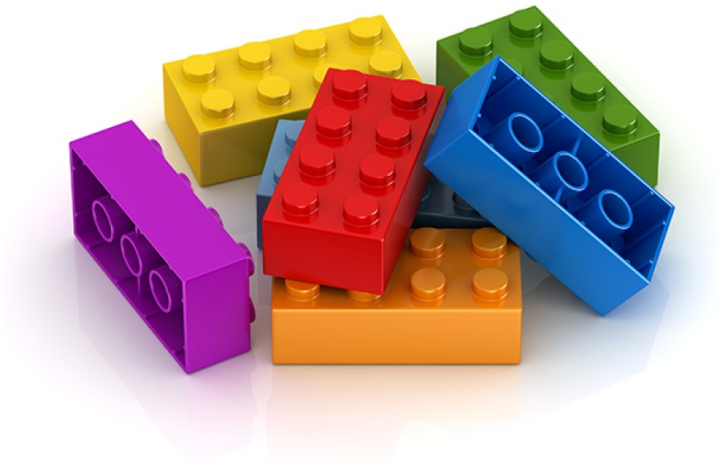
5. Modularização (Funções)

Até o momento estamos programando soluções pequenas. Cada exercício se baseava em um problema específico que conseguimos resolver com poucas linhas de código, mas em aplicações reais é comum um único programa resolver diversas coisas diferentes.

A criação de soluções mais complexas e programas maiores se torna uma tarefa bastante difícil se continuarmos programando da maneira que estamos. Precisamos fazer com que o programa seja separado em peças menores.

Dessa forma, todas as vezes que precisarmos resolver um problema mais complexo, podemos conectar essas peças para formar uma solução.

Para isso, precisamos que estas peças sejam reutilizáveis, assim como peças de lego.



Estas peças reutilizáveis são chamadas de funções. Elas são formadas por um bloco de código que executa uma determinada ação.

Na verdade, durante os exercícios anteriores, já utilizamos várias vezes funções disponíveis no Python! O print e o input são exemplos de funções disponíveis, mas o mais interessante disso é que podemos definir nossas próprias funções!

De modo geral, cada função deve realizar apenas uma ação. Desta forma ela se torna mais reutilizável e útil para seus programas.

Sempre que você tiver uma função que realiza mais de uma tarefa fique atento, isso provavelmente é um sinal de código ruim. Talvez esta função deva ser quebrada em duas ou mais funções.

Vamos ver como podemos definir uma função em nosso código Python!

```
def ola_mundo ():
    print(“Ola
```

Primeiro é necessário utilizar a palavra reservada def, esta palavra é responsável por definir a função. Depois disso incluímos o nome da função, esse nome deve seguir as mesmas regras para nome de variáveis.

Por fim, estamos incluindo os caracteres **()**, a partir daqui utilizamos os **:** e todos os comandos pertencentes à este bloco segue as mesmas regras dos comandos já vistos: if, while, for e etc.

Mas há uma diferença importante, ao definirmos uma função não fazemos com que aqueles comandos sejam executados, estamos apenas descrevendo o que irá acontecer quando executarmos a função.

Para executar uma função você precisa utilizar o nome dela seguido dos parênteses, veja:

```
def ola_mundo ():          # Definição
    print(“Ola mundo”)

ola_mundo ()                        # Chamada
```

Você pode colocar dentro de uma função quantos comandos desejar e todos eles serão executados assim que a função for chamada.

Parâmetros

É comum precisarmos de dados para realizar algum tipo de operação. Veja a função print por exemplo, a tarefa desta função é escrever qualquer coisa no console certo.

Para permitir que isso seja possível, a função print precisa receber um parâmetro. No caso da função print o parâmetro é o texto que será escrito, mas os parâmetros podem ser utilizados para tarefas diferentes.

Vamos criar uma função que eleve qualquer número ao quadrado!

```
def elevar_ao_quadrado (numero):
    print(numero * numero)

elevar_ao_quadrado(3)  # Saida: 9
```

Uma função também pode receber vários parâmetros. Vamos ver uma função que soma dois números.

```
def soma (numero1, numero2):
    print(numero1 + numero2)

soma (2, 5) # Saida: 7
soma (3, 3) # Saída: 6
```

Retornos

Além de receber informações (parâmetros) as funções também podem produzir resultados. Esses resultados são chamados de retornos.

Nem todas as funções produzem resultados, veja a função ola_mundo criada anteriormente. De fato ela não precisa produzir nenhum resultado.

Por outro lado, não faz muito sentido a função soma apenas escrever o resultado da soma. A função se tornaria muito mais reutilizável produzindo um resultado. Desta forma o desenvolvedor poderia fazer o que bem entendesse com o resultado, inclusive escrevê-lo.

Sempre que precisamos criar um retorno utilizamos a palavra reservada **return**. Ela faz com que a função produza um resultado, assim como uma expressão.

Vamos refactorar a função soma:

```
def soma (numero1, numero2):
    return numero1 + numero2

resultado = soma (1, 2)
print (resultado)   # Saída: 3

resultado = soma (2, 2) * 3
print (resultado)   # Saída: 12

resultado = soma (5, 5) – 2 * 5
print (resultado)   # Saída: 10

print (soma (3, 7))        # Saída: 10
```

Para entender melhor basta pensar que a função passa a se comportar como uma expressão, portanto conseguimos realizar atribuições, somas e qualquer outra operação que poderíamos fazer com uma expressão.

EXERCÍCIOS DAS AULAS

AULA 1 – INTRODUÇÃO A LÓGICA

1- Crie uma sequência lógica para tomar banho.

- Era uma vez a história de quatro homens: João, José, Jacinto e Joel. Os quatro eram construtores de barcos e em quatro dias conseguiam construir quatro embarcações. Quanto tempo demoraria um dos quatro homens para construir um único barco?

- A senhora Adelaide tem um galinheiro muito grande e agora pondera começar a vender os ovos na aldeia onde vive. Precisa então de fazer contas à vida e quer saber quantos ovos terá para vender. Ela sabe que uma galinha e meia põem um ovo e meio num dia e meio. Quantos ovos deverão por sete galinhas em seis dias?

- Olá! Eu sei que não sabe quem sou, mas nós somos da mesma família: o meu pai é irmão da sua irmã. Consegue adivinhar que parente sou eu: primo, sobrinho, filho, tio ou genro?

- Na escola primária da cidade, a professora Alice tem um quebra-cabeças para os meninos do terceiro ano. Para fazer uma pausa das contas de multiplicação, ela distribuiu uma lista com seis palavras: Agate, Agitate, Gates, Stags, Stage, Grate. Depois lançou-lhes um desafio: quais destas palavras são compostas pelas mesmas letras?

- Toda a gente reparou nas trocas de olhares daquele triângulo amoroso no meio do bar. O Tomás não parava de olhar para a Catarina; e a Catarina não parava de olhar para o Rui. Ora, acontece que Tomás era casado, mas o Rui não. Descubra se há alguma pessoa casada a olhar para uma pessoa não casada neste triângulo amoroso.

2- Crie uma sequência lógica da parte favorita da tua rotina.

3- Descreva a sequência lógica para trocar o pneu de um carro.

4- Faça um algoritmo para trocar o sistema operativo Windows para o sistema operativo Linux (Ubuntu).

5- Descreva a sequência lógica que usaste para sair da tua casa até à Znat Tech.

6- Faça uma sequência para lógica para terminar um namoro.

AULA 2 – VARIÁVEIS

1- Faça um programa imprime a mensagem "Ola mundo!" na tela.

Exemplo de Entrada	Exemplo de Saída
	Ola mundo!
Patrício	**Ola, Patrício!**

2- Faça um programa que leia 2 valores inteiros e armazene-os nas variáveis numero_1 e numero_2. Efetue a soma atribuindo o seu resultado na variável soma, e imprima na tela o valor de soma.

Exemplo de Entrada	Exemplo de Saída
10 e 9	19
-10 e 4	-6
15 e -7	8

3- Faça um programa para calcular a área de uma circunferência.
Considerando: area = π . raio^2 e π = 3.14159.
Efetue o cálculo da área, eleva o valor do raio ao quadrado e multiplica por π.

Exemplo de Entrada	Exemplo de Saída
2.00	12.5664

100.64	31819.3103
150.00	70685.7750

4- Faça um programa que leia dois valores inteiros. A seguir, calcule o produto entre estes dois valores e atribua esta operação à variável PROD. A seguir mostre a variável PROD com uma mensagem a tua escolha.

Exemplo de Entrada	Exemplo de Saída
3 e 7	21
-30 e 20	-300
0 e 9	0

5- Faça um programa que leia 2 valores de ponto flutuante de uma casa decimal A e B, que correspondem a duas notas de um aluno. A seguir, calcule a média do aluno, com duas casas decimais, sabendo que a nota A tem peso 3.5 e a nota B tem peso 7.5

Exemplo de Entrada	Exemplo de Saída
5 e 7.1	6.43
0 e 7.1	4.81
10 e 10	10.00

6- Faça um programa que leia 3 valores, no caso, variáveis n1, n2 e n3, que são as três notas de um aluno. A seguir, calcule a média do aluno, sabendo que a nota n1 tem peso 2, a nota n2 tem peso 3 e a nota n3 tem peso 5. Considere que cada nota pode ir de 0 até 10.0, sempre com uma casa decimal.

Exemplo de Entrada	Exemplo de Saída
5, 6, 7	6.3
5, 10 e 10	9.0
10, 10 e 5	7.5

7- Faça um programa que leia quatro valores inteiros A, B, C e D. A seguir, calcule e mostre a diferença do produto de A e B pelo produto de C e D.

Exemplo de Entrada	Exemplo de Saída
5, 6, 7 e 8	-26
0, 0, 7 e 8	-56
5, 6, -7 e 8	86

8- Escreva um programa que leia o número de um funcionário, seu número de horas trabalhadas, o valor que recebe por hora e calcula o salário desse

funcionário. A seguir, mostre o número e o salário do funcionário, com duas casas decimais.

Exemplo de Entrada	Exemplo de Saída
25A, 100, e 5.50	Código: 25A - Salário: 550.00Kz
01A, 200 e 20.50	Código: 01A - Salário: 4100.00Kz
06D, 145, e 15.55	Código: 06D - Salário: 254.75

9- Faça um programa que leia o nome de um vendedor, o seu salário fixo e o total de vendas efetuadas por ele no mês (em dinheiro). Sabendo que este vendedor ganha 15% de comissão sobre suas vendas efetuadas, informar o total a receber no final do mês, com duas casas decimais.

Exemplo de Entrada	Exemplo de Saída
Júlio, 500 e 1230.30	Salário (+15%): 684.54Kz
Pedro, 700 e 0.00	Salário (+15%): 700.00Kz
Vivaldo, 1700 e 1230.50	Salário (+15%): 1884.58Kz

10- Neste problema, deve-se ler o nome e o código de uma peça 1, o número de peças 1, o valor unitário de cada peça 1, o nome e o código de uma peça 2, o número de peças 2 e o valor unitário de cada peça 2. Após, calcule e mostre o valor a ser pago.

Exemplo de Entrada	Exemplo de Saída
Mouse_02c, 1 e 5.30 Teclado_1n3, 2 e 5.10	Total: 15.50Kz
Cadeira_33f, 2 e 15.30 Desktop_2x10, 1 e 15.10	Total: 30.20Kz

⇨ Aula 3 – Estruturas de Controlo

- Faça um Algoritmo que leia um número e informa se esse número é par ou ímpar.

- Calcular a média final e a situação de um aluno seguindo as seguintes condições: 3 provas e obrigação de no mínimo 75% de freqüência. Serão fornecidos pelo usuário as notas das três provas e o percentual de freqüência de cada aluno. Com estas informações fornecidas apresentar a média final do aluno e a sua situação – aprovado ou reprovado. Média para aprovação maior ou igual a 7.0.

- Indicar a classificação do peso das pessoas de acordo com os padrões da OMS. Para obter este valores da classificação a operação que deve-se realizar é peso dividido pela altura ao quadrado. O valor obtido desta operação deve ser aplicado a tabela apresentada acima. Apresentar a situação de cada pessoa que está sendo consultada. (0,10] = Desnutrido ; (10,18] = Abaixo do peso; (18,26]= Normal; (26, 30] = Acima do Peso; (30,...) = Obeso

- Escreva o comando de seleção para cada uma das situações a seguir: Se x for maior que y ou se z for menor ou igual a 30, multiplique x por 2. Caso contrário, divida x por 2 e divida z por 5.
- Um vendedor tem seu salário calculado em função do valor total de suas vendas. Esse calculo é feito de acordo com o seguinte critério: se o valor total de suas vendas for maior que 20000,00 KZ, o vendedor receberá como salário 10% do valor das vendas. Caso contrario, receberá apenas 7,5% do valor das vendas. Escrever um algoritmo que determine o valor ganho pelo vendedor.
- Elabore um algoritmo que leia as variáveis C e N, respectivamente código e número de horas trabalhadas de um operário. E calcule o salário sabendo-se que ele ganha 10,00 KZ por hora. Quando o número de horas exceder a 50 calcule o excesso de pagamento armazenando-o na variável E, caso contrário zerar tal variável. A hora excedente de trabalho vale 20,00 KZ. No final do processamento imprimir o salário total e o salário excedente.
- Escrever um algoritmo que leia uma quantidade desconhecida de números e conte quantos deles estão nos seguintes intervalos: [0-25], [26-50], [51-75] e [76-100]. A entrada de dados deve terminar quando for lido um número negativo.
- Desenvolver um algoritmo que efetue a soma de todos os números ímpares que são múltiplos de três e que se encontram no conjunto dos números de 1 até 500.
- 2) Desenvolver um algoritmo que leia a altura de 15 pessoas. Este programa deverá calcular e mostrar :

  1. A menor altura do grupo;
  2. A maior altura do grupo;

⇨ Aula 4 – Matrizes

Elabore um subprograma C que tenha como parâmetros de entrada duas matrizes de números reais e forneça como resposta o produto das mesmas. Caso não seja possível efetuar a multiplicação, o subprograma deve retornar um código de erro. Caso as dimensões não permitam que se efetue a multiplicação o subprograma deve retornar o código de erro diferente de zero.

Elabore um subprograma python que tenha como parâmetros de entrada um vetor (cujos elementos são do tipo real) e o número de elementos do vetor, e forneça como saída o menor elemento do vetor.

Em uma cidade do interior, sabe-se que de janeiro a abril de 1976 (121 dias) não ocorreu temperatura inferior a 15ºC e nem superior a 40ºC. Faça um programa em python que determine:

- A menor temperatura ocorrida
- A maior temperatura ocorrida
- A temperatura média do período.

https://gabrielschade.github.io/2018/04/23/basics-python-8-functions.html