GRAN CONCURSOS

CONHECIMENTOS ESPECÍFICOS

Linguagem de Programação Python

Presidente: Gabriel Granjeiro

Vice-Presidente: Rodrigo Calado

Diretor Pedagógico: Erico Teixeira

Diretora de Produção Educacional: Vivian Higashi

Gerência de Produção de Conteúdo: Magno Coimbra

Coordenadora Pedagógica: Élica Lopes

CÓDIGO:

230227375889

PATRÍCIA QUINTÃO

Mestre em Engenharia de Sistemas e computação pela COPPE/UFRJ, Especialista em Gerência de Informática e Bacharel em Informática pela UFV. Atualmente é professora no Gran Cursos Online; Analista Legislativo (Área de Governança de TI), na Assembleia Legislativa de MG; Escritora e Personal & Professional Coach.

Atua como professora de Cursinhos e Faculdades, na área de Tecnologia da Informação, desde 2008. É membro: da Sociedade Brasileira de Coaching, do PMI, da ISACA, da Comissão de Estudo de Técnicas de Segurança (CE-21:027.00) da ABNT, responsável pela elaboração das normas brasileiras sobre gestão da Segurança da Informação.

Autora dos livros: Informática FCC - Questões comentadas e organizadas por assunto, 3ª. edição e 1001 questões comentadas de informática (Cespe/UnB), 2ª. edição, pela Editora Gen/Método.

Foi aprovada nos seguintes concursos: Analista Legislativo, na especialidade de Administração de Rede, na Assembleia Legislativa do Estado de MG; Professora titular do Departamento de Ciência da Computação do Instituto Federal de Educação, Ciência e Tecnologia; Professora substituta do DCC da UFJF; Analista de TI/Suporte, PRODABEL; Analista do Ministério Público MG; Analista de Sistemas, DATAPREV, Segurança da Informação; Analista de Sistemas, INFRAERO; Analista - TIC, PRODEMGE; Analista de Sistemas, Prefeitura de Juiz de Fora; Analista de Sistemas, SERPRO; Analista Judiciário (Informática), TRF 2ª Região RJ/ES, etc.

Redes Sociais: @coachpatriciaquintao (Instagram) /profapatriciaquintao (YouTube) / @plquintao (Twitter) / t.me/coachpatriciaquintao (Telegram)

SUMÁRIO

APRESENTAÇÃO

Saudações, querido(a) amigo(a)!

É sempre um prazer fazer parte dessa trajetória de muito sucesso com todos vocês!

Vamos então à aula sobre **Python**. Também acrescentei alguns comentários importantes sobre **lógica de programação**, de grande valia para a sua prova.

Pense POSITIVO E FORÇA nos estudos!

Grande abraço,

LINGUAGEM DE PROGRAMAÇÃO PYTHON

1. FUNDAMENTOS DE LÓGICA DE PROGRAMAÇÃO

- **Lógica de programação** é a técnica de encadear pensamentos para atingir determinado **objetivo**, possibilitando a resolução de problemas e, consequentemente, automatização de tarefas.
- Esses pensamentos podem ser descritos como uma sequência de instruções, que devem ser seguidas para se cumprir uma determinada tarefa.
- **Sequência lógica** são passos executados até atingir um objetivo ou solução de um **problema**.

1.1. INSTRUÇÕES

- **Instruções** são um conjunto de regras ou normas definidas para a realização ou emprego de algo. Em informática, é o que indica a um computador uma **ação elementar a executar**.
- Convém ressaltar que **uma ordem isolada não permite realizar o processo completo**, para isso é necessário um conjunto de instruções colocadas em ordem sequencial lógica. Mas, **até mesmo as coisas mais simples, podem ser descritas por sequências lógicas**.

EXEMPLO

Para fazer um bolo segue-se uma **sequência lógica de passos**:
bater ovos com açúcar;
acrescentar farinha de trigo e leite;
colocar fermento em pó e levar para assar.
Você tem que seguir essa sequência da forma que está, senão o bolo não fica bom.

- Assim, uma instrução tomada em separado não tem muito sentido; para obtermos o resultado, precisamos colocar em prática o conjunto de todas as instruções, na ordem correta.

1.2. ALGORITMOS

- Um **algoritmo** é uma sequência **finita** de **passos** (ou **instruções**), descritos em uma **ordem lógica**, que **visam atingir um objetivo bem definido**.

Obs.: Dá-se o nome de **algoritmo** a uma sequência lógica finita de instruções cujo objetivo é a execução de uma determinada tarefa.

- As **instruções devem ser bem definidas e não ambíguas**, bem como o tempo e o esforço para executar as instruções devem ser **finitos**.
- O **problema solucionado** por **algoritmo** deve ser: claro e bem definido.

EXEMPLO
Como **exemplos** tem-se os **algoritmos das operações básicas** (adição, multiplicação, divisão e subtração) **de números reais decimais**. Outros exemplos seriam os manuais de aparelhos eletrônicos, como um videocassete, que explicam passo-a-passo como, por exemplo, gravar um evento.

- Um algoritmo tem **cinco características importantes**:

Finitude	O algoritmo **deve** sempre **terminar após um número finito de passos**.
Definição	Cada passo de um algoritmo deve ser precisamente definido. **As ações devem ser definidas rigorosamente e sem ambiguidades**.
Entradas	O algoritmo **deve ter zero ou mais entradas**, quantidades que lhe são fornecidas antes do algoritmo iniciar.
Saídas	**Pelo menos um valor é produzido**.
Efetividade	**Cada passo/instrução/etapa** de um algoritmo **deve ser executável**.

- **Um algoritmo opera sobre um conjunto de** entradas **de modo a gerar uma** saída **que seja útil para o usuário.**
- A formulação de um **algoritmo** geralmente consiste em um texto contendo **comandos** (**instruções**), que devem ser executados numa sequência prescrita (receita).

1.3. LINGUAGENS DE PROGRAMAÇÃO

- As **instruções** dadas ao computador possuem **regras** e uma **sintaxe** própria, como uma linguagem tipo português ou inglês.
- Infelizmente, um computador só é capaz de seguir programas que estejam escritos em **linguagem de máquina**, que normalmente é obscura e desconfortável.
- **A linguagem de máquina** é a **linguagem natural do computador**, definida pelo seu projeto de hardware.
  - As instruções do programa, escritas em linguagem de máquina, consistem em uma série de dígitos binários.
  - Como estão mais próximas da linguagem do computador, são muito complexas para o entendimento humano.
- Os seres humanos, entretanto, acham mais conveniente escrever os programas em linguagem de nível mais elevado, como o Pascal por exemplo.
- As **linguagens de alto nível** são linguagens que otimizam o processo de programação por utilizar instruções mais parecidas com a linguagem humana (inglês cotidiano) e notações matemáticas comuns. **Exemplo de linguagens de alto nível: C, C++,.NET, Visual Basic, Pascal e Java.**

Obs.: É interessante notar que, **quanto mais próxima da linguagem humana (alto nível)** é uma linguagem de programação, **mais fácil e produtivo é o processo de desenvolvimento**, e mais lento é o processo de tradução das instruções.

Por outro lado, **quanto mais distante da linguagem humana (baixo nível)** é uma linguagem de programação, **mais rápido é o processo de tradução**, e mais lento é o processo de desenvolvimento de programas.

1.4. PROGRAMAS

- Os **programas de computadores** nada mais são do que **algoritmos escritos numa linguagem de computador** (Pascal, C, Visual Basic, etc.) e que são interpretados e executados por uma máquina, no caso um computador.
- Notem que dada essa interpretação rigorosa, um programa é por natureza muito específico e rígido, em relação aos algoritmos da vida real.
- Chama-se de **programas** **os algoritmos que são traduzidos para uma linguagem de computador** (por exemplo: Cobal, Fortran, Pascal, C, Java, etc.), cujos passos (chamados de **comandos**) são interpretados e executados pelo computador.
- **Um algoritmo NÃO representa, necessariamente, um programa de computador**, e sim os passos necessários para realizar uma tarefa ou solucionar um problema.

Obs.: Um algoritmo não é uma solução para um problema, mas um caminho para obtê-la. Sendo assim, **podem existir vários caminhos para uma mesma solução, e, consequentemente, vários algoritmos diferentes para o mesmo problema.**

1.5. FASES DE UM ALGORITMO

- Ao construir um algoritmo, precisamos primeiro dividir o problema apresentado em três fases fundamentais:
  - **Entrada**: sãos os **dados que serão processados pelo algoritmo**;
  - **Processamento**: representa **os procedimentos necessários para se obter o resultado final**; e
  - **Saída**: são os **dados gerados depois do processamento**, ou seja, os dados já processados!

1.6. REPRESENTAÇÃO DE ALGORITMOS

Os algoritmos podem ser representados de várias formas, como:

- **Descrição narrativa**
  - Expressos em uma linguagem natural, podendo dar margem a más interpretações, ambiguidades ou imprecisões;
  - Linguagem **mais informal**.

- **Fluxograma**
  - Expressos graficamente utilizando-se de formas geométricas padronizadas, cada uma com um significado próprio.

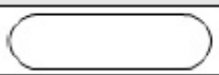
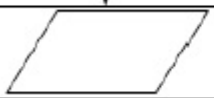
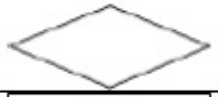
Símbolo	Descrição
	Início ou fim do algoritmo
	Indica o sentido do fluxo de execução do algoritmo. Conecta os objetos gráficos
	Representa a entrada de dados
	Indica cálculos e atribuições de valores (processamento)
	Indica desvios ou tomadas de decisões (Por exemplo: SE isso, ENTÃO aquilo)
	Representa a saída de dados

Figura. Símbolos utilizados em um fluxograma. Fonte: Araújo, 2012.

- **Linguagem algorítmica:**
  - Também conhecida por **pseudocódigo, portugol, pseudolinguagem** ou **Português Estruturado**.
  - Os algoritmos são expressos em uma **linguagem intermediária** entre a **linguagem natural** e uma **linguagem de programação,** que não segue um formalismo tão rígido quanto o das linguagens de programação.
  - **É a representação mais utilizada em concursos públicos e que utilizaremos nesta aula**!

1.7. VARIÁVEIS

- **Variáveis** são **endereços de memória** destinados a armazenar informações **TEMPORARIAMENTE**. Para que a programação se torne mais fácil, podemos colocar nomes legíveis nesses endereços de memória e utilizá-los no algoritmo.
- Uma **variável** sempre está associada a quatro (4) **características**:
  - o **nome** da variável: necessário para diferenciá-la das demais;
  - o **endereço** da variável: necessário para localizar a variável na memória principal;
  - o **tipo de dado associado** à variável: indica o tipo de informação que pode ser armazenada naquela área de memória e é necessário para que o compilador trate cada variável de acordo com o seu tipo;
  - o **conteúdo** da variável: é a informação efetivamente guardada na variável.

- As variáveis de um algoritmo representam os dados que devem ser armazenados na memória do computador para posterior processamento durante a execução do programa.
  - **Para o armazenamento** desses dados é preciso saber **qual o seu tipo** e **como eles poderão ser identificados** para viabilizar a sua utilização e manipulação, a qualquer momento.
  - Assim, as **variáveis precisam ser declaradas no programa**, recebendo um nome para serem identificadas, um tipo para especificar que valores pode armazenar e também podem receber um valor inicial.
- A **declaração de variáveis** pode ser feita da seguinte forma:

```
var
<identificador1>[, <identificador2>,...]: <tipo1>;
<identificador3>[, <identificador4>,...]: <tipo2>;
```

Os elementos entre [] são opcionais.

- **Uma variável nunca é eternamente igual a um valor, seu conteúdo pode ser alterado a qualquer momento**. Portanto para atribuir valores a variáveis podemos usar o sinal de ":".

EXEMPLO

var
Salário: real;
Idade: inteiro;
Nome: cadeia;
Sexo: caracter;

- Variáveis de entrada armazenam informações fornecidas por um meio externo (normalmente usuários ou discos).
- Variáveis de saída armazenam dados processados como resultados.

1.8. CONSTANTES

- São **ENDEREÇOS DE MEMÓRIA destinados a armazenar informações FIXAS**, que **NÃO se alteram durante a execução do programa**. Exemplo: PI = 3.1416.
- A declaração em pseudolinguagem das constantes utilizadas em um algoritmo é feita logo no início, antes da declaração das variáveis, da seguinte forma:

EXEMPLO

const
<identificador1> = <valor1>;
<identificador2> = <valor2>;
<identificadorN> = <valorN>;

EXEMPLO

```
var
nota1, nota2, nota3: inteiro;
peso: real;
_nome123: caractere;
const
PI = 3,14;
```

Obs.: A **diferença** entre **variáveis** e **constantes** é que **as informações contidas nas** variáveis **podem ser modificadas no decorrer de um algoritmo** e as **informações relacionadas a** constantes são declaradas no início do algoritmo e **não podem ser mais modificadas**.

1.9. IDENTIFICADORES

- São os **nomes atribuídos a variáveis, constantes e algoritmos.**

EXEMPLO

```
nome telefone idade_filha
nota1 salario pi
```

1.10. COMANDOS DE ENTRADA E SAÍDA

- A maioria dos algoritmos que escrevemos necessita receber dados externos, e, em algum momento, necessitará comunicar respostas, para tanto usamos os comandos de entrada e saída.
- Para a tarefa de buscar valores externos ao algoritmo utilizamos o comando LEIA e para enviar dados para a unidade de saída utilizamos o IMPRIMA ou ESCREVA.
- A **sintaxe** destes comandos é:

1.10.1. ENTRADA DE DADOS

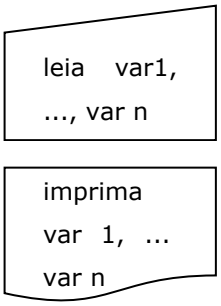
EXEMPLO

```
leia (<identificador 1>, <identificador 2>,..., <identificador n>)
```

1.10.2. SAÍDA DE DADOS

EXEMPLO

```
imprima (<identificador 1>, <identificador 2>,..., <identificador n>)
imprima (<expressão numérica>)
imprima( “cadeia de caracteres”)
```

FLUXOGRAMA	PORTUGOL
	LEIA (var 1,..., var n) IMPRIMA (var 1,..., var n) ou IMPRIMA (“mensagem”, var)

EXEMPLO

leia(num1, num2);
escreva(media);

1.11. OPERADORES

- São símbolos utilizados em expressões que contêm variáveis, constantes e funções.
- De acordo com os tipos de dados das variáveis e o resultado da operação, os operadores podem ser divididos em três tipos: **aritméticos**, **relacionais** e **lógicos**.

1.11.1. OPERADORES ARITMÉTICOS

São aqueles que atuam apenas sobre constantes, variáveis e funções numéricas, gerando um **resultado numérico** em uma expressão. São eles:

+	adição
-	subtração binária
*	multiplicação
/	divisão
** ou ^	exponenciação
-	menos unário

1.11.2. OPERADORES RELACIONAIS

São aqueles que realizam uma **comparação entre duas expressões e geram resultados lógicos**, isto é Verdadeiro ou Falso, são eles:

=	igual
<> ou ≠	diferente
>	maior que
<	menor que
>= ou ≥	maior ou igual
<= ou ≤	menor ou igual

1.11.3. OPERADORES LÓGICOS

São aqueles que geram **resultados lógicos através da comparação entre duas expressões lógicas**, são três:

conjunção	**E**	É aquele que exige que todos os termos da expressão sejam verdadeiros para que a expressão inteira seja verdadeira.
disjunção	**OU**	É aquele que exige que apenas um dos termos da expressão seja verdadeiro para que a expressão inteira seja verdadeira.
negação	**NÃO**	É aquele que inverte ou nega o valor lógico de um elemento.

Os **operadores lógicos têm seu resultado baseado na Tabela Verdade**. Eles operam sobre variáveis lógicas (V ou F) e têm como resultado valores lógicos (V ou F).

OU	V	F
V	V	V
F	V	F

E	V	F
V	V	F
F	F	F

NÃO	V	F
	F	V

1.12. PRIORIDADES

- Na execução de um comando complexo, podemos encontrar duas ou mais operações numéricas uma ao lado da outra, e neste caso devemos seguir regras de prioridades de operadores para sabermos qual será executado primeiro.
- No Portugol, as **prioridades** são:

PRIORIDADE	COMANDO	
1ª	parênteses	
2ª	funções	
3ª	menos unário	
4ª	** ou ^	
5ª	* e /	
6ª	+ e -	
7ª	relacionais	
8ª	lógicos	NÃO
9ª		E
10ª		OU

- **Se existirem duas operações de mesma prioridade, as operações serão realizadas da esquerda para a direita**.

1.13. EXPRESSÕES

- Para a construção dos algoritmos todas as **expressões** aritméticas devem ser colocadas em linhas. É importante também ressalvar o uso dos operadores correspondentes da aritmética tradicional para a computacional.

$\left[\frac{2}{3}+(5-3)\right]+1=$	$(2/3+(5-3))+1=$
Tradicional	Computacional

Fonte: Paradzinski, 2013

- A **modularização** é a divisão da expressão em partes, proporcionando maior compreensão e definindo prioridades para resolução da mesma. Como pode ser visto no exemplo anterior, em expressões computacionais usamos somente parênteses "( )" para modularização. Na informática podemos ter parênteses dentro de parênteses.

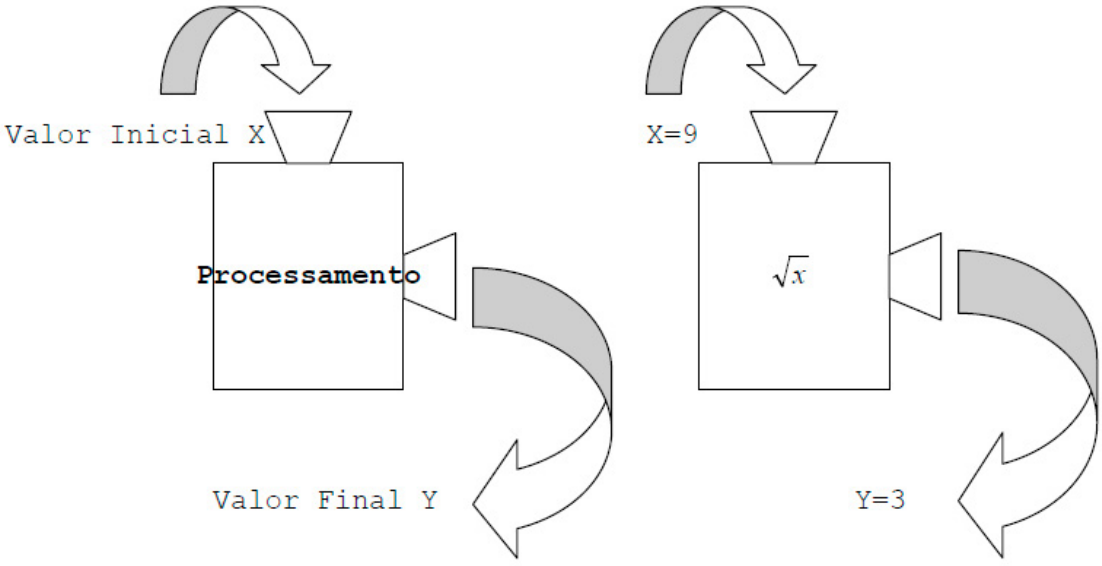
EXEMPLO

(2+2)/2=2

2+2/2=3

1.14. FUNÇÕES

- Uma **função** é um instrumento (Sub–algoritmo) que tem como objetivo **RETORNAR UM VALOR OU UMA INFORMAÇÃO**.
- A **chamada de uma função** é feita através da citação do seu nome seguido opcionalmente de seu argumento inicial entre parênteses.
- As **funções** podem ser predefinidas pela linguagem ou criadas pelo programador de acordo com o seu interesse.



Fonte: Paradzinski, 2013

- A seguir, ilustramos algumas funções comuns e importantes para nosso desenvolvimento lógico, entretanto, **cada linguagem de programação possui suas funções próprias**. As funções podem ser aritméticas, temporais, de texto, etc.

ABS ( )	VALOR ABSOLUTO
SQRT ( )	RAIZ QUADRADA
SQR ( )	ELEVA AO QUADRADO

Fonte: Paradzinski, 2013

1.15. {LINHAS DE COMENTÁRIO}

- Podemos inserir em um algoritmo **comentários** para aumentar a compreensão do mesmo, para isso basta que o texto fique entre chaves “{}”.

EXEMPLO

ler (raio); **{entrada}**

1.16. ESTRUTURA BÁSICA DE UM ALGORITMO

EXEMPLO

Algoritmo
declarações (variáveis, constantes, módulos)
início
comandos
fim.

EXEMPLO

Algoritmo maiorDeDoisNumerosInteiros;
var
num1, num2, maior: **inteiro**;
início
escreva(“Digite o valor do número 1:”);
leia(num1); {recebe o primeiro valor de entrada}
escreva(“Digite o valor do número 2:”);
leia(num2); {recebe o segundo valor de entrada}
se (num1 > num2) **então**
maior := num1; {define para a variável maior o valor da variável num1}

```
senão
maior:= num2; {define para a variável maior o valor da variável num2}
fim do se
escreva("O maior número é", maior);
fim
```

1.17. TESTE DE MESA OU TESTE CHINÊS

- Um algoritmo, depois de ser elaborado, pode (e deve) ser testado. Para tal, utilizamos um método conhecido como **teste de mesa** ou **teste chinês**, **que é como uma simulação de todos os passos**, ou seja, entradas, comandos e instruções do algoritmo, **a fim de saber se ele chega ao resultado a que se propõe e se a lógica está correta**.
- Para tal, **preenche-se uma tabela** com valores para as variáveis e **segue-se o fluxo de execução do algoritmo**, simulando a execução de cada instrução, ou seja, refazendo o que o computador faria ao executar cada instrução.
- **A cada comando simulado (executado), o valor das variáveis na tabela deve ser atualizado**.
- Se, para uma instrução executada, uma ou mais variáveis não ficaram com os valores esperados, há um erro na lógica do algoritmo.
- Vamos utilizar o algoritmo seguinte para colocar em prática a técnica. O problema será calcular o salário a receber seguindo os seguintes itens:
  - Informar o salário-base;
  - Haverá uma gratificação que é 5% do valor do salário-base;
  - Haverá um imposto que é 3% do valor do salário-base; e
  - O salário a receber é a soma do salário-base com a gratificação descontado o imposto.

```
Algoritmo calcularSalarioReceber;
  var
    salarioBase, gratificacao, imposto, salarioReceber: real;

  início
1   escreva("Informe o salário-base: ");
2   leia(salarioBase);
3   gratificacao := salarioBase * 5 / 100;
4   imposto := salarioBase * 3 / 100;
5   salarioReceber := salarioBase + gratificacao - imposto;
6   escreva("O salário a receber é ", salarioReceber);
  fim
```

Fonte: Araújo, 2012

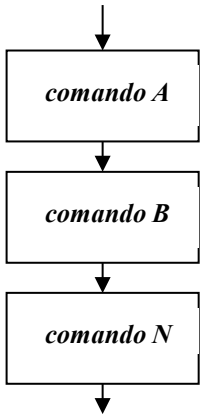
- Para facilitar a inspeção do algoritmo linha a linha, foi utilizada uma tabela na qual preenchemos os valores de cada variável em um determinado passo do algoritmo. Para esse teste, o salário-base será de R$ 1.000,00.

Linha do Algoritmo	salarioBase	gratificacao	imposto	salarioReceber
1	-	-	-	-
2	1000	-	-	-
3	1000	50	-	-
4	1000	50	30	-
5	1000	50	30	1020
6	1000	50	30	1020

- **Economize tempo não escrevendo os comandos e anotando apenas as alterações ocorridas nas variáveis**.
- Quando algum comando não efetuar alterações de valores nem uma saída, então não precisará anotar. Isto resultará em ganho de tempo na construção do teste.
- Para um algoritmo, podemos fazer vários testes de mesa, informando várias entradas diferentes para saber se o algoritmo trabalha de forma consistente para qualquer entrada.

1.18. ALGORITMOS COM ESTRUTURAS SEQUENCIAIS

- Trata-se de um grupo de comandos executados de **forma sequencial**. Os comandos são executados de cima para baixo, sendo que o próximo comando da lista só poderá ser executado após o término do comando anterior.
- É comum delimitar os comandos de uma sequência simples pelas palavras reservadas INÍCIO e FIM, formando assim um bloco de comandos.

FLUXOGRAMA	PORTUGOL
	início comando A; comando B; ... comando N; fim;

Exemplo de um algoritmo em que todos os comandos são executados de cima para baixo de forma sequencial -> **Calcular a área de um triângulo**.

EXEMPLO

Algoritmo
var
B, H, AREA: **real**;
início
leia (B) {base};
leia (H) {altura};
AREA ← (B * H) / 2 {calcula a área};
escreva(AREA);
fim.

1.19. ESTRUTURAS DE REPETIÇÃO

A- ENQUANTO

•Enquanto a condição for VERDADEIRA, o conjunto de comandos é executado.
•Se a condição é FALSA o conjunto NÃO é executado.
•**ENQUANTO**
< condição > **FAÇA**
comando 1
comando n
FIM-ENQUANTO

B- REPITA

•Os comandos internos a ele são executados ao menos uma vez, independente da condição.
•O laço REPITA é realizado somente se a condição for FALSA.
•**REPITA**
comando 1
...
comando n
ATÉ <condição>

C- PARA

•Uma variável que controla o início e o fim da execução.
•**PARA** variável **DE** valor1 **ATÉ** valor2 PASSO valor3 FAÇA
comando 1
...
comando n
FIM-PARA

1.20. ESTRUTURAS DE DADOS

1.20.1. PILHA

- Dada uma pilha P=(a(1), a(2),..., a(n)), dizemos que a(1) e o elemento da base da pilha; a(n) e o elemento topo da pilha; e a(i+1) está acima de a(i).
- Pilhas são também conhecidas como listas **LIFO (last *in first out*)**.

O último a entrar é o primeiro a sair

Vamos exercitar!

DIRETO DO CONCURSO

001. (FCC/TRT-16R(MA)/TÉCNICOJUDICIÁRIO/TI/2009) Pilha é uma estrutura de dados
a) cujo acesso aos seus elementos segue tanto a lógica LIFO quanto a FIFO.
b) cujo acesso aos seus elementos ocorre de forma aleatória.
c) que pode ser implementada somente por meio de vetores.
d) que pode ser implementada somente por meio de listas.
e) cujo acesso aos seus elementos segue a lógica LIFO, apenas.

Pilha é uma estrutura de dados cujo acesso aos seus elementos segue a **lógica LIFO,** apenas!

Letra e.

1.20.2. FILA

- É uma **lista linear** em que **a inserção é feita numa extremidade e a eliminação na outra**.
- Filas são também conhecidas como listas com disciplina de acesso: **FIFO (Fist *In First Out).***

O primeiro a entrar é o primeiro a sair ou em sua forma aportuguesada, PEPS ("Primeiro a Entrar Primeiro a Sair").

1.20.3. OUTRAS ESTRUTURAS DE DADOS RELEVANTES

- **Vetores**
  - Compostos por um **número fixo (finito) de elementos** de um determinado tipo de dados;
- **Lista**
  - Compostas por **nós que apontam para o próximo**, podendo ser encadeadas de formas diferentes; e
- **Árvores**
  - Cada elemento tem um ou mais elementos associados.

2. PYTHON

Rumo ao estudo do **Python**, uma linguagem de altíssimo nível, orientada a objeto, de tipagem dinâmica, fortemente interpretada e interativa.

2.1. O QUE É O PYTHON?

Python é uma linguagem de programação de alto nível e que tem como principal princípio permitir uma alta legibilidade de código através de uma sintaxe simples, porém, poderosa e que permite que programadores desenvolvam suas soluções utilizando poucas linhas de código (TREINAWEB, 2022).

A linguagem Python foi desenvolvida por Guido Van Rossum no final dos anos 80.

Dentre suas principais aplicações merecem destaque: *Data Science*, *Machine Learning*, *Big Data*, Desenvolvimento Web (Django e Flask) etc.

2.2. CARACTERÍSTICAS PRINCIPAIS DA LINGUAGEM

O Python possui as seguintes **características**:

- **Python** é uma **linguagem de *scripting***. Com o Python podemos criar *scripts* para automatizar diversas tarefas repetitivas (LUTZ, 2013).
- É uma **linguagem interpretada** (O código criado com o Python **não** é compilado) e **pseudo-compilada**, o que significa que um código-fonte escrito em Python é executado linha a linha pelo **interpretador** e, em seguida, executado pelo sistema operacional.
- É *free*: **Python** é **distribuída sob uma licença própria** (compatível com a GPL), que impõe poucas restrições.

A implementação de **Python** está disponível em www.python.org, que fornece também informações sobre a linguagem.

A linguagem Python **possui uma curva de aprendizado muito pequena** e é bastante reconhecida pela sua comunidade diversa, acolhedora e bastante ativa (TREINAWEB, 2022).

Os programas em **Python** podem ser escritos em um editor de texto ou em um **Ambiente de Desenvolvimento Integrado (IDE – *Integrated Development Environment*)**.

Obs.: **IDE** é um software que auxilia no desenvolvimento de aplicações, muito utilizado por desenvolvedores, com o objetivo de facilitar diversos processos (ligados ao desenvolvimento), que combinam ferramentas comuns em uma única interface gráfica do usuário (GUI).

- Principais IDEs para desenvolvimento (TREINAWEB, 2022):
  - **Eclipse**: é uma excelente IDE, muito utilizada no mercado. Seu uso facilita a criação de aplicações Python tanto para Desktop ou Web.
  - **PyCharm**: conta com desenvolvimento multitecnologias, em que, além do Python, oferece suporte para CoffeeScript, TypeScript, Cython, JavaScript, SQL, HTML/CSS, linguagens de modelo, AngularJS, Node.js, etc.
  - **Jupyter Notebook**: derivado do IPython, é baseada na estrutura servidor-cliente, que permite a manipulação de documentos. O Jupyter Notebook independe de linguagem e suporta diversos ambientes de execução, entre elas: Julia, R, Haskell, Ruby, e o próprio Python.
  - **Spyder**: muito utilizado principalmente por cientistas de dados, já que possui integração com as principais bibliotecas como NumPy, SciPy, Matplotlib e IPython.
- Assim, como desenvolvedor, você escreve **arquivos Python (.py)** em um **editor de texto** e coloca esses arquivos no interpretador Python a ser executado. **Através da linha de comando**, é possível testar um trecho de código em **Python sem a necessidade de escrever em um editor de texto ou em um IDE (*Integrated Development Environment* – Ambiente de Desenvolvimento Integrado).**
- A linguagem Python e seu **interpretador** estão **disponíveis para as mais diversas plataformas**, desde Unix (Linux, FreeBSD, Solaris, MacOS X, etc.), Windows, .NET,

versões antigas de MacOS até consoles de jogos eletrônicos ou mesmo alguns celulares e palmtops.

- Para que seja usado em determinado **sistema operacional não suportado**, **é possível gerar o Python** a partir do programa fonte **utilizando um compilador C**. Nesse caso, o código fonte é traduzido para o formato **bytecode**, que é **multiplataforma** e pode ser distribuído de forma **independente**.
  - Você faz seu **programa Python** (**.py**), compila-o transformando num **bytecode** (**.pyc**), e, então, pode-se pegar esse bytecode e jogar em qualquer plataforma que possua uma **máquina virtual Python** (conhecida como PVM -*Python Virtual Machine*): Windows, Mac, Linux, etc., SEM precisar recompilar. É O MESMO CÓDIGO!
- **Python** é uma **linguagem com verificação de tipos**, mas **tipada dinamicamente**, o que significa que o próprio interpretador infere os tipos de dados **SEM** a necessidade de o desenvolvedor informar.
  - De **tipagem dinâmica** – As variáveis no Python podem armazenar qualquer tipo de dados, independente do seu valor atual.

Veja a seguir as **características de tipagem** das variáveis **Python**:

Tipagem forte	**Não permite fazer operações com tipos que sejam incompatíveis. A linguagem não realiza conversões automaticamente entre os tipos suportados.**
Tipagem dinâmica	**O tipo de variável pode mudar ao longo do programa.** A tipagem dinâmica é a característica que muitas linguagens de programação possuem por não exigirem que os tipos de dados sejam declarados, pois são capazes de realizar esta escolha dinamicamente. Desta forma, durante a execução do programa ou até mesmo durante a sua compilação, o tipo de uma variável poderá ser alterado.

- **Python** é uma **linguagem multiparadigma**. **Suporta vários paradigmas de programação**, como:
  - **funcional**;
  - **imperativo** (baseado em comandos que instruem a execução do programa);
  - **procedural** (permite definir procedimentos e funções para serem executados linearmente);
  - **orientado a objetos**.
- Lembre-se do mnemônico **FIPO**, para facilitar a memorização!
- O **Python** é uma **linguagem orientada a objetos**.
  - Python permite criar programas utilizando um dos paradigmas mais utilizados no mercado, a orientação a objetos;
  - **Quase tudo em Python é um** objeto **(Objetos** são estruturas compostas de atributos e métodos (funções)).
  - **Python suporta funcionalidades comuns na orientação a objetos**: herança, **herança múltipla**, polimorfismo, reflexão e introspecção.

  - **Python não suporta sobrecarga de métodos** que é um conceito do polimorfismo.
  - **Python faz tratamento de exceções.** A coleta de lixo é usada para remover elementos da memória quando não são mais necessários.
- **Python** pode ser **facilmente entendida por qualquer usuário**. Os módulos que suportam as extensões podem ser escritos em qualquer linguagem compilada. Extensões podem adicionar funções, variáveis e tipos de objetos.
- **Inclui suporte para concorrência** com suas linhas de execução (*threads*) e **suporte para programação de rede** com seus **soquetes**. Tem também mais **suporte para programação funcional** que outras linguagens de programação não funcionais.
- Em **Python, c %= a é a forma reduzida de representar c = c % a. Assim como c += a equivale a c = c+a.**

2.3. SINTAXE DO PYTHON

A sintaxe de **Python** não é baseada diretamente em nenhuma linguagem comumente usada. Python é conhecido por possuir uma sintaxe simples e possui algumas características marcantes da linguagem (TREINAWEB, 2022):

- utiliza **indentação** por espaços;
- não utiliza ponto e vírgula (;)para finalizar uma instrução;
- uma **variável** pode armazenar diferentes tipos de dados;
- não há chaves ({}) para delimitar o início e final de um bloco de código.

No código escrito em Python destacado a seguir pode-se visualizar algumas das características aqui citadas (TREINAWEB, 2022):

```
print("Meu primeiro programa em Python")
nome_variavel = 6
if nome_variavel == 6:
print("O número é 6")
else:
print("O número não é 6")
```

2.4. INDENTAÇÃO (OU RECUO)

Em **Python**, **o código é agrupado através da indentação**, ou seja, a **indentação vai dizer se uma instrução está dentro de um bloco ou de outro** – diferentemente de outras linguagens que possuem blocos limitados por chaves ({ }) ou palavras-chaves (begin/end).

2.5. COMENTÁRIOS E DOCSTRINGS

Os **comentários**, incluídos nas linhas dos *scripts* Python, **não são processados, uma vez que correspondem a notas explicativas que têm o objetivo de descrever algo que se tenha necessidade**, para melhor organizar os códigos.

Em Python, comentários são iniciados com #, tudo que estiver depois do # será ignorado pelo interpretador, portanto considerado como comentários, o fechamento do comentário acaba quando acabar a linha do interpretador.

As **docstrings** são mais comumente introduzidas no início de uma classe, de uma função ou no início do programa para definir o escopo do software ou o escopo de métodos, seu símbolo padrão são **as três aspas duplas ou simples**. Também não modificam o programa executado.

Por exemplo, entre com as linhas abaixo.

```
#
# Comentários:
#
def fat1(n):
# Entre com fat1(n) para calcular o fatorial de n
# Exemplo: fat1(5)
if (n <= 1):
return 1
return n * fat1(n-1)
#
# Docstrings:
#
def fat2(n):
'''
Entre com fat2(n) para calcular o fatorial de n
Exemplo: fat2(5)
'''
if (n <= 1):
return 1
return n * fat1(n-1)
```

2.6. CONHECENDO VARIÁVEIS E CONSTANTES

O Python é uma das principais linguagens que possui **tipagem dinâmica.** Além disso, o Python possui como característica a **tipagem forte**, ou seja, a linguagem **não** realiza conversões automaticamente entre os tipos suportados.

No Python os **tipos suportados** são:

int ou inteiro	Representado por toda e qualquer informação numérica que pertença ao conjunto de números inteiros relativos (números positivos, negativos ou o zero). Ex.: 5, 1354982, -11

float	*Float* ou "número de ponto flutuante" é um número, positivo ou negativo, contendo uma ou mais casas decimais. É representado, portanto, por números decimais, ou seja, números que possuem partes fracionadas. Ex.: 1.10, -26.59, 35e3 (e indica uma potência de 10).
string	É uma cadeia de caracteres que representam textos.
booleano	Variáveis booleanas armazenam valores lógicos que podem ser verdadeiro ou falso.
tipo complexo (complex)	Armazenam dados com formato misto, ou seja, dados de diferentes tipos em uma mesma sentença. Escritos com um "j" como a parte imaginária. Ex.: 3 + 5j, 2j, -5j.

2.7. CONVERSÕES E CASTING DE TIPOS DE DADOS EM PYTHON

A conversão de tipo de dados ocorre quando **um tipo é convertido em outro**.

Na linguagem Python, podem acontecer **dois tipos de conversão:**

- **implícita**; e
- **explícita**.

A **conversão implícita** acontece quando um tipo de dados é convertido automaticamente para outro tipo (Ex.: quando um tipo int é convertido para um tipo maior como float).

Veja mais:

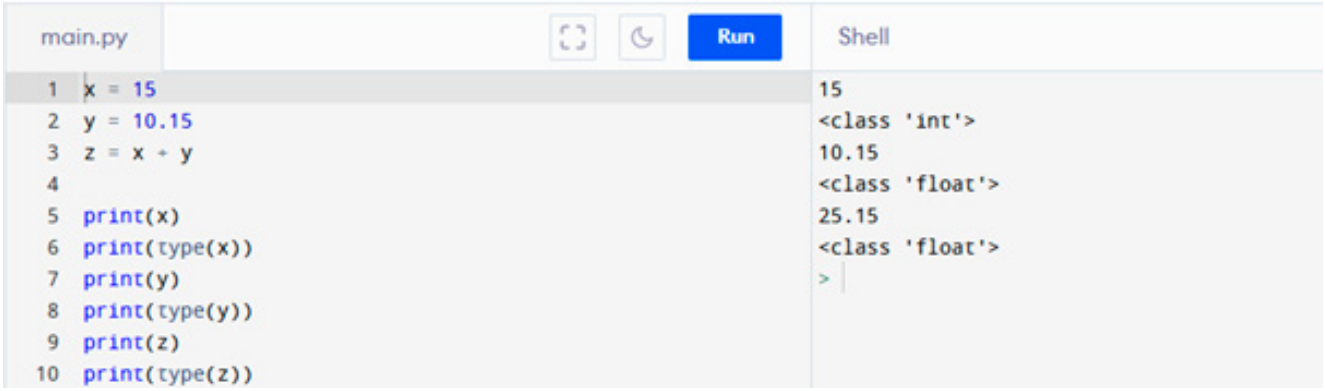


Figura. Conversão Implícita (ARAÚJO, 2022)

Observe pela figura que a variável x é do tipo int e a variável y é do tipo float. A variável z, no contexto dado, recebe a soma de x mais y. **Para acontecer isso, a linguagem Python sempre converte tipos de dados menores** (no exemplo, o tipo int) **em tipos de dados maiores** (no exemplo, o tipo float) **para evitar a perda de dados**. No final, é do tipo float o valor da soma de um número tipo int mais um número do tipo float. Por isso, a variável z é do tipo float (ARAÚJO, 2022).

A seguir, veja o que acontece se fizermos uma operação entre variável do tipo int com uma variável do tipo str (ARAÚJO, 2022):

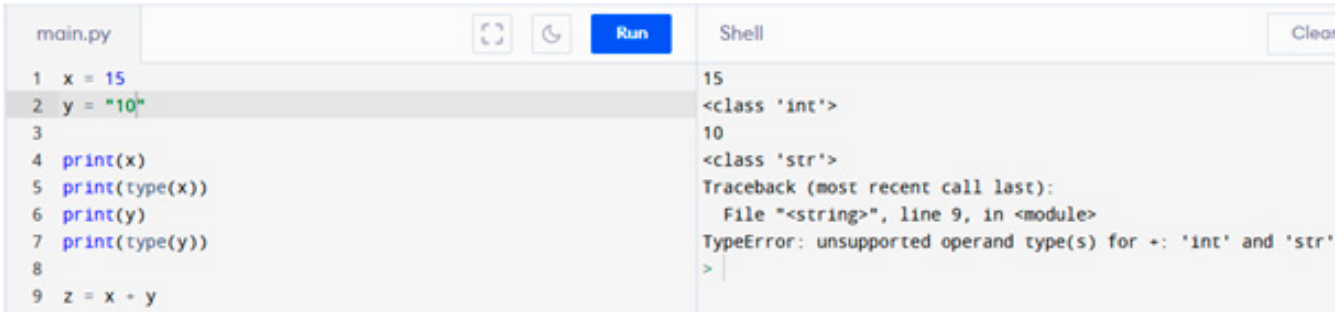


Figura. Conversão Implícita (ARAÚJO, 2022).

No exemplo anterior, tentamos fazer uma operação com as variáveis x (do tipo int) e y (do tipo str). Porém, **Python não foi capaz de fazer a conversão implícita nessa situação.** Nesse caso, é necessário fazer uma **conversão explícita**, destacada a seguir.

Na **conversão explícita**, o programador converte o tipo de dados de um objeto no tipo de dados necessário (ARAÚJO, 2022).

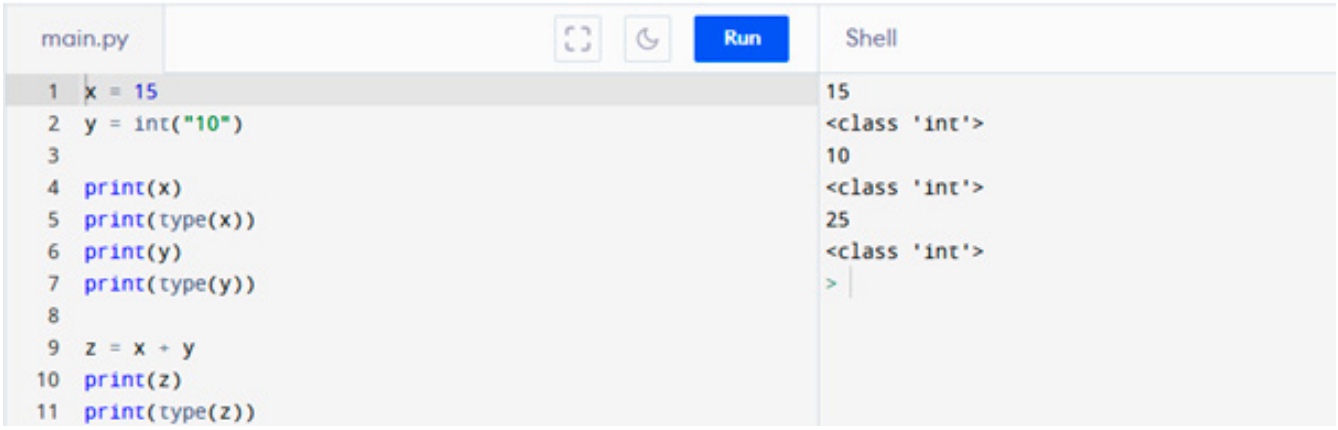
Para isso, a linguagem Python disponibiliza algumas funções, tais como:

- **int()**
- **float()**
- **complex()**
- **str()**

A **conversão explícita** também é conhecida como ***casting* de tipos de dados**, pois o programador converte (altera) o tipo de dados dos objetos (ARAÚJO, 2022).

```
Sintaxe:
<tipo de dados necessário> (expressão)
```

Então, corrigindo o exemplo anterior, tem-se:



Agora a variável y receber o valor string “10” convertido explicitamente para o tipo int (y = int(“10”)). A variável z então pode receber a operação entre duas variáveis do tipo int (no caso, a operação de soma).

Mas podemos corrigir de uma outra forma:

```
main.py                                   Shell
1  x = str(15)                            15
2  y = "10"                               <class 'str'>
3                                         10
4  print(x)                               <class 'str'>
5  print(type(x))                         1510
6  print(y)                               <class 'str'>
7  print(type(y))                         >
8
9  z = x + y
10 print(z)
11 print(type(z))
```

No exemplo anterior, a variável x receber o valor inteiro 10 convertido explicitamente para o tipo str (x = str(10)). A variável z então pode receber a operação entre duas variáveis do tipo str (no caso, a operação de concatenação).

Vejamos então as funções para conversão de tipos de dados!

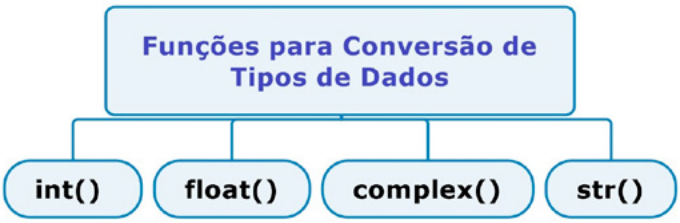


Figura. Funções para conversões de tipos de dados (QUINTÃO, 2023)

- A **função int()** constrói um número inteiro a partir de:
  - um literal inteiro;
  - um literal de ponto flutuante;
  - arredondando para baixo, para o número inteiro anterior;
  - um literal de *string*, desde que a *string* represente um número inteiro.

Alguns exemplos:

```
main.py                                   Shell
1  w = int(15)                            15
2  print(w)                               1
3                                         3
4  x = int(1.84)                          Traceback (most recent call last):
5  print(x)                                 File "<string>", line 10, in <module>
6                                         TypeError: can't convert complex to int
7  y = int("3")                           >
8  print(y)
9
10 z = int(15 + 5j)
11 print(z)
```

Figura. Usos função int() (ARAÚJO, 2022)

Sobre os exemplos anteriores, temos algumas observações:

- Na linha 4, o valor float 1.84 foi arredondado para baixo, o que fez a variável x receber o valor int 1;
- Não há possibilidade do tipo complexo para o tipo int, o que gerou erro na linha 10.
- A **função float()** constrói um número de ponto flutuante a partir de:
  - um literal inteiro;
  - um literal de ponto flutuante; ou
  - um literal de *string*;
    - Desde que a *string* represente um número de ponto flutuante ou um inteiro.

Exemplos:

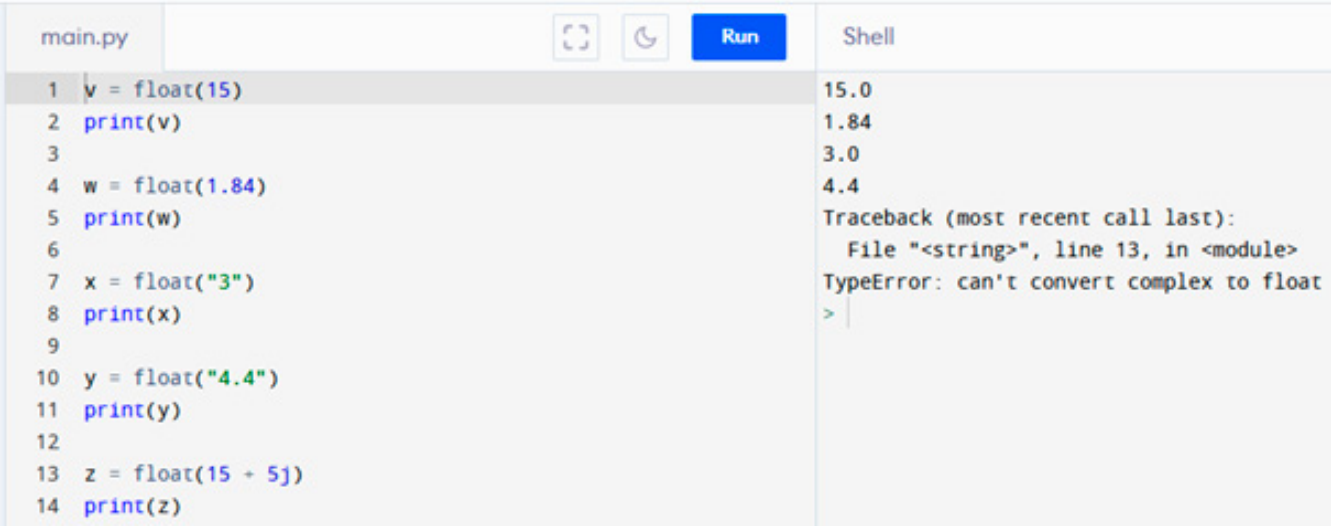


Figura. Usos função float() (ARAÚJO, 2022)

Sobre os exemplos do uso da função float, temos algumas observações:

- Na linha 1, o valor int 15 foi convertido para o valor de ponto flutuante 15.0;
- Na linha 7, a string "3", que representaria um valor int, foi convertido para o valor de ponto flutuante 3.0;
- Não há possibilidade do tipo complexo para o tipo float;
  - O que gerou erro na linha 13.
- A **função complex()** constrói um número complexo a partir de:
  - um literal inteiro;
  - um literal de ponto flutuante; ou
  - um literal de *string*;
    - Desde que a *string* represente um número de ponto flutuante ou um inteiro.

Exemplos:

```
main.py                                    Run        Shell
1  v = complex(15)                                    (15+0j)
2  print(v)                                           (1.84+0j)
3                                                     (3+0j)
4  w = complex(1.84)                                  (4.4+0j)
5  print(w)                                           (15+5j)
6                                                     >
7  x = complex("3")
8  print(x)
9
10 y = complex("4.4")
11 print(y)
12
13 z = complex(15 + 5j)
14 print(z)
```

Figura. Usos função complex() (ARAÚJO, 2022)

- A **função str()** constrói uma string a partir de uma ampla variedade de tipos de dados, incluindo:
  - literais de string;
  - literais inteiros; ou
  - literais de ponto flutuante.

Exemplos:

```
main.py                                    Run        Shell
1  x = str("str1")                                    str1
2  print(x)                                           15
3                                                     1.84
4  y = str(15)                                        >
5  print(y)
6
7  z = str(1.84)
8  print(z)
```

Figura. Usos função str() (ARAÚJO, 2022)

DIRETO DO CONCURSO

002. (QUADRIX/CREA-TO/ANALISTA DE SISTEMAS/2019) Quanto aos conceitos e às técnicas de programação de computadores, julgue o item.

Em um programa escrito em linguagem Python, o comando de atribuição x = int(5.9) fará com que a variável x passe a armazenar um valor inteiro igual a 6.

Vimos que a **função int()** constrói um número inteiro a partir de um inteiro, um float (arredondando para baixo para o número inteiro anterior), ou uma *string* (fornecendo o inteiro que representa a string).

```
main.py                                   Run    Shell
1  x = int(5.9)                                  5
2  print(x)                                      >
```

Conforme visto, o comando de atribuição x = int(5.9) fará com que a variável x passe a armazenar um valor inteiro igual a 5.

Errado.

---

003. (CESPE/MPOG/ANALISTA DE SISTEMAS/TECNOLOGIA DA INFORMAÇÃO/2013) A expressividade do código é uma característica importante no desenvolvimento e manutenção de um software. Python e Ruby são exemplos de linguagens que apresentam essa qualidade. Acerca dessas linguagens, julgue os itens subsequentes.

Em Python, o comando int("1") cria um objeto do tipo int, que recebe 1 como parâmetro no seu construtor.

Na **linguagem Python** o que prevalece é o tipo **de criação do objeto**.

Na questão, o inteiro ("int") transformará um valor *string* em um valor inteiro. Se fizéssemos o contrário (str(1)), o valor 1 se transformaria em uma *string*.

A prioridade sempre será do tipo e **não** do valor passado.

Certo.

2.8. OPERADORES

Operadores Aritméticos	Operadores Lógicos
+ **Soma**	> **Maior**
- **Subtração**	< **Menor**
* **Multiplicação**	>= **Maior ou igual**
/ **Divisão**	<= **Menor ou igual**
// **Divisão de Inteiros**	== **Igual**
** **Potenciação**	!= **Diferente**
% **Módulo**	not **Negação**
	and **Conjunção**
	or **Disjunção**

- O "~" é o **NOT binário**, no **Python**. O XOR binário é o "^".

Vamos ao detalhamento dos operadores lógicos (Negação, Conjunção e Disjunção):

Operador	Nome	Descrição	Tipo
and	**Conjunção**	Retorna True se ambos os operandos tiverem o valor True	Binário
or	**Disjunção**	Retorna True se um dos operandos tiver o valor True	Binário
not	**Negação**	Inverte o resultado lógico	Unário

Tabela. Operadores Lógicos em Python

Exemplos:

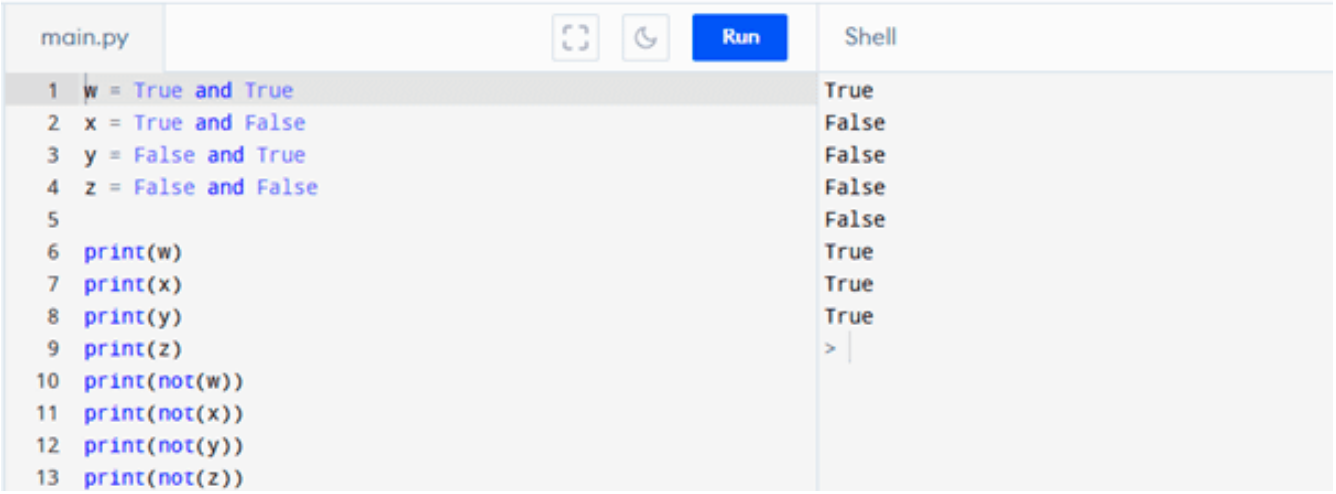


Figura. Uso de Operadores Lógicos (ARAÚJO, 2022)

DIRETO DO CONCURSO

004. (CETAP/AL-RR/ANALISTA DE SISTEMAS – ADAPTADA/2010) Sobre a linguagem de programação PYTHON, julgue o item seguinte.
O operador lógico de conjunção (“e”, como em a e b) é &&.

Em Python, o operador de conjunção é o and, como em a and b.
Errado.

2.9. STRINGS

Uma **string** é uma **cadeia de caracteres variáveis de tamanho dinâmico.**
Vejamos exemplos de funções do Python aplicáveis a *strings*:

Função	Objetivo
capitalize()	Retorna uma *string* com o primeiro caractere maiúsculo.
count()	Retorna o número de vezes que um valor aparece em uma *string*.

Função	Objetivo
endswith()	Retorna True se a *string* terminar com um valor.
find()	Retorna a posição da primeira ocorrência de um valor em uma *string* ou -1 se não existir.
isalnun()	Retorna True para uma condição (todos os caracteres serem alfanuméricos) e False se não for.
len()	Retorna o tamanho da *string*.
lower() ou casefold()	Retorna a *string* em letras minúsculas.
replace()	Substitui uma *string* por outra.
startswith()	Retorna True se a *string* começar com um valor.
split()	Retorna as *substrings* encontradas entre o separador indicado.
strip()	Remove os espaços em branco do início e do fim da *string*.
title()	Converte o primeiro caractere de cada palavra para maiúscula.
upper()	Retorna a *string* em letras maiúsculas.

2.10. COLEÇÕES

- Uma **coleção** é uma estrutura de dados utilizada para armazenar objetos. Em Python, **existem quatro tipos de coleções**, que são:

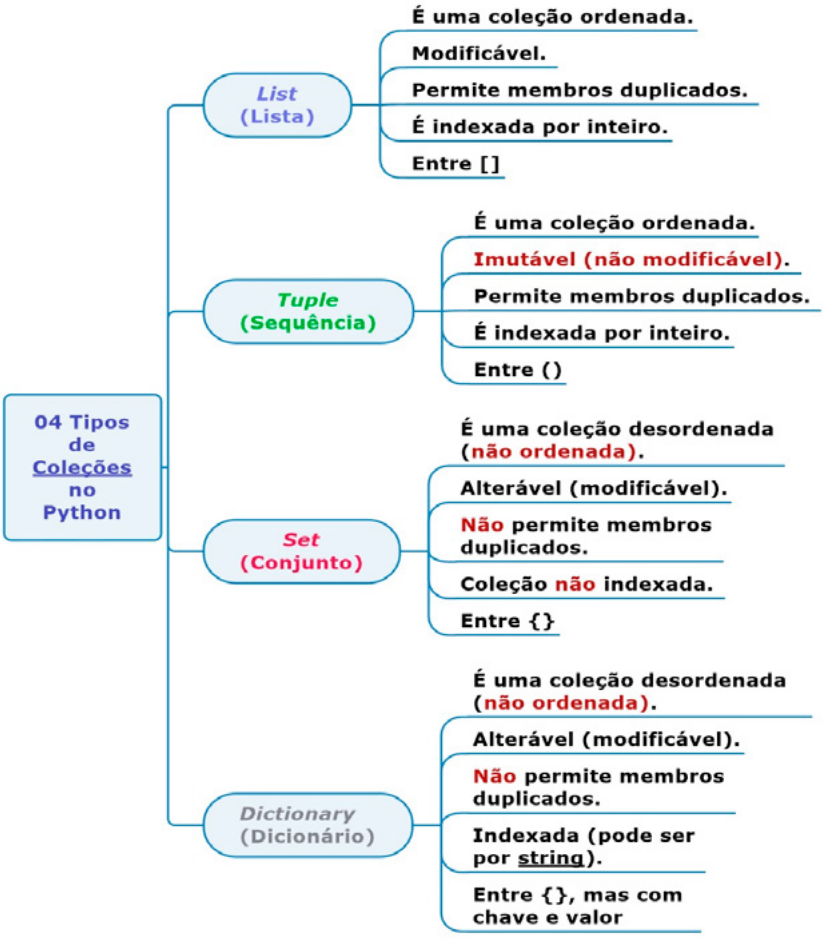
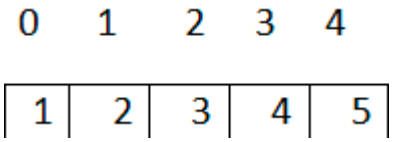


Figura: Coleções no Python (QUINTÃO, 2023)

2.11. LISTA

Uma lista é a estrutura de dados mais básica do Python e armazena os dados em sequência, em que cada elemento possui sua posição na lista, denominada de **índice**. O primeiro elemento é sempre o índice zero e a cada elemento inserido na lista esse valor é incrementado (TREINAWEB, 2022).

No Python, **uma lista pode armazenar qualquer tipo de dado primitivo (*string*, inteiro, *float*,** *etc.***)**. Na imagem seguinte pode-se ver como uma **lista** se comporta:



Fonte: (TREINAWEB, 2023)

Além disso, elas são mutáveis e definidas pelos colchetes, da seguinte maneira:

```
[1, 2, 3]
```

```
nome_da_lista = [] # Criação de uma lista vazia
nome_da_lista = [1, 2, 3] # Criação de uma lista de inteiros
nome_da_lista = [1, "Olá, mundo!", 1.1] # Lista com vários tipos diferentes
```

A definição de uma **lista** é análoga à de uma **variável** qualquer, porém isolando o conteúdo com colchetes.

O uso das aspas (duplas ou simples) **apenas** é necessário caso estejamos inserindo na lista uma ***string***, se armazenássemos apenas um número, não seria necessário.

Em **Python** é possível acessar **listas** de qualquer ponta:

- **índices positivos**: relacionam a posição na lista, começando em 0 (primeiro);
- **índices negativos**: relacionam a posição na lista, de trás para frente, começando em -1 (último). Assim, o índice -1 acessa o último item da lista, -2 acessa o penúltimo, etc.

Existe uma coleção de **métodos de** lista, como:

- inserir ao final (***append***),
- inserir em uma posição arbitrária (***insert***),
- remover (***remove***); e
- ordenar (***sort***).

Para **inserir um novo dado a uma lista qualquer**, utilizamos um método chamado **.append**. Exemplo:

```
>>>teste = []
>>>teste.append('zero')
>>>teste.append('um')
>>>teste
 ['zero','um']
```

Obs.: Infelizmente o comando.**append só consegue adicionar um dado na lista por vez.**

list.extend(L) -> é o mesmo que o **append**, mas ao invés de receber apenas um valor, recebe uma nova lista, que será adicionada ao final de "list";

list.count(c) -> retorna quantas vezes o valor "c" aparece na lista;

list.sort() -> ordena os valores da lista.

2.12. DISPERSÕES (OU DICIONÁRIOS)

Um **dicionário** é um **conjunto de elementos que possuem** índices, ou seja, dicionários são formados por **chaves** e seus respectivos **valores**, em que as chaves são os **índices**.

Exemplo de dicionário:

```
>>>calculos = { 1:'primeiro periodo' , 2:'segundo periodo' , 4:'terceiro periodo' , 8:'quinto periodo' }
>>>print calculos
{ 1:'primeiro periodo' , 2:'segundo periodo' , 4:'terceiro periodo' , 8:'quinto periodo' }
>>>print calculos[8]
'quinto periodo'
>>>print calculos[8]='números complexos'
>>>print calculos
{ 1:'primeiro periodo' , 2:'segundo periodo' , 4:'terceiro periodo' , 8:'números complexos' }
```

Figura. Exemplo de Dicionário

- Possui também uma coleção de **métodos para** dicionários, como:
  - para obter chaves (***Keys***),
  - para obter valores (***values***),
  - para copiar (***copy***); e
  - para verificar a existência de uma chave (***has_key***).

2.13. ESTRUTURAS DE DECISÃO

- A **estrutura de** decisão serve para desviar o fluxo de execução do programa dependendo de uma condição.
- A linguagem **Python** possui 3 estruturas de decisão: **if**, **if-else** e **if-elif-else**.

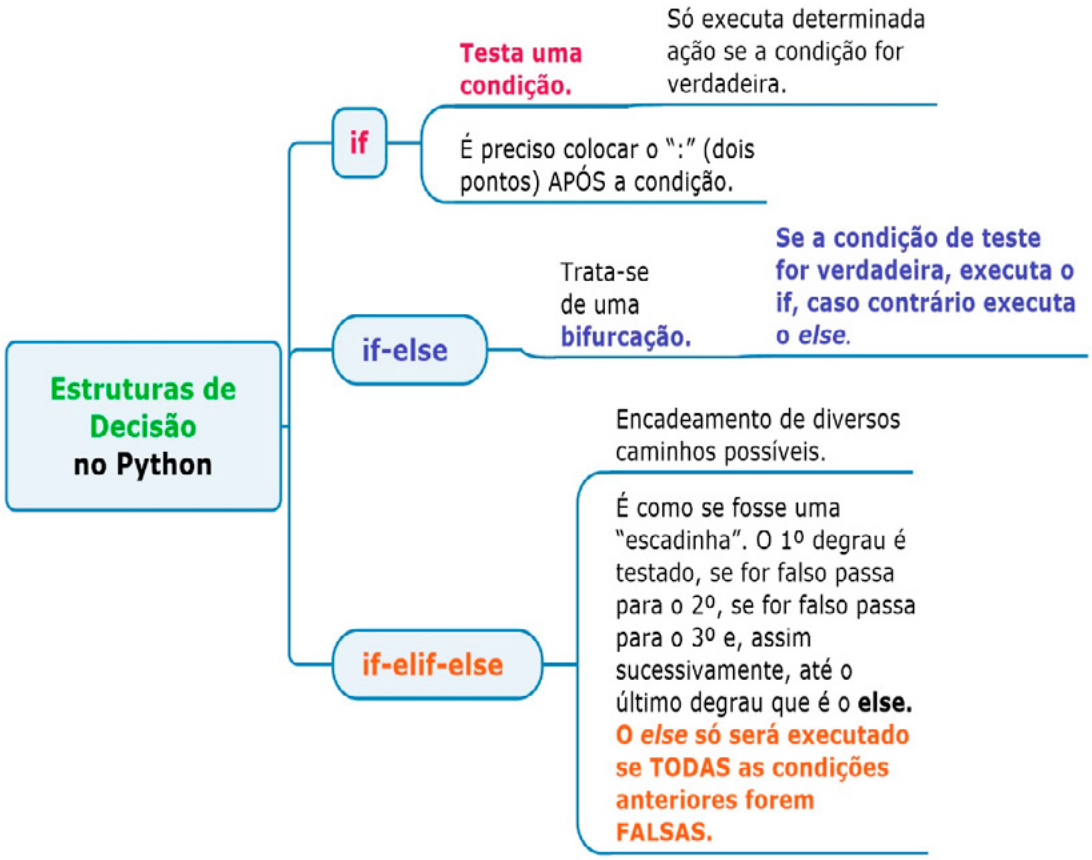


Figura. Estruturas de Decisão no Python. Fonte: QUINTÃO (2023)

2.14. ESTRUTURAS DE REPETIÇÃO

- A **estrutura de repetição (laços** ou **loops)** no **Python** permite que determinado bloco de instruções seja executado repetidamente segundo certos critérios. São elas: **for, for-else, while, while-else**.

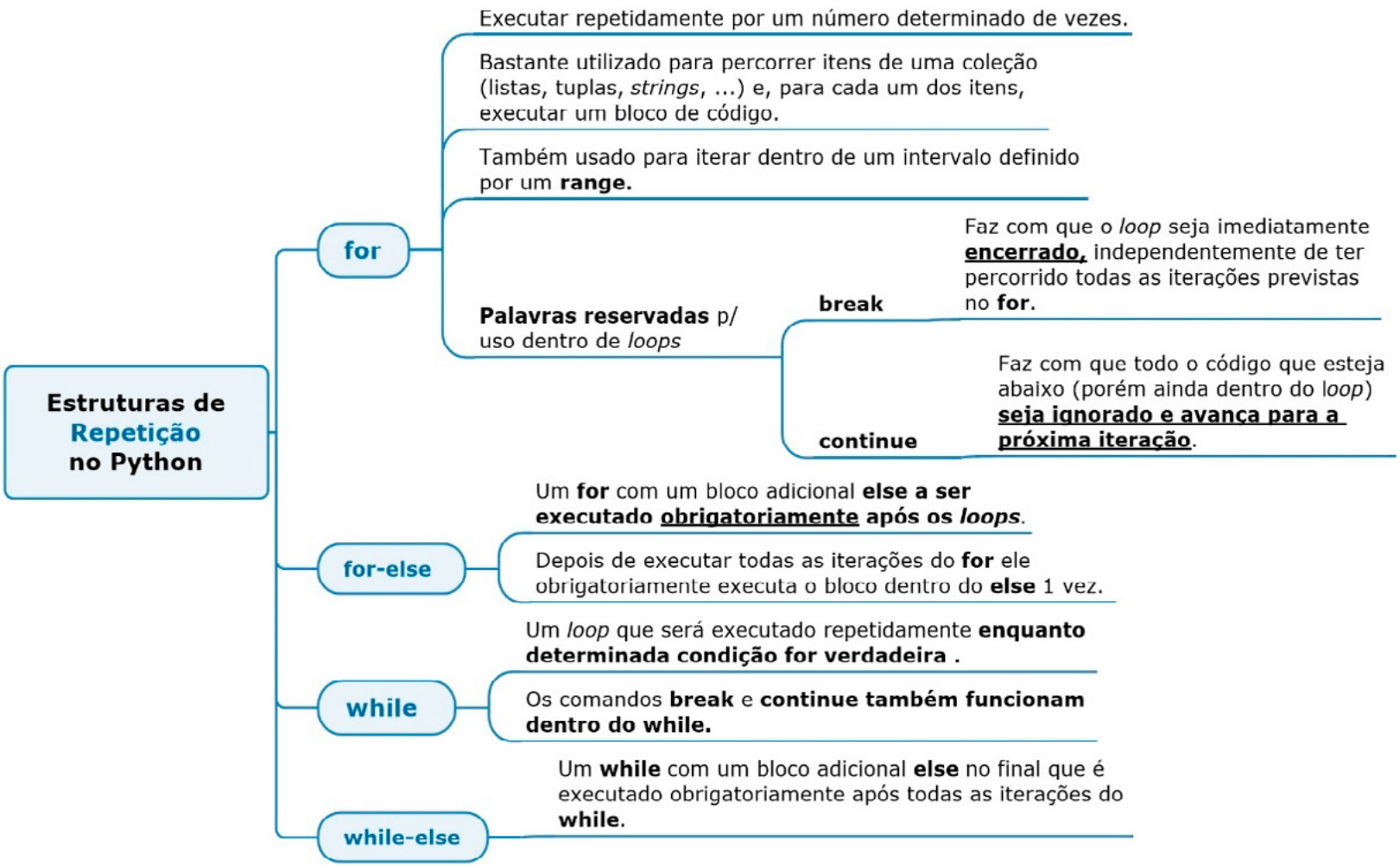


Figura. Estruturas de Repetição no Python. Fonte: QUINTÃO (2023)

2.15. FUNÇÕES

- As **funções** são **blocos de código** que apenas são executados quando são chamados. As funções podem:
  - receber dados através de parâmetros; e
  - retornar dados como resultado.
- **Python** permite a definição de **FUNÇÕES** por meio da palavra-chave **def**, **seguida do nome da função e parênteses** (Exemplo: def NomeFuncao(parametrosFuncao)).

Sintaxe:

```
def nomeFuncao([parâmetro formal 1, parâmetro formal 2,...]):
bloco de instruções
[return] valor
```

EXEMPLO

Exemplo de programa em Python, destacando a **sequência de Fibonacci**.

```
# Python 3: Fibonacci series up to n
>>> def fib(n):
>>>     a, b = 0, 1
>>>     while a < n:
>>>         print(a, end=' ')
>>>         a, b = b, a+b
>>>     print()
>>> fib(1000)
0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987
```

- **Uma função não precisa de parâmetros. Se precisar, pode-se usar um ou mais parâmetros**. Esses parâmetros definidos na função são chamados de **parâmetros formais**. Eles representam as informações passadas para funções e são especificados após o nome da função, entre parênteses.
- **Uma função pode ou não retornar algum valor**. Quando for necessário retornar, usa-se a **declaração return**. Essa declaração gera a saída do método atual e faz com que o controle de fluxo retorne onde o método foi invocado.
- Traz consigo inúmeras **bibliotecas**, além de possibilitar integração com outras.
- A palavra-chave **lambda,** em **Python**, é utilizada para criar **funções anônimas (f**unções sem nome predefinido).
- Detalhes para a **função range** no Python:

class range(start, stop[, step])	**start: número de início. Nesse caso, conta de start (incluso) até stop (não-incluso). Por padrão, o step tem valor igual a 1. Se step for fornecido, faz a iteração de step em step.**
class range(stop)	Se o argumento *start* for omitido, o valor padrão é 0.

EXEMPLO

```
>>> list(range(10))
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> list(range(1, 11))
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
>>> list(range(0, 30, 5))
[0, 5, 10, 15, 20, 25]
>>> list(range(0, 10, 3))
[0, 3, 6, 9]
>>> list(range(0, -10, -1))
[0, -1, -2, -3, -4, -5, -6, -7, -8, -9]
>>> list(range(5))
[0, 1, 2, 3, 4,]
>>> list(range(0))
[]
>>> list(range(1, 0))
[]
>>> list(range(0, 12))
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]
```

DIRETO DO CONCURSO

005. (UNIRIO/UNIRIO/2014) Sobre o comando range para construção de listas na linguagem Python, é CORRETO afirmar que:

a) range(4,6) gera a lista [4,5].

b) range(5) gera a lista [1,2,3,4,5].

c) range(4,6) gera a lista [4,5,6,7,8,9].

d) range(5,1) gera a lista [5].

e) range(5,1,-2) gera a lista [4,5].

a) Certa. Temos que range(4,6) retornará a lista [4, 5]. Vamos aos detalhes da **função range** no Python:

class **range(**start, stop**[**, step**])**	**start:** número de início. **Nesse caso**, conta **de start (incluso) até stop (não-incluso)**. Por padrão, o *step* tem valor igual a 1. **Se *step* for fornecido, faz a iteração de *step* em *step***.
Assim, **range(4,6)** possui start 4; stop 6 (não incluso); step 1 (Padrão) e retorna a **lista [4,5]**	

b) Errada. Observe que **range(5)** retorna a lista [0, 1, 2, 3, 4].

c) Errada. Conforme visto, **range(4,6)** retorna a lista [4, 5].

d) Errada. Temos que **range(5,1)** retorna [ ] porque o step padrão é 1 e o stop é menor que o start.

e) Errada. Temos que **range(5,1,-2)** retorna [5, 3] porque o step é negativo.

Letra a.

---

006. (FGV/ALE-RO/2018) Analise o código Pyhton a seguir.

```
for k in range(0, 4, -1):
   print k
```

Assinale a opção que indica o número de valores printados na execução desse código.

a) Zero.

b) Um.

c) Dois.

d) Quatro.

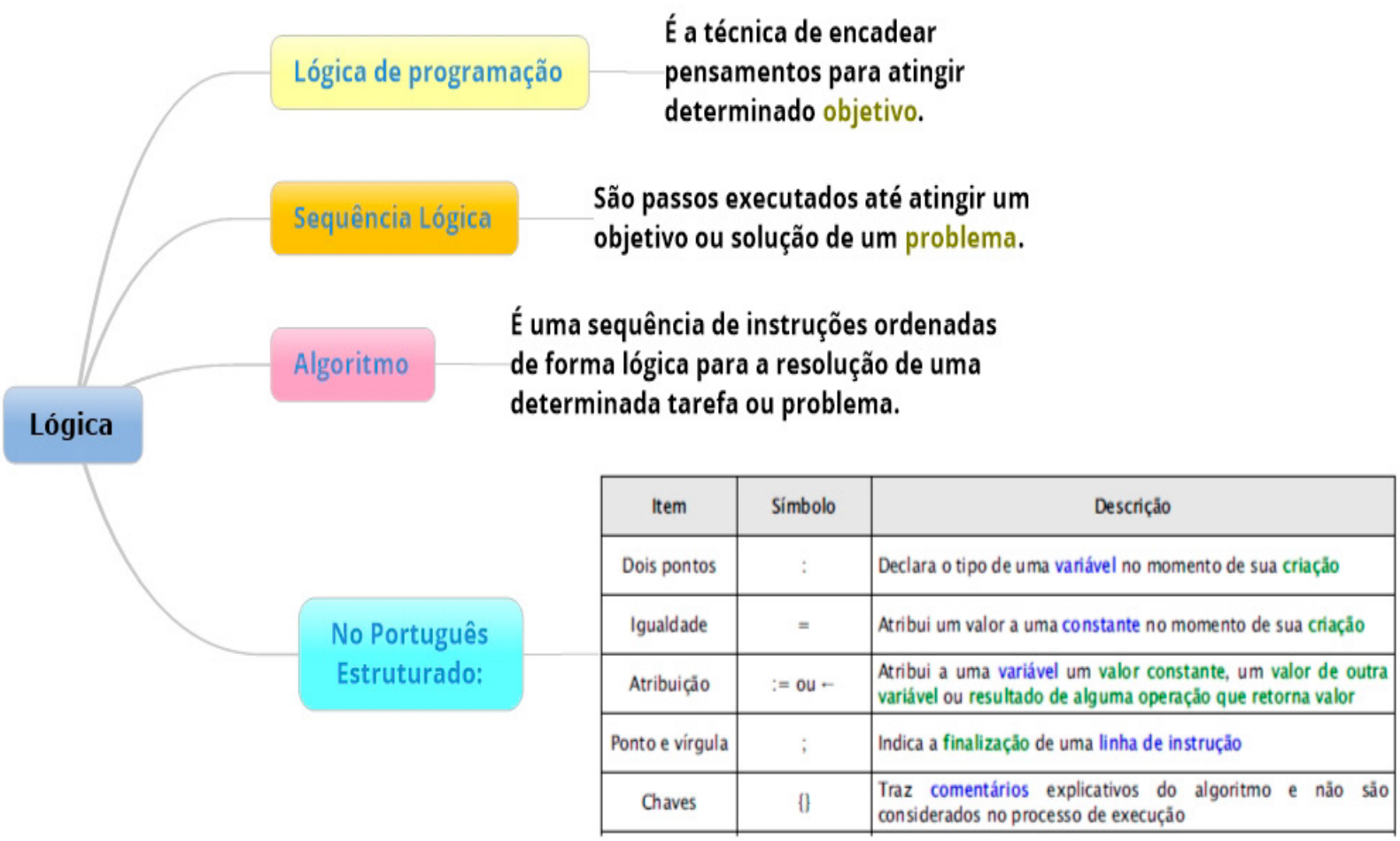
e) Cinco.

A **função range(0, 4, -1)** irá retornar os valores iniciando em zero e terminando em 4 (não incluso), com um step (passo) de -1 em -1.

Mas, de -1 em -1, nunca chegaremos a 4, e a função range, portanto, não retornará nenhum valor.

Letra a.

RESUMO



Item	Símbolo	Descrição
Dois pontos	:	Declara o tipo de uma variável no momento de sua criação
Igualdade	=	Atribui um valor a uma constante no momento de sua criação
Atribuição	:= ou ←	Atribui a uma variável um valor constante, um valor de outra variável ou resultado de alguma operação que retorna valor
Ponto e vírgula	;	Indica a finalização de uma linha de instrução
Chaves	{}	Traz comentários explicativos do algoritmo e não são considerados no processo de execução

Figura. Lógica de Programação. Fonte: Quintão (2023)

O **Python** possui as seguintes **características**:

- Open Source;
- Multiplataforma;
- Linguagem Interpretada;
- Tipagem Dinâmica;
- Tipagem Forte;
- Multiparadigma;
- Multinicho.

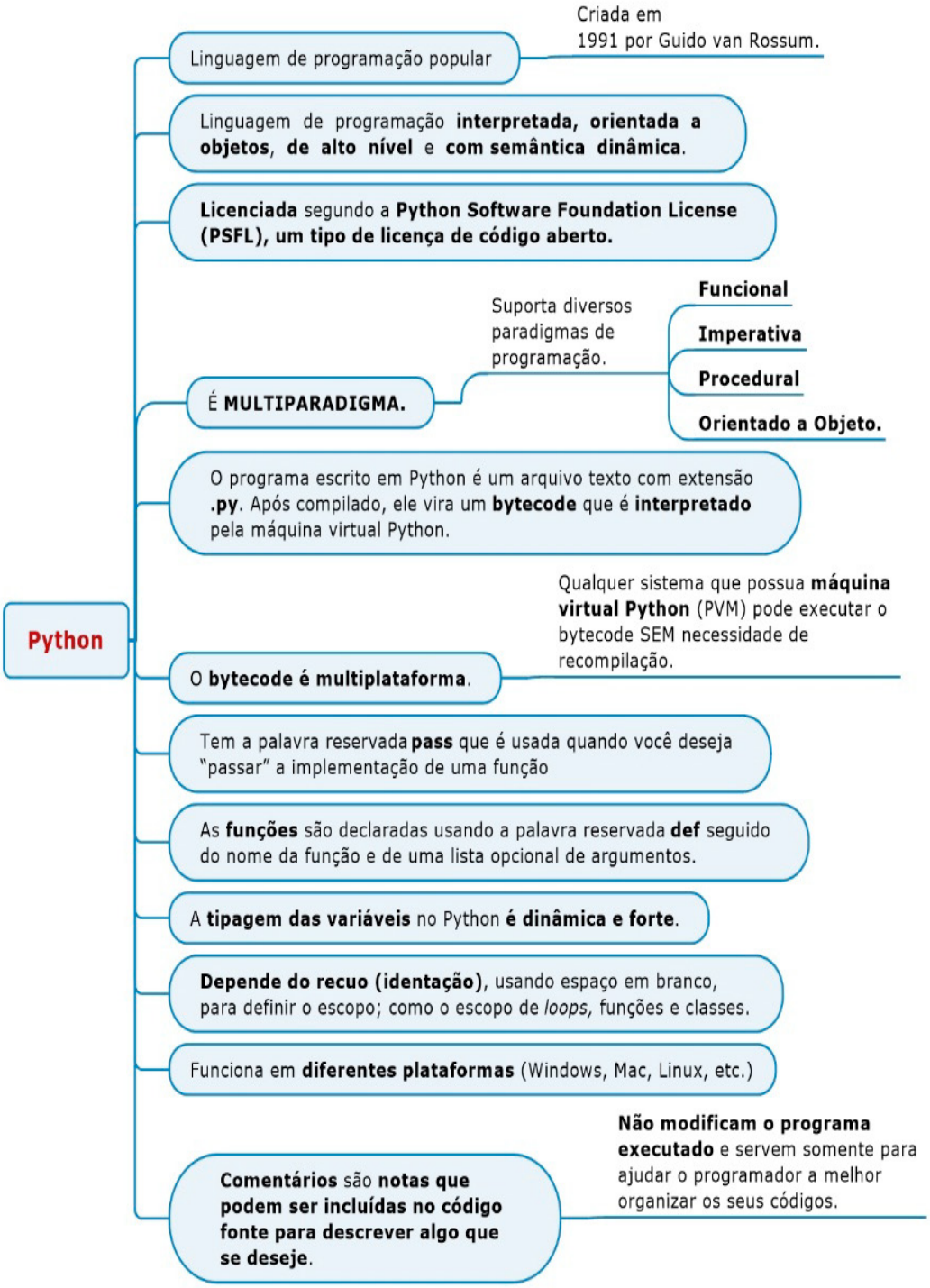


Figura Linguagem Python. Fonte: QUINTÃO (2023)

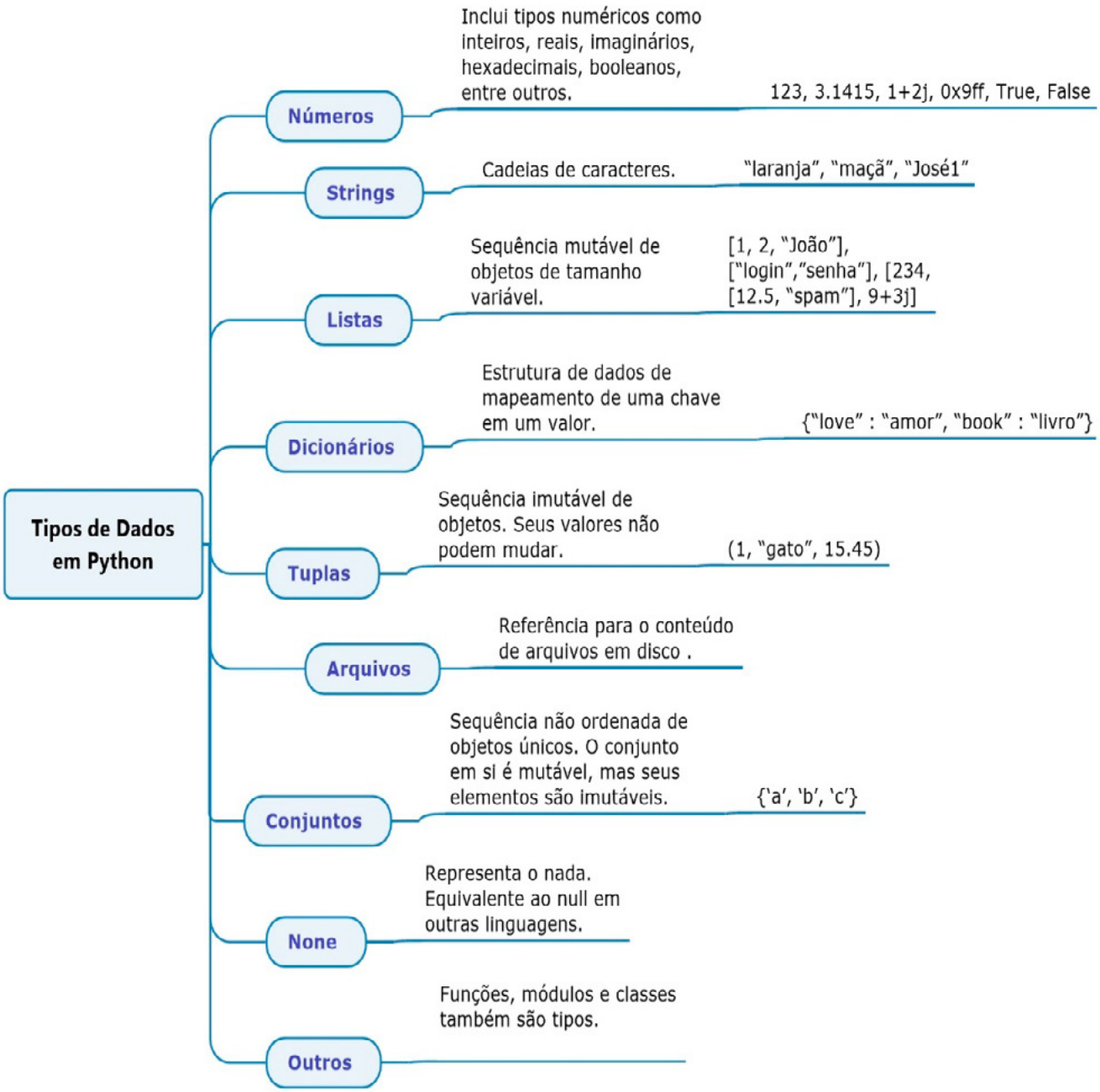


Figura. Exemplos de tipos de Dados em Python (QUINTÃO, 2023)

- Uma **coleção** é uma estrutura de dados utilizada para armazenar objetos. Em Python, **existem quatro tipos de coleções**, que são:

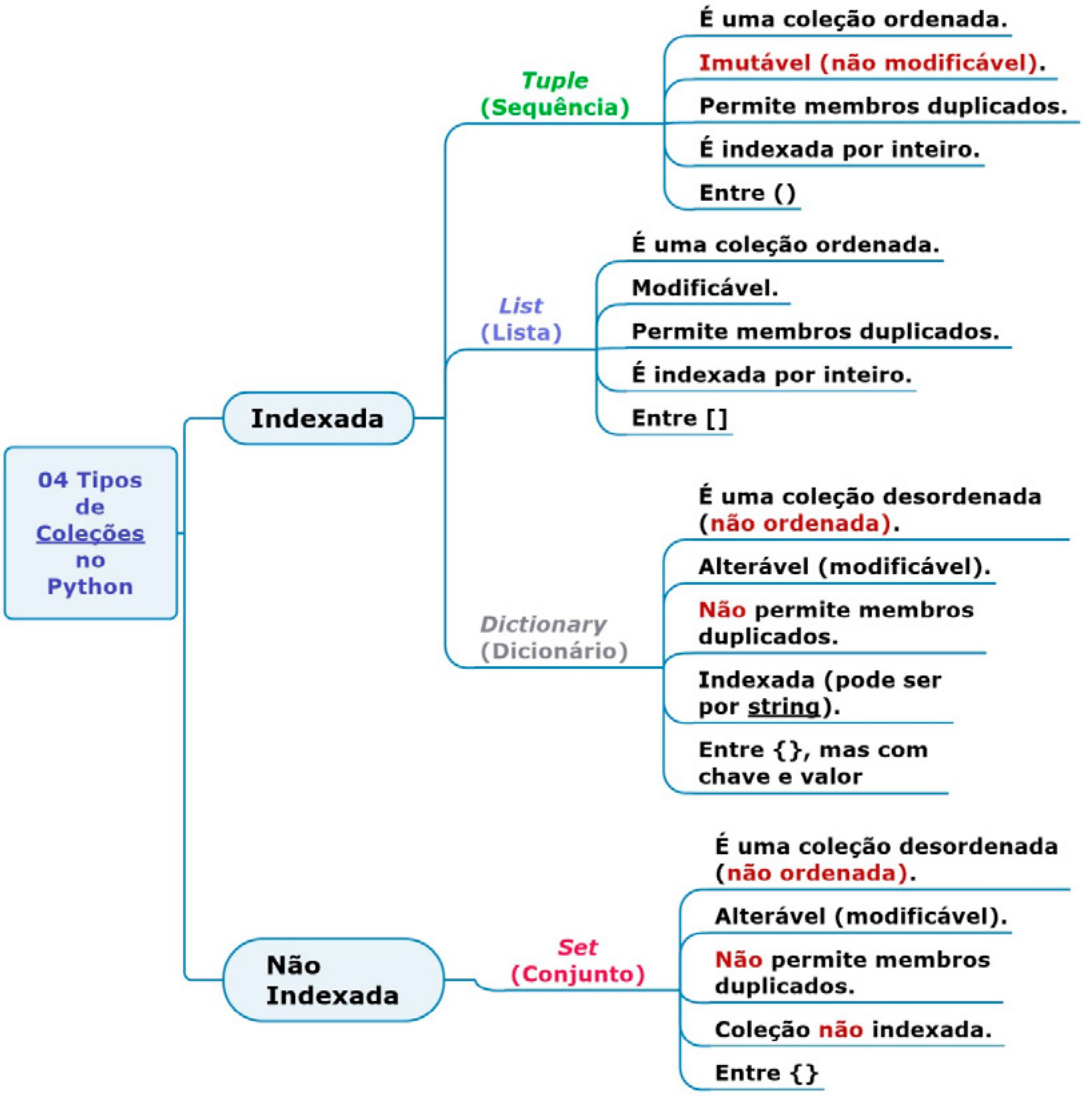


Figura. 4 Tipos de Coleções no Python (QUINTÃO, 2023)

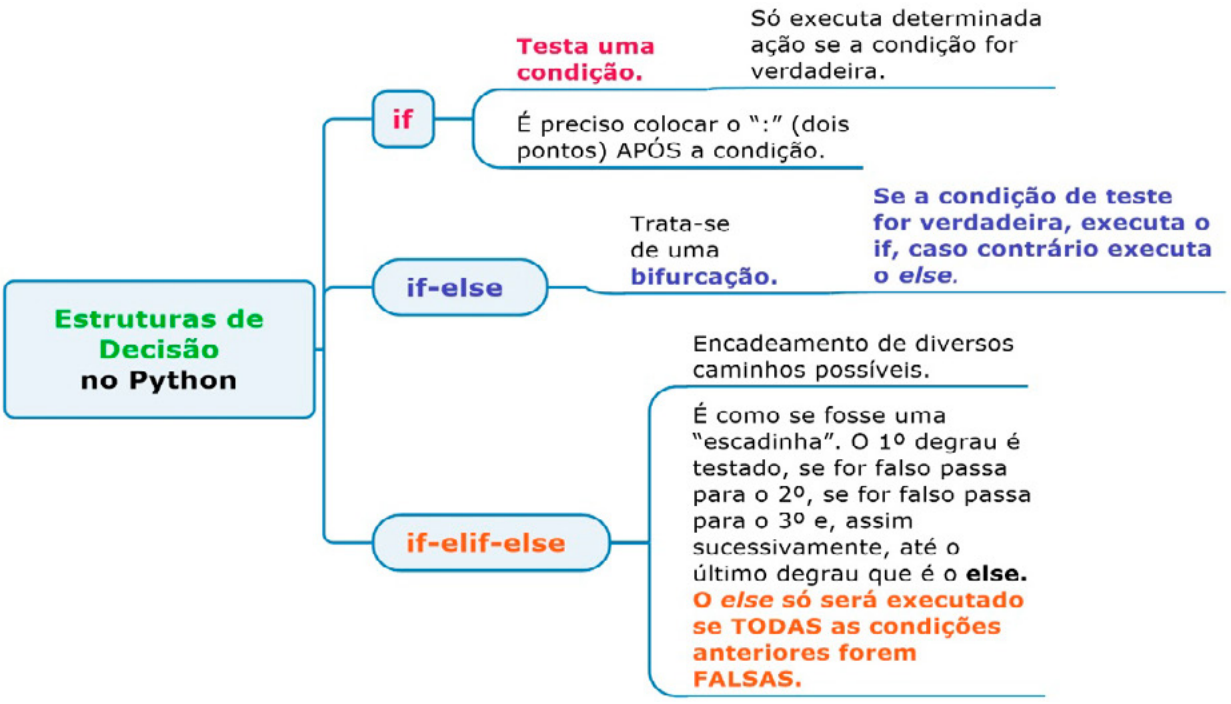


Figura. Estruturas de Decisão no Python. Fonte: QUINTÃO (2023)

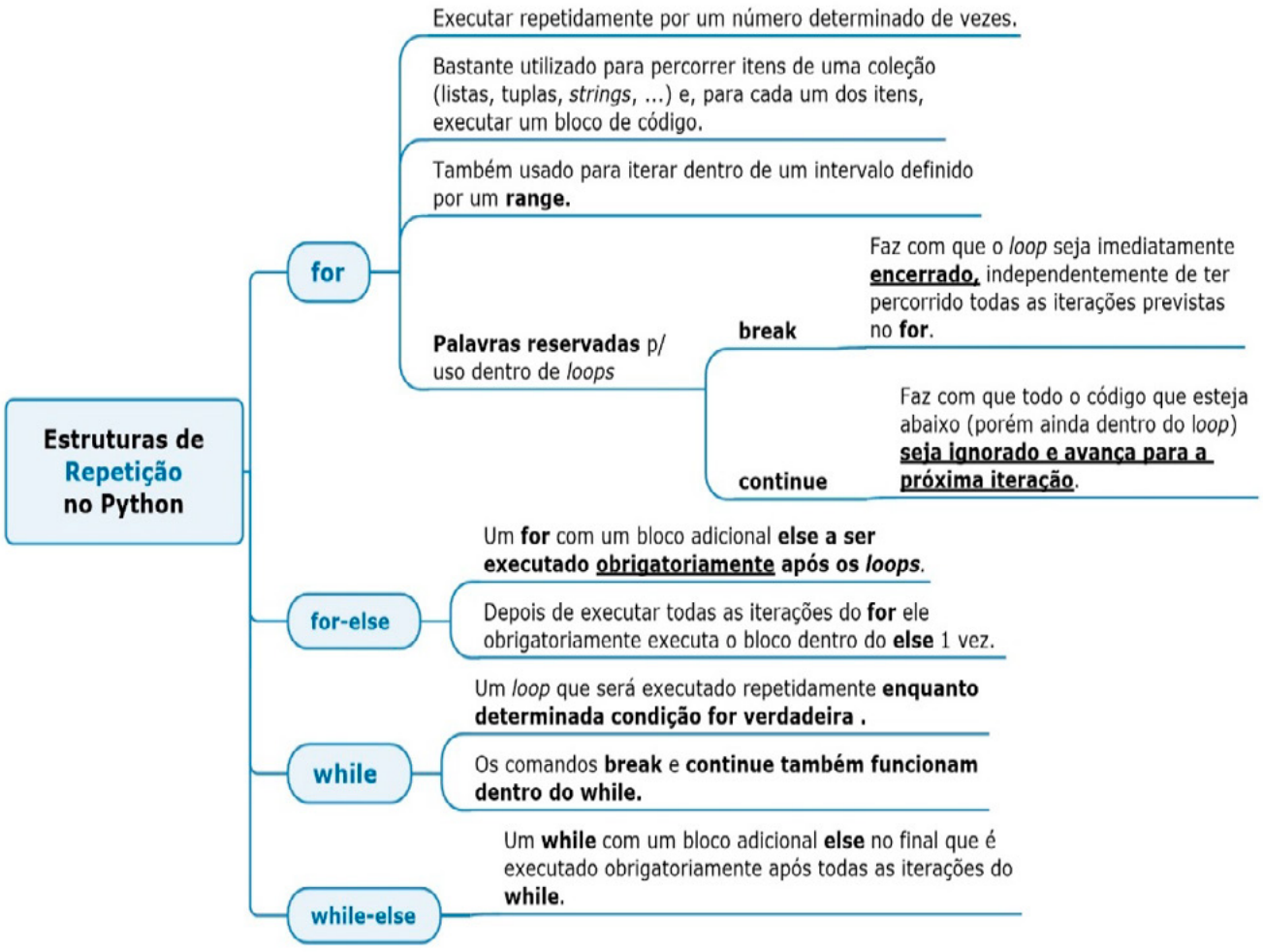


Figura. Estruturas de Repetição no Python. Fonte: QUINTÃO (2023)

QUESTÕES COMENTADAS EM AULA

001. (FCC/TRT-16R(MA)/TÉCNICOJUDICIÁRIO/TI/2009) Pilha é uma estrutura de dados
a) cujo acesso aos seus elementos segue tanto a lógica LIFO quanto a FIFO.
b) cujo acesso aos seus elementos ocorre de forma aleatória.
c) que pode ser implementada somente por meio de vetores.
d) que pode ser implementada somente por meio de listas.
e) cujo acesso aos seus elementos segue a lógica LIFO, apenas.

002. (QUADRIX/CREA-TO/ANALISTA DE SISTEMAS/2019) Quanto aos conceitos e às técnicas de programação de computadores, julgue o item.
Em um programa escrito em linguagem Python, o comando de atribuição x = int(5.9) fará com que a variável x passe a armazenar um valor inteiro igual a 6.

003. (CESPE/MPOG/ANALISTA DE SISTEMAS/TECNOLOGIA DA INFORMAÇÃO/2013) A expressividade do código é uma característica importante no desenvolvimento e manutenção de um software. Python e Ruby são exemplos de linguagens que apresentam essa qualidade. Acerca dessas linguagens, julgue os itens subsequentes.
Em Python, o comando int("1") cria um objeto do tipo int, que recebe 1 como parâmetro no seu construtor.

004. (CETAP/AL-RR/ANALISTA DE SISTEMAS – ADAPTADA/2010) Sobre a linguagem de programação PYTHON, julgue o item seguinte.
O operador lógico de conjunção ("e", como em a e b) é &&.

005. (UNIRIO/UNIRIO/2014) Sobre o comando range para construção de listas na linguagem Python, é CORRETO afirmar que:
a) range(4,6) gera a lista [4,5].
b) range(5) gera a lista [1,2,3,4,5].
c) range(4,6) gera a lista [4,5,6,7,8,9].
d) range(5,1) gera a lista [5].
e) range(5,1,-2) gera a lista [4,5].

006. (FGV/ALE-RO/2018) Analise o código Pyhton a seguir.

```
for k in range(0, 4, -1):
   print k
```

Assinale a opção que indica o número de valores printados na execução desse código.
a) Zero.
b) Um.

c) Dois.
d) Quatro.
e) Cinco.

EXERCÍCIOS

PYTHON

007. (CESPE (CEBRASPE)/DPE-RO/ANALISTA DA DEFENSORIA PÚBLICA – PROGRAMAÇÃO/2022) Na linguagem Python, são consideradas sequências mutáveis as
a) strings.
b) cadeias.
c) tuplas.
d) listas.
e) ranges.

008. (CESPE (CEBRASPE)/PC PB/ESCRIVÃO DE POLÍCIA/2022) Na linguagem Python, o tipo de uma variável em tempo de execução é definido pelo interpretador pelo recurso denominado
a) tipagem dinâmica.
b) modo interativo.
c) sintaxe.
d) interpretação bytecode.
e) empacotamento.

009. (CESPE (CEBRASPE)/PC PB/PERITO OFICIAL – CRIMINAL – TECNOLOGIA DA INFORMAÇÃO/2022) Python é uma linguagem procedural que utiliza quatro tipos de dados predefinidos para lidar com coleções: conjuntos, dicionários, listas e tuplas. A respeito desses tipos de dados, julgue os itens a seguir.
I – O conjunto permite o armazenamento de uma tupla, mas não o de uma lista.
II – A tupla é idêntica à lista, exceto pela forma mais simples com que sua declaração é realizada.
III – A lista é um tipo de dados variável que permite a alteração de seus elementos após a sua criação.
Assinale a opção correta.
a) Apenas o item I está certo.
b) Todos os itens estão certos.
c) Apenas o item II está certo.
d) Apenas os itens I e III estão certos.
e) Apenas os itens II e III estão certos.

010. (CESPE (CEBRASPE)/SERPRO/ANALISTA – ESPECIALIZAÇÃO: CIÊNCIA DE DADOS/2021) A respeito da linguagem de programação Python, julgue o item a seguir.
Listas são coleções alteráveis de qualquer tipo de objeto — como, por exemplo, outras listas — capazes de gerar um efeito top-down sem limite de níveis.

011. (CESPE (CEBRASPE)/SEED-PR/PROFESSOR – EDUCAÇÃO BÁSICA E JORNADA/2021) Na linguagem de programação Python, existem 3 estruturas para armazenar dados indexados. A estrutura cujos valores são imutáveis depois de sua criação é conhecida como

a) lista.

b) operador.

c) tupla.

d) classe.

e) dicionário.

012. (CEBRASPE(CESPE)/PF/AGENTE DE POLÍCIA FEDERAL/2018) Julgue o item, relativo a noções de programação Python e R.

Considere o programa a seguir, na linguagem Python.

```
if 5 > 2
{
print(“True!”)
}
```

A sintaxe do programa está correta e, quando executado, ele apresentará o seguinte resultado.

```
True!
```

013. (FCC/TRE-CE/TÉCNICO JUDICIÁRIO – PROGRAMAÇÃO DE SISTEMAS/2012) Sobre Python é correto afirmar:

a) É uma linguagem compilada, ou seja, o código-fonte de um programa é lido pelo compilador, que cria um arquivo binário, executável diretamente pelo hardware.

b) É uma linguagem fortemente tipada, ou seja, é preciso declarar variáveis e seus tipos.

c) Suporta funcionalidades comuns na orientação a objetos: herança, herança múltipla, polimorfismo, reflexão e introspecção.

d) Uma lista em Python é um conjunto de valores acessados por um índice numérico, inteiro, começando em 1. Assim como em outras linguagens, a lista pode armazenar apenas valores de um mesmo tipo.

e) Uma String Python é uma sequência imutável, alocada estaticamente, com restrição de tamanho.

014. (FGV/CM/CARUARU/ANALISTA LEGISLATIVO – INFORMÁTICA/2015) Analise o código Python a seguir.

```
L1=[10,20,30]
L2=[40,50]
L1.append(L2)
print L1
```

Assinale a opção que descreve corretamente o que acontece quando esse programa é executado no Python 2.7.

a) Produz uma mensagem de erro, porque tenta executar uma operação inválida.

b) Exibe "[10, 20, 30, [40, 50]]".

c) Exibe "[10, 20, 30, 40, 50]".

d) Exibe "[10, 20, 30], [40, 50]".

e) Exibe "[]".

015. (FGV/CM/CARUARU/ANALISTA LEGISLATIVO – INFORMÁTICA/2015/ADAPTADA) Analise o código Python a seguir.

```
L1=[10,20,30]
L2=[40,50]
L1.append(L2)
print (L1)
```

Assinale a opção que descreve corretamente o que acontece quando esse programa é executado no Python 3.7.4.

a) Produz uma mensagem de erro, porque tenta executar uma operação inválida.

b) Exibe "[10, 20, 30, [40, 50]]".

c) Exibe "[10, 20, 30, 40, 50]".

d) Exibe "[10, 20, 30], [40, 50]".

e) Exibe "[]".

016. (INÉDITA/2023) Em vez de vetores, Python inclui somente dois tipos de estrutura de dados: listas e dispersões, chamadas de dicionários. R é uma linguagem e também um ambiente de desenvolvimento integrado para cálculos estatísticos e gráficos.

017. (FGV/MPE-AL/ANALISTA DO MINISTÉRIO PÚBLICO – DESENVOLVIMENTO DE SISTEMAS/2018) Analise o código Python 2.7 a seguir.

```
L=[10, 12, 14, 16]
for k in range(4, -5,-1):
print L[k]
```

Esse programa causa

a) erro de sintaxe.

b) erro de execução.

c) a exibição de 4 valores, 16,14,12,10, nessa ordem.

d) a exibição de 8 valores, 16,14,12,10,16,14,12,10, nessa ordem.

e) a exibição do valor 16, somente.

018. (QUADRIX/COREN/RS/2018) No que se refere à linguagem de programação Python, assinale a alternativa correta.
a) A Python é uma linguagem de alto nível e robusta. Ela possui seu próprio framework e é incompatível com frameworks de terceiros.
b) A Python utiliza a duck typing (tipagem dinâmica), que nada mais é do que definir um tipo para a variável, com as operações que podem ser aplicadas, antes mesmo de ela ter sido criada, com base em conhecimento prévio do programa. Esta tarefa é executada pelo interpretador.
c) O caractere "/" marca o início de comentário. Qualquer texto depois do "/"será ignorado até o fim da linha.
d) A Python permite que os conteúdos das variáveis sejam sempre alterados, não existindo, dessa forma, tipos imutáveis.
e) Pode ser utilizada como linguagem principal no desenvolvimento de sistemas e também pode ser utilizada como linguagem script em vários softwares.

019. (UERJ/UERJ/ANALISTA DE SISTEMAS/2015) A linguagem Python possui a seguinte característica:
a) é uma linguagem compilada
b) exige declaração de código
c) a tupla é um tipo mutável
d) é orientada a objetos

020. (IESES/IFC-SC/PROGRAMAÇÃO WEB E DISPOSITIVOS MÓVEIS/2015) O conjunto correto de palavras reservadas para a construção de uma estrutura de controle em Python 3.4.3 é:
a) "if", "elsif" e "else".
b) "if", "else if" e "else".
c) Somente "if"; o restante da estrutura de controle ("senão se" e "senão") é realizado simplesmente com indentação.
d) "if", "elif" e "else".

021. (CESPE/SERPRO/ANALISTA – REDES/2010) Julgue os itens que se seguem, a respeito da linguagem Python, versão 3.1.
Se, em qualquer linha do script Python, a expressão regular coding [=:] \s*([~\w.]+) corresponder a um comentário, este será processado como uma declaração de codificação.

022. (FCC/MPE-PE/ANALISTA DE SISTEMAS/2012) Em Python, os métodos de lista permitem utilizar listas como pilhas, onde o item adicionado por último é o primeiro a ser recuperado. Para adicionar um item ao topo da pilha, e para recuperar um item do topo da pilha utilizam-se, respectivamente os métodos:

a) append() e pop().
b) insert() e top().
c) addTop() e pop().
d) add() e get().
e) addItem() e top().

023. (CESPE/SERPRO/ANALISTA DE SISTEMAS/2008) O método capitalize da classe String do Python é utilizado para:
a) remover todos os espaços de uma string.
b) verificar se todos os caracteres da string são numéricos.
c) procurar uma substring em uma string retornando seu índice caso seja encontrada.
d) retornar uma cópia de uma string somente com o primeiro caractere em maiúsculo.
e) retornar uma cópia de uma string com todos os caracteres em minúsculo.

024. (CESGRANRIO/SECAD-TO/ANALISTA DE SISTEMAS/2004) A palavra raise, da linguagem Python, é utilizada para:
a) converter uma string.
b) definir uma classe.
c) gerar uma exceção.
d) imprimir um resultado.
e) incorporar um módulo.

025. (CESGRANRIO/SECAD-TO/ANALISTA DE SISTEMAS/2004) Um programador de Python recebeu a tarefa de criar uma função chamada calcular que recebe dois parâmetros. Para executar sua atividade, ele deve utilizar a expressão:
a) def calcular (a,b):
b) function calcular (a,b):
c) import calcular (a,b):
d) procedure calcular (a,b):
e) sub calcular (a,b):

026. (IESES/IFC-SC/PROGRAMAÇÃO WEB E DISPOSITIVOS MÓVEIS/2015) Sobre listas em Python 3.1.5:
a) list.remove(a) remove o primeiro item da lista cujo valor é a.
b) list.pop(a) adiciona um item de valor a ao início da lista.
c) list.append(a) adiciona um item à lista cujo índice será a.
d) list.index(a) retorna o valor do item cujo índice é a

027. (CETAP/AL-RR/ANALISTA DE SISTEMAS/2010) Sobre a linguagem de programação PYTHON, é INCORRETO destacar que o operador lógico de conjunção (“e”, como em a e b) é &&.

028. (CESPE/SERPRO/ANALISTA – DESENVOLVIMENTO DE SISTEMAS/2008) Com relação às linguagens, julgue os itens a seguir.
Python é uma linguagem livre de alto nível, orientada a objetos e de difícil leitura, pois não permite identação de linhas de código.

029. (CESGRANRIO/BANCO DO BRASIL/ESCRITURÁRIO/2018) O programa a seguir, em Python, implementa o algoritmo do método de bolha, imprimindo o resultado de cada passo.

```
def bolha(lista):
    for passo in range(len(lista)-1,0,-1):
        for i in range(passo):
            if lista[i]>lista[i+1]:
                lista[i],lista[i+1]=lista[i+1],lista[i]
        print(lista)
```

Qual será a quarta linha impressa para a chamada bolha([4, 3, 1, 9, 8, 7, 2, 5])?
a) [3, 1, 4, 8, 7, 2, 5, 9]
b) [1, 3, 4, 7, 2, 5, 8, 9]
c) [1, 2, 3, 4, 5, 7, 8, 9]
d) [1, 3, 2, 4, 5, 7, 8, 9]
e) [1, 3, 4, 2, 5, 7, 8, 9]

030. (CETAP/AL-RR/ANALISTA DE SISTEMAS/2010) Sobre a linguagem de programação PYTHON, marque a alternativa INCORRETA.
a) Python suporta a maioria das técnicas da programação orientada a objetos.
b) Python suporta e faz uso constante de tratamento de exceções como uma forma de testar condições de erro e outros eventos inesperados no programa.
c) As funções são definidas em Phyton utilizando a palavra chave def.
d) A separação de blocos de código em Phyton é feita utilizando a endentação de código.
e) O operador lógico de conjunção (“e”, como em a e b) é &&.

031. (FCC/TRT-MG/TÉCNICO JUDICIÁRIO – TECNOLOGIA DA INFORMAÇÃO/2015) Considere o código fonte Python abaixo.
Para que seja exibido [0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89] a lacuna “I” precisa ser preenchida corretamente com:

```
def calcular(n):
    resultado = []
    a, b = 0, 1
    while a < n:
            I
    return resultado

res = calcular(100)
print res
```

a) resultado.add(a)
a, b = b, a+b
b) resultado.insert(a)
a, b = b, a+b
c) resultado.append(a)
a, b = b, a+b
d) resultado.add(a)
a, b = a, a+b
e) resultado.append(a)
a, b = a+b, b

032. (FGV/TJ-BA/ANALISTA JUDICIÁRIO/2015) Analise o trecho de programa, escrito em Pyhton na versão 2.7, mostrado a seguir.

```
X=[]
for i in range(10,1,-1):
    X.append(i)
print X[3:]
```

O trecho exibe:
a) [7, 6, 5]
b) [7]
c) []
d) [7, 6, 5, 4, 3, 2]
e) [7, 6, 5, 4, 3, 2, 1]

033. (IESES/IFC-SC/INFORMÁTICA/PROGRAMAÇÃO WEB E DISPOSITIVOS MÓVEIS/2015) Analise o seguinte trecho de código em Python 3.3.2 e assinale a alternativa correta:

```
class Cachorro:
        def tipo(self):
                return "10"

class Basset(Cachorro):
        def tipo(self):
                return "100"

cachorro1 = Cachorro()
cachorro2 = Basset()

print(cachorro1.tipo())
print(cachorro2.tipo())
```

a) Ocorrerá um erro de execução na linha "def tipo(self):" logo abaixo de "class Basset(Cachorro):".
b) A execução de "print(cachorro2.tipo())" mostrará o valor "10" na saída padrão.
c) A execução de "print(cachorro2.tipo())" mostrará o valor "100" na saída padrão.
d) Ocorrerá um erro de execução na linha "print(cachorro2.tipo())".

034. (PAQTCPB/UEPB/ANALISTA DE SISTEMAS/2012) No comando de atribuição em Python valor = raw_input("Digite um valor:"), qual o tipo da variável valor?

a) str

b) bool

c) int

d) float

e) file

035. (FCC/CNMP/DESENVOLVIMENTO DE SISTEMAS/2015) Considere os fragmentos de programas Phyton a seguir:

Fragmento 1:

```
for n in range(2, 10):
  for x in range(2, n):
      if n % x == 0:
        print n, '=', x, '*', n/x
          break
  else:
     print n, 'é um número primo'
```

Fragmento 2:

```
a = ['Casa', 'Mala', 'Prova']
for x in a:
      print x, len(x)
```

É correto afirmar que:

a) o Fragmento 1 está incorreto, pois laços não podem ter uma cláusula else.

b) no Fragmento 2, a instrução for está incorreta, pois ela não pode iterar sobre a.

c) o Fragmento 1 está incorreto, pois não é possível iterar sobre sequências numéricas utilizando a função range.

d) no Fragmento 1 é verificado se o quociente da divisão de n por x corresponde a 0.

e) os dois fragmentos de código estão corretos.

036. (FGV/TJ-BA/ANALISTA JUDICIÁRIO/2015) Analise o trecho de programa Python, na versão 2.7, apresentado a seguir.

```
L=[1,2,3,4,5,6,7,8]
print L[::-1]
```

Ao ser executado, o resultado exibido é:

a) [1, 2, 3, 4, 5, 6, 7, 8]

b) [8]

c) [ ]

d) [8, 7, 6, 5, 4, 3, 2, 1]

e) [1]

037. (UERJ/UERJ/ANALISTA DE SISTEMAS/GRID/2015) Considere o trecho do programa Python abaixo:

```
def dobra (y):
x = y + y
return x
x = 5
dobra(x)
dobra(x)
print x
```

O valor impresso ao executarmos o programa é:

a) 5
b) 10
c) 15
d) 25

038. (CESPE/CORREIOS/ANALISTA DE CORREIOS/ANALISTA DE SISTEMAS – DESENVOLVIMENTO DE SISTEMAS/2011) Com relação aos sistemas de suporte a decisão e gestão de conteúdo, julgue os seguintes itens.

A linguagem Python e seu interpretador estão disponíveis para as mais diversas plataformas. Para que seja usado em determinado sistema operacional não suportado, é possível gerar o Python a partir do programa fonte utilizando um compilador C. Nesse caso, o código fonte é traduzido para o formato bytecode, que é multiplataforma e pode ser distribuído de forma independente.

039. (FGV/TÉCNICO DA PROCURADORIA (PGE RO)/TECNOLOGIA DA INFORMAÇÃO/2015) Na linguagem **Python 2.7**, os comandos

```
L=range(0,12)
print L
```

produzem:

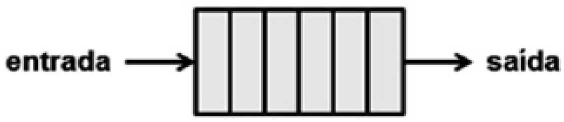
a) [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
b) [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]
c) [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]
d) [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 12]
e) uma mensagem de erro

040. (CESPE/SERPRO/ANALISTA – SUPORTE TÉCNICO/2010) Acerca das linguagens de programação, julgue os itens subsequentes.

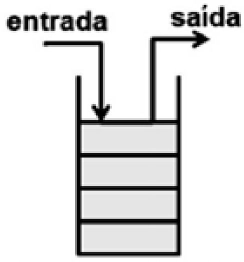
As linguagens Perl e Python são dinamicamente tipadas, ou seja, o tipo da variável é definido em tempo de execução. As linguagens C e Java são estaticamente tipadas, ou seja, o tipo de variável é definido em tempo de compilação.

LÓGICA DE PROGRAMAÇÃO

041. (FUNCAB/PRODAM – PROCESSAMENTO DE DADOS AMAZONAS S.A /PROGRAMADOR-DESENVOLVEDOR/2014) Observe as figuras I e II, que representam duas estruturas de dados.



I - FIFO – "First In First Out"



II - LIFO – "Last In First Out"

Essas estruturas de dados são denominadas respectivamente:

a) fila e pilha.
b) vetor e lista.
c) fila e vetor.
d) vetor e pilha.
e) fila e lista.

042. (IBFC/PC-RJ/PERITO CRIMINAL/ENGENHARIA DA COMPUTAÇÃO/2013) Marque a opção que determina uma lista de procedimentos bem definida, que pega algum valor, ou conjunto de valores como entrada, e produz algum valor ou conjunto de valores como saída.

a) Algoritmo.
b) Código.
c) Chave
d) Inserção.
e) Loop.

043. (VUNESP/CTA/TÉCNICO EM INFORMÁTICA/2013) Em linguagem de programação, uma variável é:

a) o resultado de uma expressão aritmética.
b) o nome dado às informações salvas no disco.
c) um número, uma letra ou um ponto-flutuante.
d) uma posição de memória identificada.
e) uma palavra especial utilizada pela linguagem para identificar suas instruções de controle.

044. (FCC/TRT – 9ª REGIÃO (PR)/ANALISTA JUDICIÁRIO – TECNOLOGIA DA INFORMAÇÃO/2013) Considere o algoritmo em pseudo linguagem:

```
início
caracter: nome, sexo;
    imprima ("Qual é o seu nome? ");
    leia(nome);
    imprima ("Qual é o seu sexo? (f/m) ");
    leia (sexo);

    se (sexo = 'f' E sexo = 'F')
    então imprima ("Você é do sexo feminino. ");
    senão
        se (sexo = 'm' E sexo = 'M')
            então imprima ("Você é do sexo masculino. ");
        senão
            imprima ("Você digitou um valor de sexo invalido ");
        fim se;
    fim se;
fim.
```

Sobre o algoritmo acima é correto afirmar que

a) a lógica do algoritmo está comprometida pela falta de um comando de repetição.

b) em vez de utilizar comandos de decisão **se** aninhados, deveria ter sido usado um único comando de seleção múltipla, por isso a lógica ficou comprometida.

c) se for digitada uma letra maiúscula 'F' ou minúscula 'f' será impresso **Você é do sexo feminino.**

d) somente se a letra digitada para o sexo for diferente de 'F', 'f', 'M', 'm' é que a frase **Você digitou um valor de sexo inválido** será impressa.

e) não importa o valor digitado no sexo, pois sempre será impresso **Você digitou um valor de sexo inválido**.

GABARITO

1. e
2. E
3. C
4. E
5. a
6. a
7. d
8. a
9. d
10. C
11. c
12. E
13. c
14. b
15. b
16. E
17. a
18. e
19. d
20. d
21. E
22. a
23. d
24. c
25. a
26. a
27. C
28. E
29. d
30. e
31. c
32. d
33. c
34. a
35. e
36. d
37. a
38. C
39. b
40. C
41. a
42. a
43. d
44. e

GABARITO COMENTADO

PYTHON

007. (CESPE (CEBRASPE)/DPE-RO/ANALISTA DA DEFENSORIA PÚBLICA – PROGRAMAÇÃO/2022) Na linguagem Python, são consideradas sequências mutáveis as

a) strings.
b) cadeias.
c) tuplas.
d) listas.
e) ranges.

a) Errada. ***String*** é uma sequência de caracteres geralmente utilizada para representar caracteres, palavras, frases ou textos.
b) Errada. Uma **string** é uma **cadeia de caracteres variáveis de tamanho dinâmico.**
c) Errada. **Tuplas:** trata-se de uma coleção de valores ordenados, imutáveis e indexáveis que pode conter valores duplicados (também podem ser chamadas de sequências).
d) Certa. Uma **coleção** é uma estrutura de dados utilizada para armazenar objetos. Em Python, **existem quatro tipos de coleções**, que são:

List (Lista)	**É uma coleção ordenada e modificável. Permite membros duplicados. É indexada por inteiro.**
Tuple (Sequência)	É uma **coleção ordenada** e **imutável**. Permite membros duplicados. É indexada por inteiro.
Set (Conjunto)	É uma **coleção desordenada**, alterável e não indexada. Nenhum membro duplicado.
Dictionary (Dicionário)	É uma **coleção desordenada**, alterável e indexada (pode ser por *string*). Nenhum membro duplicado.

Conforme visto, a **lista** é considerada sequência mutável.
e) Errada. **Range** tem vários significados: variedade, sequência de números representados entre parênteses, etc.
Letra d.

---

008. (CESPE (CEBRASPE)/PC PB/ESCRIVÃO DE POLÍCIA/2022) Na linguagem Python, o tipo de uma variável em tempo de execução é definido pelo interpretador pelo recurso denominado

a) tipagem dinâmica.
b) modo interativo.
c) sintaxe.
d) interpretação bytecode.
e) empacotamento.

Python é **uma linguagem com verificação de tipos**, mas **tipada dinamicamente (com tipagem dinâmica),** o que significa que o próprio interpretador infere os tipos de dados **SEM** a necessidade de o desenvolvedor informar.

Veja a seguir as **características de tipagem** das variáveis **Python**:

Tipagem forte	**Não permite fazer operações com tipos que sejam incompatíveis**
Tipagem dinâmica	**O tipo de variável pode mudar ao longo do programa.**

Letra a.

---

009. (CESPE (CEBRASPE)/PC PB/PERITO OFICIAL – CRIMINAL – TECNOLOGIA DA INFORMAÇÃO/2022) Python é uma linguagem procedural que utiliza quatro tipos de dados predefinidos para lidar com coleções: conjuntos, dicionários, listas e tuplas. A respeito desses tipos de dados, julgue os itens a seguir.

I – O conjunto permite o armazenamento de uma tupla, mas não o de uma lista.

II – A tupla é idêntica à lista, exceto pela forma mais simples com que sua declaração é realizada.

III – A lista é um tipo de dados variável que permite a alteração de seus elementos após a sua criação.

Assinale a opção correta.

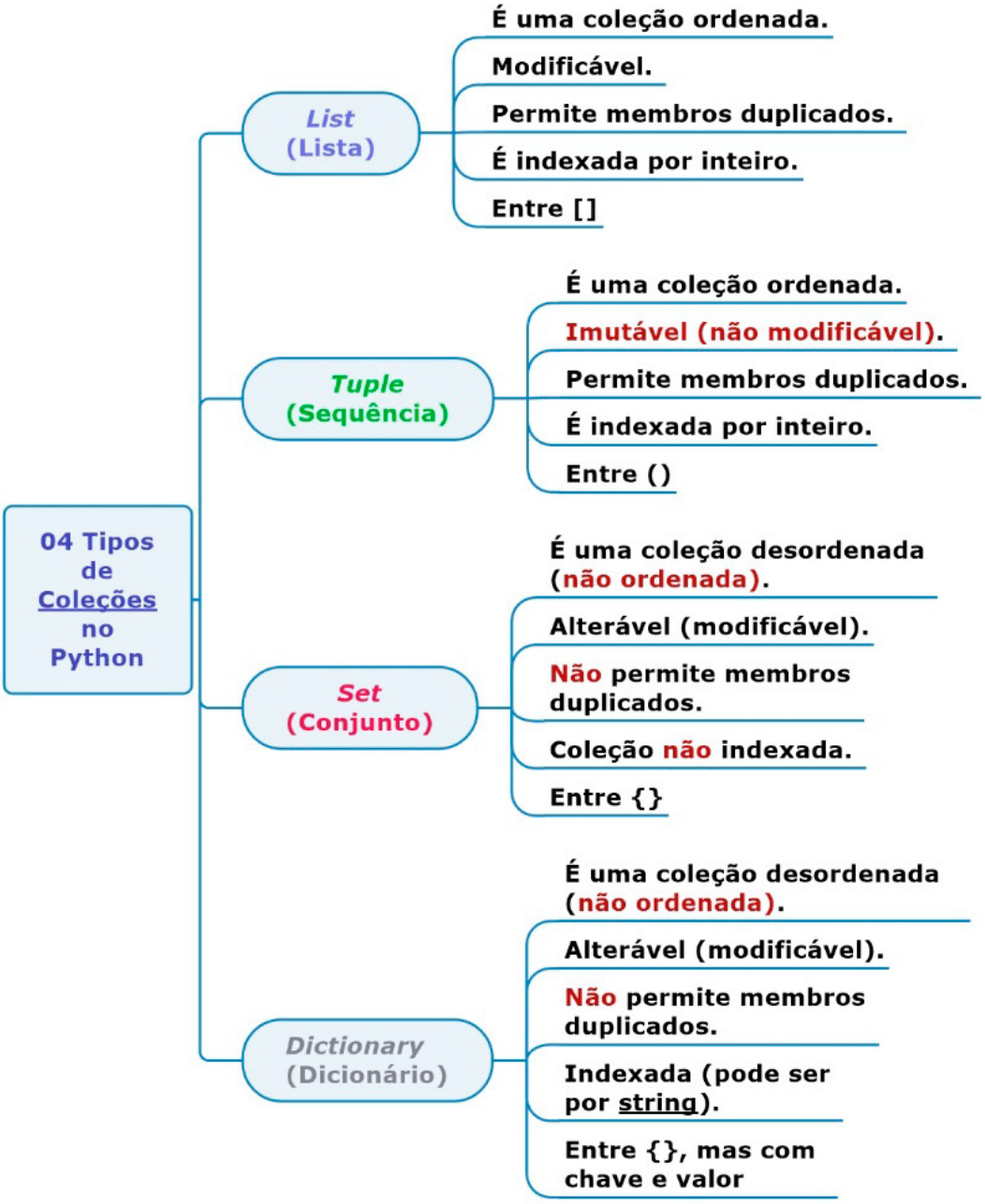
a) Apenas o item I está certo.

b) Todos os itens estão certos.

c) Apenas o item II está certo.

d) Apenas os itens I e III estão certos.

e) Apenas os itens II e III estão certos.



I – Certa. O **conjunto** (*set*) permite o armazenamento de uma tupla (não modificável), mas não o de uma lista (listas são modificáveis). O conjunto pode conter apenas objetos imutáveis, como strings, ints, floats e tuplas.

II – Errada. **Tuple** (sequência) é uma coleção que é ordenada e imutável. Permite membros duplicados. É indexada por inteiro. Em Python, tuplas são escritas entre (). Uma **lista** (List) é uma coleção ordenada e modificável. Em Python, listas são escritas com colchetes [].

III – Certa. A lista é um tipo de dados variável que permite a alteração de seus elementos após a sua criação.

Logo, apenas itens I e III estão corretos.

Letra d.

010. (CESPE (CEBRASPE)/SERPRO/ANALISTA – ESPECIALIZAÇÃO: CIÊNCIA DE DADOS/2021)
A respeito da linguagem de programação Python, julgue o item a seguir.
Listas são coleções alteráveis de qualquer tipo de objeto — como, por exemplo, outras listas — capazes de gerar um efeito top-down sem limite de níveis.

Uma lista (List) é uma coleção ordenada e modificável. Em Python, listas são escritas com colchetes [].
Veja a seguir os quatro tipos de coleções existentes no Python:

Certo.

011. (CESPE (CEBRASPE)/SEED-PR/PROFESSOR – EDUCAÇÃO BÁSICA E JORNADA/2021) Na linguagem de programação Python, existem 3 estruturas para armazenar dados indexados. A estrutura cujos valores são imutáveis depois de sua criação é conhecida como

a) lista.

b) operador.

c) tupla.

d) classe.

e) dicionário.

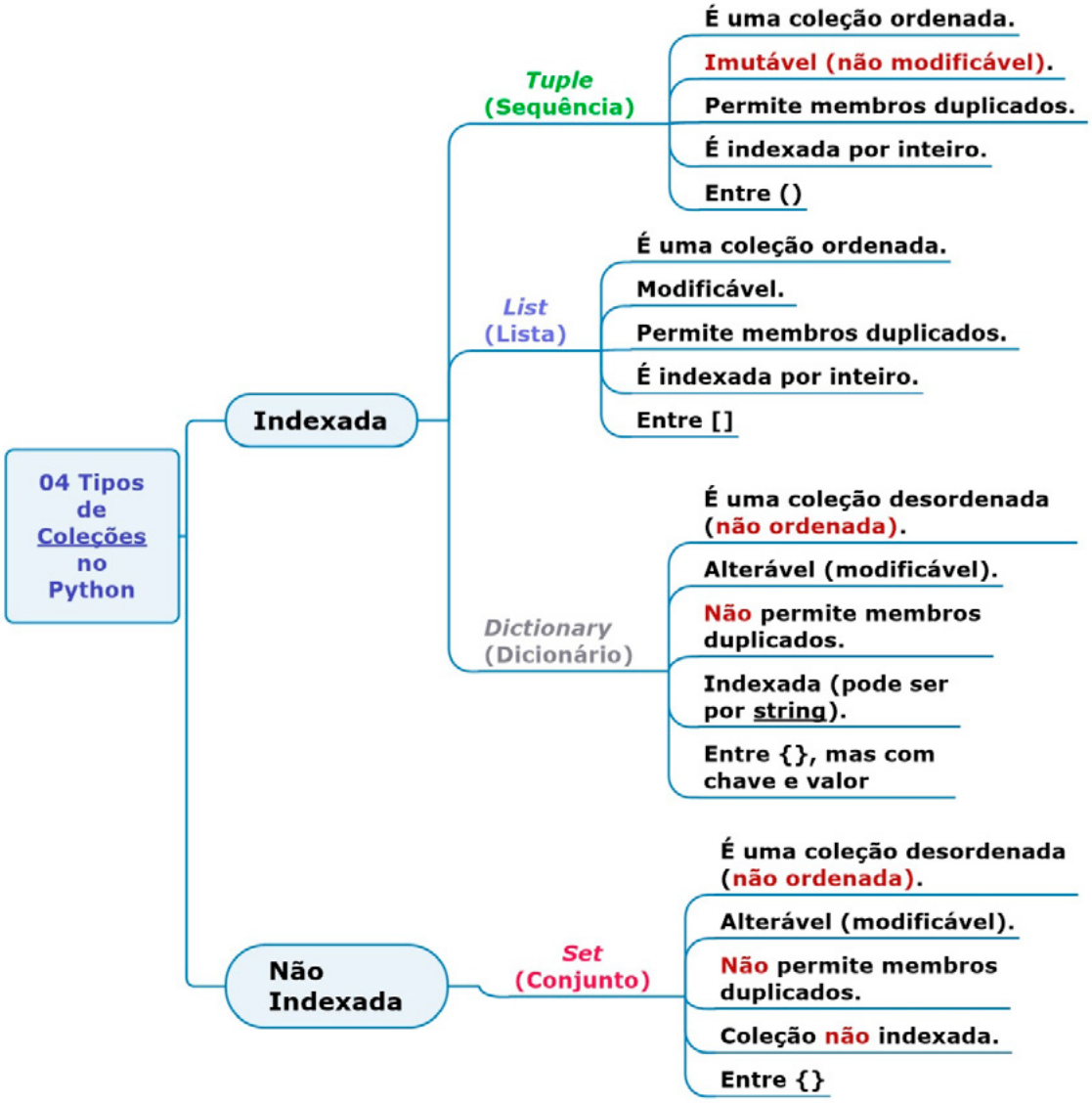


Figura. 4 Tipos de Coleções no Python (QUINTÃO, 2023)

Na linguagem de programação Python, existem 3 estruturas para armazenar dados indexados, que são: sequências, listas e dicionários. Dentre essas estruturas, a que é imutável é conhecida como tupla.

Letra c.

012. (CEBRASPE(CESPE)/PF/AGENTE DE POLÍCIA FEDERAL/2018) Julgue o item, relativo a noções de programação Python e R.

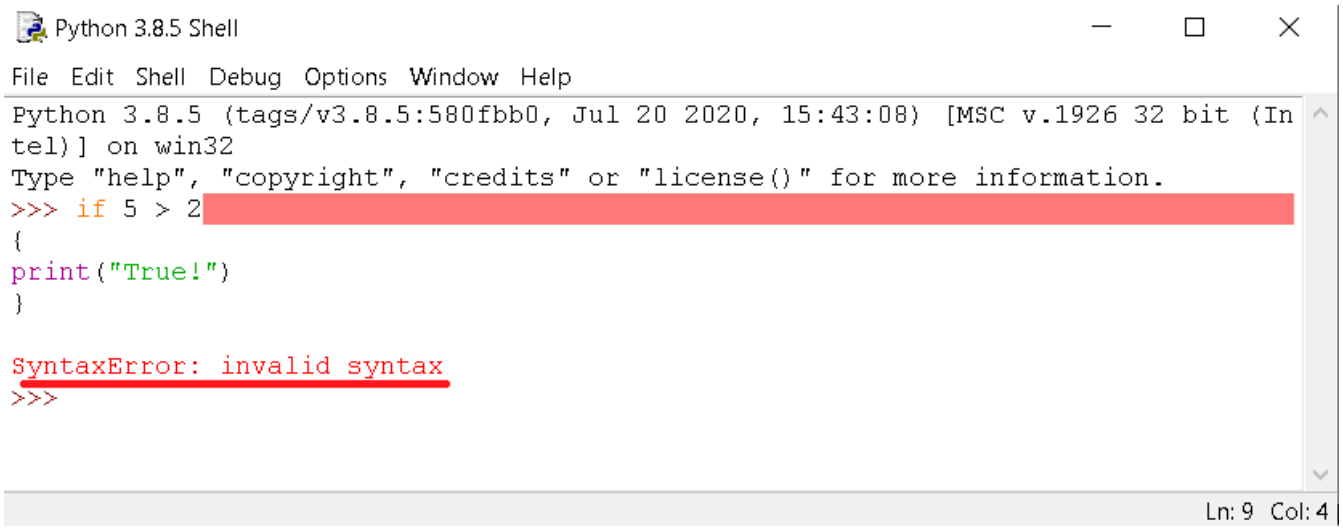
Considere o programa a seguir, na linguagem Python.

```
if 5 > 2
{
print(“True!”)
}
```

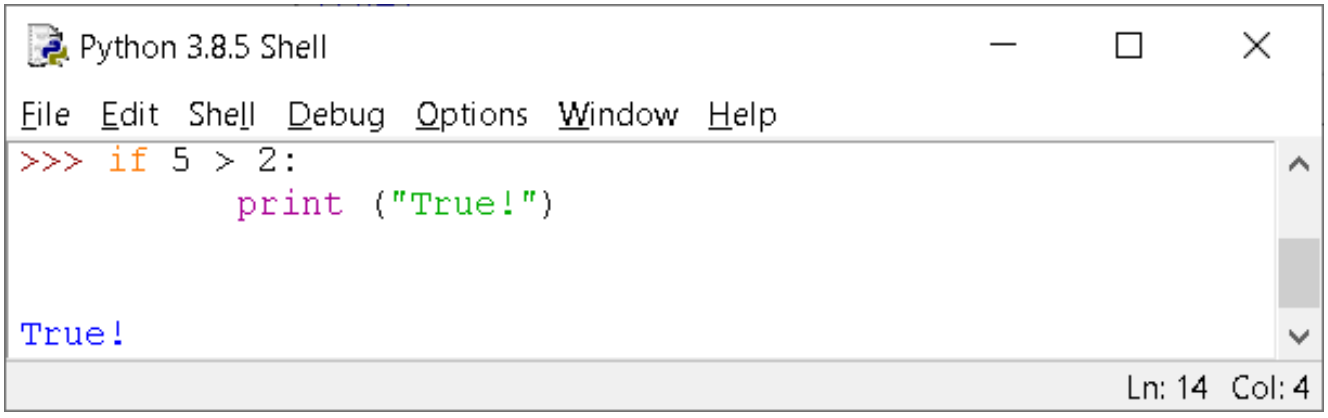
A sintaxe do programa está correta e, quando executado, ele apresentará o seguinte resultado.

```
True!
```

A sintaxe utilizada está incorreta, pois possui erro de sintaxe, em virtude de utilização indevida das chaves. Também não foi inserida a indentação do código. Vide o erro destacado na figura seguinte.



A figura seguinte destaca o código como deveria ser reescrito.



Errado.

013. (FCC/TRE-CE/TÉCNICO JUDICIÁRIO – PROGRAMAÇÃO DE SISTEMAS/2012) Sobre Python é correto afirmar:

a) É uma linguagem compilada, ou seja, o código-fonte de um programa é lido pelo compilador, que cria um arquivo binário, executável diretamente pelo hardware.

b) É uma linguagem fortemente tipada, ou seja, é preciso declarar variáveis e seus tipos.

c) Suporta funcionalidades comuns na orientação a objetos: herança, herança múltipla, polimorfismo, reflexão e introspecção.

d) Uma lista em Python é um conjunto de valores acessados por um índice numérico, inteiro, começando em 1. Assim como em outras linguagens, a lista pode armazenar apenas valores de um mesmo tipo.

e) Uma String Python é uma sequência imutável, alocada estaticamente, com restrição de tamanho.

a) Errada. **Python é uma linguagem interpretada e pseudo-compilada**, o que significa que um código-fonte escrito em Python é executado pelo interpretador e, em seguida, executado pelo sistema operacional.

b) Errada. **Python é uma linguagem com verificação de tipos**, mas **tipada dinamicamente,** o que significa que o próprio interpretador infere os tipos de dados sem a necessidade de o desenvolvedor informar.

c) Certa. **Tudo na linguagem Python é um objeto, suportando funcionalidades comuns na orientação a objetos**.

d) Errada. **Listas** são um conjunto de objetos, não necessariamente do mesmo tipo. **O endereçamento dos componentes começa no 0**. Então, se você deseja usar um determinado elemento da lista, basta chamá-lo: lista[x]. Com isso, você estará se referindo ao elemento de posição x na lista.

e) Errada. ***Strings* em Python são imutáveis**, no entanto, não há restrição de tamanho. Uma ***string* é uma cadeia de caracteres variáveis de tamanho dinâmico**.

Letra c.

---

014. (FGV/CM/CARUARU/ANALISTA LEGISLATIVO – INFORMÁTICA/2015) Analise o código Python a seguir.

```
L1=[10,20,30]
L2=[40,50]
L1.append(L2)
print L1
```

Assinale a opção que descreve corretamente o que acontece quando esse programa é executado no Python 2.7.

a) Produz uma mensagem de erro, porque tenta executar uma operação inválida.
b) Exibe "[10, 20, 30, [40, 50]]".
c) Exibe "[10, 20, 30, 40, 50]".
d) Exibe "[10, 20, 30], [40, 50]".
e) Exibe "[]".

O comando L1.**append**(L2) irá inserir o conteúdo de L2 ao final de L1.
Mas, observe que o método **append** inclui a lista L2 como se fosse só um elemento, assim L2 é tratado como se fosse uma coisa só nesse contexto!
Teremos então L1=[10,20,30,[40,50]] e essa lista final possui 4 elementos ao invés de 5.
Cuidado com essa pegadinha, a grande maioria dos concurseiros marcaria a letra C como resposta, caso não tivesse prestado atenção!
Quando esse programa for executado, então exibirá o valor de L1 que é [10,20,30,[40,50]].
Letra b.

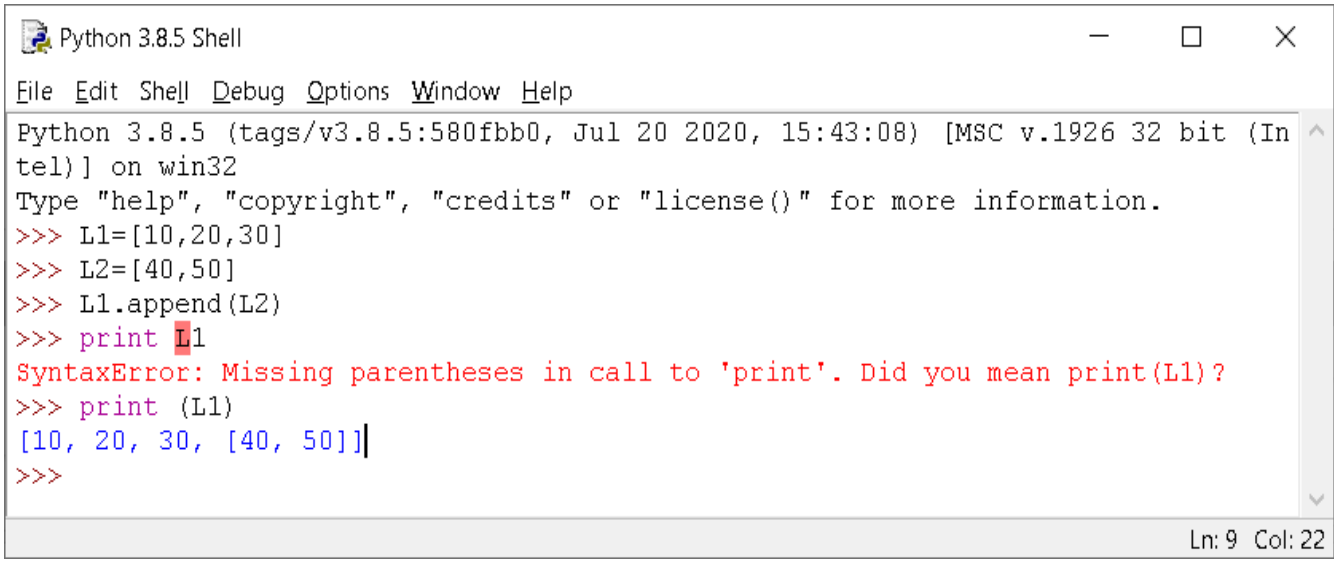
---

015. (FGV/CM/CARUARU/ANALISTA LEGISLATIVO – INFORMÁTICA/2015/ADAPTADA) Analise o código Python a seguir.

```
L1=[10,20,30]
L2=[40,50]
L1.append(L2)
print (L1)
```

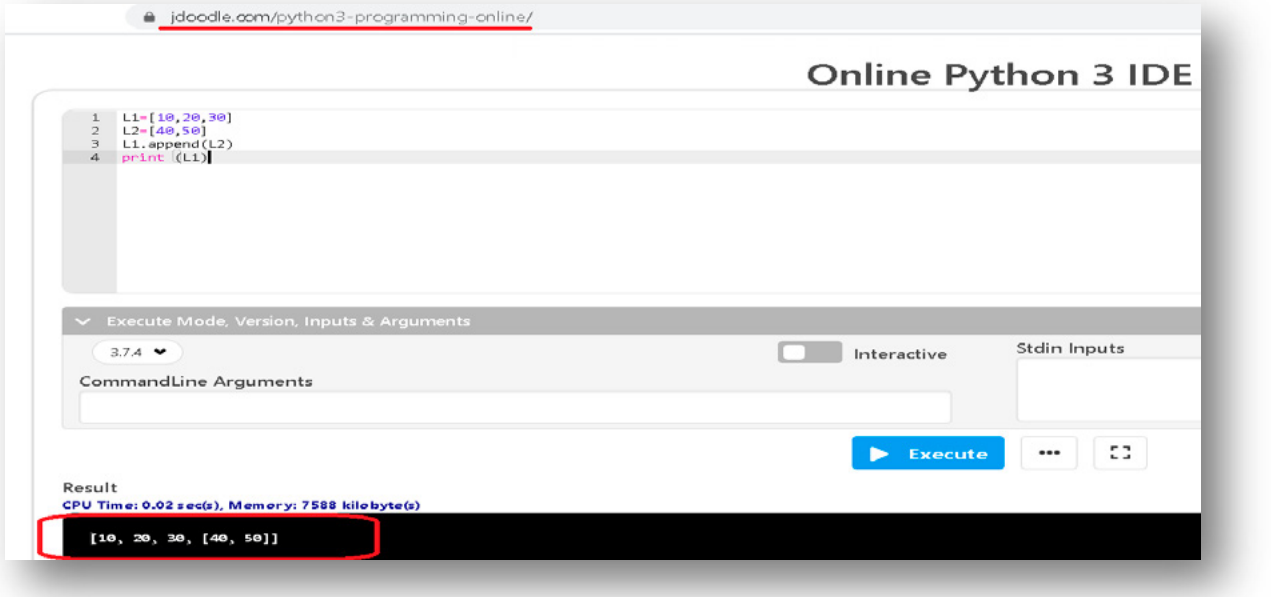
Assinale a opção que descreve corretamente o que acontece quando esse programa é executado no Python 3.7.4.
a) Produz uma mensagem de erro, porque tenta executar uma operação inválida.
b) Exibe "[10, 20, 30, [40, 50]]".
c) Exibe "[10, 20, 30, 40, 50]".
d) Exibe "[10, 20, 30], [40, 50]".
e) Exibe "[]".

O **print()** é uma **função do Python utilizada para imprimir alguma mensagem na tela**. **O uso de parênteses no comando PRINT se tornou** obrigatório **a partir da versão 3.X do Python**. Observe na figura seguinte o erro reportado pela falta de parênteses. Após correção, tivemos sucesso!



O comando L1.append(L2) irá inserir o conteúdo de L2 ao final de L1. Mas, observe que o método **append** inclui a lista L2 como se fosse só um elemento, assim L2 é tratado como se fosse uma coisa só nesse contexto! Teremos então L1=[10,20,30,[40,50]] e essa lista final possui 4 elementos ao invés de 5. Cuidado com essa pegadinha, a grande maioria dos concurseiros marcaria a letra C como resposta, caso não tivesse prestado atenção!

Quando esse programa for executado, então exibirá o valor de L1 que é [10,20,30,[40,50]].



Letra b.

---

016. (INÉDITA/2023) Em vez de vetores, Python inclui somente dois tipos de estrutura de dados: listas e dispersões, chamadas de dicionários. R é uma linguagem e também um ambiente de desenvolvimento integrado para cálculos estatísticos e gráficos.

Em vez de vetores, **Python inclui os seguintes tipos de estrutura de dados**: **listas**; **listas imutáveis**, chamadas de **tuplas**; **dispersões**, chamadas de **dicionários,** e **conjuntos**.

R é uma **linguagem orientada a objetos muito usada para análise e manipulação de dados estatísticos e também pode ser considerada um** ambiente **de desenvolvimento integrado para cálculos estatísticos e gráficos.**
Errado.

---

017. (FGV/MPE-AL/ANALISTA DO MINISTÉRIO PÚBLICO – DESENVOLVIMENTO DE SISTEMAS/2018) Analise o código Python 2.7 a seguir.

L=[10, 12, 14, 16]

for k in range(4, -5,-1):

print L[k]

Esse programa causa

a) erro de sintaxe.

b) erro de execução.

c) a exibição de 4 valores, 16,14,12,10, nessa ordem.

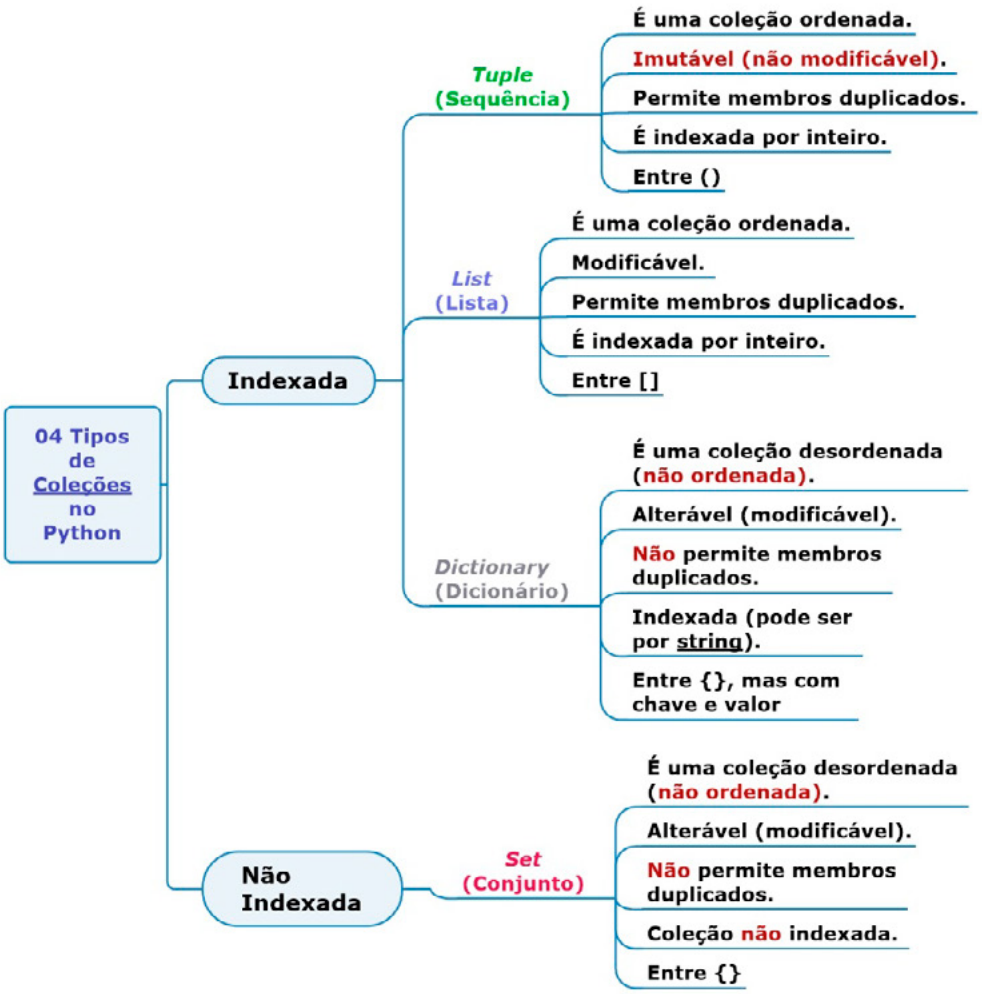
d) a exibição de 8 valores, 16,14,12,10,16,14,12,10, nessa ordem.
e) a exibição do valor 16, somente.

Observe que a **indentação do código não** foi respeitada, ocasionando **erro de sintaxe**. O for abriu um bloco (:), no entanto não possui nenhum conteúdo. O print está fora do for.
Assim, é retornado um erro de sintaxe, e a letra A é a resposta.
Letra a.

---

018. (QUADRIX/COREN/RS/2018) No que se refere à linguagem de programação Python, assinale a alternativa correta.
a) A Python é uma linguagem de alto nível e robusta. Ela possui seu próprio framework e é incompatível com frameworks de terceiros.
b) A Python utiliza a duck typing (tipagem dinâmica), que nada mais é do que definir um tipo para a variável, com as operações que podem ser aplicadas, antes mesmo de ela ter sido criada, com base em conhecimento prévio do programa. Esta tarefa é executada pelo interpretador.
c) O caractere "/" marca o início de comentário. Qualquer texto depois do "/"será ignorado até o fim da linha.
d) A Python permite que os conteúdos das variáveis sejam sempre alterados, não existindo, dessa forma, tipos imutáveis.
e) Pode ser utilizada como linguagem principal no desenvolvimento de sistemas e também pode ser utilizada como linguagem script em vários softwares.

a) Errada. A Python é uma linguagem de alto nível e robusta. Ela possui seu próprio framework, que é incompatível com frameworks de terceiros.
b) Errada. A Python utiliza a duck typing (tipagem dinâmica), o que significa que o próprio interpretador infere os tipos de dados SEM a necessidade de o desenvolvedor informar. Assim não é necessário definir um tipo para variável antes de ela ser criada – isso pode ser inferido pelo interpretador.
c) Errada. O caractere "#" marca o início de comentário. Qualquer texto depois do "#"será ignorado até o fim da linha.
d) Errada. Os tipos podem ser **mutáveis** ou **imutáveis (não modificável)**.



e) Certa. **Python** pode ser utilizada como linguagem principal no desenvolvimento de sistemas e também pode ser utilizada como linguagem *script* em vários softwares. Com o Python podemos criar *scripts* para automatizar diversas tarefas repetitivas, por exemplo.
Letra e.

---

019. (UERJ/UERJ/ANALISTA DE SISTEMAS/2015) A linguagem Python possui a seguinte característica:

a) é uma linguagem compilada
b) exige declaração de código
c) a tupla é um tipo mutável
d) é orientada a objetos

a) Errada. **Python é uma linguagem interpretada**, não compilada.
b) Errada. **Python é uma linguagem com verificação de tipos**, mas **tipada dinamicamente,** o que significa que o próprio interpretador infere os tipos de dados sem a necessidade de o desenvolvedor informar.

c) Errada. A **tupla** é equivalente a uma **lista imutável**, seu valor não pode ser alterado.
d) Certa. **Python é uma linguagem de *scripting* orientada a objetos, interpretada** e relativamente recente (início dos anos 90).
Letra d.

---

020. (IESES/IFC-SC/PROGRAMAÇÃO WEB E DISPOSITIVOS MÓVEIS/2015) O conjunto correto de palavras reservadas para a construção de uma estrutura de controle em Python 3.4.3 é:
a) "if", "elsif" e "else".
b) "if", "else if" e "else".
c) Somente "if"; o restante da estrutura de controle ("senão se" e "senão") é realizado simplesmente com indentação.
d) "if", "elif" e "else".

A **linguagem Python** é uma linguagem de alto nível construída para trabalhar com o conceito de **produtividade** e **legibilidade**.
A legibilidade visa tornar simples, porém eficaz, a escrita do código. Nas estruturas de controle condicional, por exemplo, a forma correta na escrita é **if** e **else**. Caso haja condições intermediárias usa-se **elif**.

EXEMPLO

```
1 idade = int(input('Digite sua idade: '))
2 if idade >= 10 and idade < 20:
3       print('Você é adolescente')
4 elif idade >= 20 and idade < 30:
5       print('Você é jovem')
6 elif idade >= 30 and idade <= 100:
7       print('Você é adulto')
8 else:
9       print('idade não encontrada!')
```

Letra d.

---

021. (CESPE/SERPRO/ANALISTA – REDES/2010) Julgue os itens que se seguem, a respeito da linguagem Python, versão 3.1.
Se, em qualquer linha do script Python, a expressão regular coding [=:] \s*([~\w.]+) corresponder a um comentário, este será processado como uma declaração de codificação.

Os **comentários** incluídos nas linhas dos *scripts* Python não são processados, uma vez que correspondem a **notas explicativas** que têm o objetivo de descrever algo que se tenha necessidade, para melhor organizar os códigos.

Errado.

---

022. (FCC/MPE-PE/ANALISTA DE SISTEMAS/2012) Em Python, os métodos de lista permitem utilizar listas como pilhas, onde o item adicionado por último é o primeiro a ser recuperado. Para adicionar um item ao topo da pilha, e para recuperar um item do topo da pilha utilizam-se, respectivamente os métodos:

a) append() e pop().
b) insert() e top().
c) addTop() e pop().
d) add() e get().
e) addItem() e top().

Na **função append()** **o elemento é adicionado no final da lista ou no topo de uma pilha**, enquanto a **função pop()** é utilizada para **recuperar um item de uma lista ou do topo de uma pilha**.

Letra a.

---

023. (CESPE/SERPRO/ANALISTA DE SISTEMAS/2008) O método capitalize da classe String do Python é utilizado para:

a) remover todos os espaços de uma string.
b) verificar se todos os caracteres da string são numéricos.
c) procurar uma substring em uma string retornando seu índice caso seja encontrada.
d) retornar uma cópia de uma string somente com o primeiro caractere em maiúsculo.
e) retornar uma cópia de uma string com todos os caracteres em minúsculo.

A linguagem **Python** é uma **linguagem de alto nível, orientada a objetos, e possui classes, métodos, herança**, etc. No Python há uma classe chamada String que possui vários métodos capazes de trabalhar com informações de texto (*strings*). O **método capitalize tem como ação transformar o primeiro caractere da *string* em maiúscula.**

EXEMPLO

info = 'o Brasil é um país ocidental'
info.capitalize()
'O Brasil é um país ocidental'

Outros exemplos:

1) Separar onde houver espaço:

EXEMPLO

info.split( )

'O', 'Brasil', 'é', 'um', 'país', 'ocidental'

2) Substituir uma palavra por outra:

EXEMPLO

info.replace('ocidental', 'tropical')

'O Brasil é um país tropical'

3) Retorna o número de ocorrências de substring

EXEMPLO

info.count('s')

Letra d.

---

024. (CESGRANRIO/SECAD-TO/ANALISTA DE SISTEMAS/2004) A palavra raise, da linguagem Python, é utilizada para:

a) converter uma string.

b) definir uma classe.

c) gerar uma exceção.

d) imprimir um resultado.

e) incorporar um módulo.

Na **linguagem Python** a sentença **raise** é usada para **provocar uma exceção**, que pode ser **tratada pelo código que chamou aquela parte do programa.**

EXEMPLO

No trecho de código abaixo na linguagem Python há uma verificação condicional para o valor da variável y. Caso y assuma o valor zero (0), uma exceção (ZeroDivisionError) é invocada neste ponto do código e o desenvolvedor tem a opção de fazer o tratamento no próprio trecho do algoritmo, impedindo a divisão por zero.

```
1 z, y = 1, 0
2 if y == 0:
3 raise ZeroDivisionError
4 else:
5 print '%d dividido por %d é %d' % (z, y, z/y)
```

Letra c.

025. (CESGRANRIO/SECAD-TO/ANALISTA DE SISTEMAS/2004) Um programador de Python recebeu a tarefa de criar uma função chamada calcular que recebe dois parâmetros. Para executar sua atividade, ele deve utilizar a expressão:

a) def calcular (a,b):

b) function calcular (a,b):

c) import calcular (a,b):

d) procedure calcular (a,b):

e) sub calcular (a,b):

Uma **função** é um **bloco de comandos que realiza uma ação para quem a chama**. É possível passar dados, chamados de parâmetros ou argumentos, dentro de uma função que também pode retornar valores para quem a chamou.

Na linguagem **Python**, há muitas **funções** previamente construídas que podem ser usadas, entretanto, é permitido ao desenvolvedor a definição ou criação de uma função devendo usar a **cláusula DEF**.

EXEMPLO

A questão propõe a criação de uma função chamada "**calcular**" que recebe dois valores *a* e *b*. No exemplo, a função **calcular** (linhas 1 a 3) é definida recebendo os valores como argumentos e realiza a soma destes. Esta função retorna, por meio do comando **return** (linha 3), o resultado da operação. Na linha 5, a função é chamada passando os valores 3 e 6 que devolve o resultado imprimindo na tela.

```
1 def calcular(a,b):
2 result=a+b
3 return result
4
5 print(calcular(3,6))
```

Letra a.

---

026. (IESES/IFC-SC/PROGRAMAÇÃO WEB E DISPOSITIVOS MÓVEIS/2015) Sobre listas em Python 3.1.5:

a) list.remove(a) remove o primeiro item da lista cujo valor é a.

b) list.pop(a) adiciona um item de valor a ao início da lista.

c) list.append(a) adiciona um item à lista cujo índice será a.

d) list.index(a) retorna o valor do item cujo índice é a

Na linguagem Python, uma **lista** é representada como uma **sequência de objetos (elementos)** e dispõe de vários **métodos** e **operadores** para sua manipulação. **Cada elemento da lista**

contém um índice que começa em zero (0). A seguir destacamos a funcionalidade de cada método proposto na questão:

list.remove()	**Remove um item da lista** em que o valor é passado como argumento. **Caso haja valores iguais o método remove o primeiro deles**.
list.pop()	**Remove e retorna o último item da lista.**
list.append()	**Acrescenta o elemento passado como parâmetro à lista.**
list.index()	**Retorna o índice do elemento passado como parâmetro.**

Letra a.

---

027. (CETAP/AL-RR/ANALISTA DE SISTEMAS/2010) Sobre a linguagem de programação PYTHON, é INCORRETO destacar que o operador lógico de conjunção (“e”, como em a e b) é &&.

Na linguagem Python, o operador lógico de conjunção (“e”) é “**and**” e não “&&”.
Veja trecho de código

EXEMPLO

```
1 idade = int(input('Digite sua idade: '))
2 if idade >= 10 and idade < 20:
3       print('Você é adolescente')
```

Certo.

---

028. (CESPE/SERPRO/ANALISTA – DESENVOLVIMENTO DE SISTEMAS/2008) Com relação às linguagens, julgue os itens a seguir.
Python é uma linguagem livre de alto nível, orientada a objetos e de difícil leitura, pois não permite identação de linhas de código.

Python é uma linguagem livre de alto nível, orientada a objetos, de **leitura simples**, e que permite **indentação** de linhas de código.

Errado.

---

029. (CESGRANRIO/BANCO DO BRASIL/ESCRITURÁRIO/2018) O programa a seguir, em Python, implementa o algoritmo do método de bolha, imprimindo o resultado de cada passo.

```
def bolha(lista):
    for passo in range(len(lista)-1,0,-1):
        for i in range(passo):
            if lista[i]>lista[i+1]:
                lista[i],lista[i+1]=lista[i+1],lista[i]
    print(lista)
```

Qual será a quarta linha impressa para a chamada bolha([4, 3, 1, 9, 8, 7, 2, 5])?

a) [3, 1, 4, 8, 7, 2, 5, 9]

b) [1, 3, 4, 7, 2, 5, 8, 9]

c) [1, 2, 3, 4, 5, 7, 8, 9]

d) [1, 3, 2, 4, 5, 7, 8, 9]

e) [1, 3, 4, 2, 5, 7, 8, 9]

Um **algoritmo de ordenação** usando o **método da bolha** consiste em percorrer uma lista ou vetor diversas vezes de modo que o maior elemento vá para o **topo (fim da lista).** Assim, o método **da bolha verifica se cada elemento da posição i é maior que o próximo** (i+1). **Se for maior, trocam de lugar, se não, não trocam.**

Deste modo, a ordenação é feita parcialmente inserindo os maiores elementos à esquerda.

[**4**, 3, 1, 9, 8, 7, 2, 5] – Como 4 é maior que 3, troca de lugar → [3, 4, 1, 9, 8, 7, 2, 5]

[3, **4**, 1, 9, 8, 7, 2, 5] – Como 4 é maior que 1, troca de lugar → [3, 1, 4, 9, 8, 7, 2, 5]

[3, 1, **4**, 9, 8, 7, 2, 5] – Como 4 é não é maior do que 9, permanecerá como está → [3, 1, 4, 9, 8, 7, 2, 5]

[3, 1, 4, **9**, 8, 7, 2, 5] – Como 9 é maior do que 8, troca de lugar → [3, 1, 4, 8, 9, 7, 2, 5]

[3, 1, 4, 8, **9**, 7, 2, 5] – Como 9 é maior do que 7, troca de lugar → [3, 1, 4, 8, 7, 9, 2, 5]

[3, 1, 4, 8, 7, **9**, 2, 5] – Como 9 é maior do que 2, troca de lugar → [3, 1, 4, 8, 7, 2, 9, 5]

[3, 1, 4, 8, 7, 2, **9**, 5] – Como 9 é maior do que 5, troca de Lugar → [**3, 1, 4, 8, 7, 2, 5, 9**]

Assim, teremos na Linha 1: [3, 1, 4, 8, 7, 2, 5, 9]

A partir daí, seguiremos o mesmo raciocínio: basta ir comparando "i" com o próximo número e trocando de lugar se o número seguinte for menor. Assim, teremos:

Linha 2: [1, 3, 4, 7, 2, 5, 8, 9]

Linha 3: [1, 3, 4, 2, 5, 7, 8, 9]

Linha 4: [1, 3, 2, 4, 5, 7, 8, 9]

A quarta linha impressa garante que os valores 9, 8, 7,5 já estejam ordenados e terá o seguinte resultado: **[1, 3, 2, 4, 5, 7, 8, 9],** sendo D a alternativa correta.

Letra d.

030. (CETAP/AL-RR/ANALISTA DE SISTEMAS/2010) Sobre a linguagem de programação PYTHON, marque a alternativa INCORRETA.

a) Python suporta a maioria das técnicas da programação orientada a objetos.

b) Python suporta e faz uso constante de tratamento de exceções como uma forma de testar condições de erro e outros eventos inesperados no programa.

c) As funções são definidas em Phyton utilizando a palavra chave def.

d) A separação de blocos de código em Phyton é feita utilizando a endentação de código.

e) O operador lógico de conjunção (“e”, como em a e b) é &&.

Na linguagem Python, o operador lógico de conjunção (“e”) é “**and**” e não “&&” como afirma a alternativa E.

Veja trecho de código

EXEMPLO

```
1 idade = int(input('Digite sua idade: '))
2 if idade >= 10 and idade < 20:
3       print('Você é adolescente')
```

Letra e.

---

031. (FCC/TRT-MG/TÉCNICO JUDICIÁRIO – TECNOLOGIA DA INFORMAÇÃO/2015) Considere o código fonte Python abaixo.

Para que seja exibido [0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89] a lacuna “I” precisa ser preenchida corretamente com:

```
def calcular(n):
    resultado = []
    a, b = 0, 1
    while a < n:
            I
            ......
    return resultado

res = calcular(100)
print res
```

a) resultado.add(a)
   a, b = b, a+b

b) resultado.insert(a)
   a, b = b, a+b

c) resultado.append(a)
   a, b = b, a+b

d) resultado.add(a)
a, b = a, a+b
e) resultado.append(a)
a, b = a+b, b

Ambas as funções **append()** e **insert()** adicionam elementos em uma estrutura de dados do tipo **lista**, enquanto a função **add()** não é usada para este fim.
Na função **append() o elemento é adicionado no final da lista**, enquanto a função **insert()** é necessário passar como argumento a posição de inserção do elemento, o que não ocorre nas alternativas dadas.
O trecho de código associado a questão deve exibir (iniciando de a=0 e b=1) uma lista de valores que representam a soma dos dois elementos antecessores que sejam menor que 100.
Excluindo as alternativas que não possuem a função **append()**, restam duas alternativas (C e E). Como a função **append()** sempre recebe como argumento "a" (**append(a)**) e "a" sempre recebe a soma de "a+b" a alternativa correta é a letra C.
Letra c.

---

032. (FGV/TJ-BA/ANALISTA JUDICIÁRIO/2015) Analise o trecho de programa, escrito em Pyhton na versão 2.7, mostrado a seguir.

```
X=[]
for i in range(10,1,-1):
     X.append(i)
print X[3:]
```

O trecho exibe:
a) [7, 6, 5]
b) [7]
c) []
d) [7, 6, 5, 4, 3, 2]
e) [7, 6, 5, 4, 3, 2, 1]

A função **range** na linguagem Python pode ser entendida como: **range(início, fim, passo)**.
O início é baseado no índice zero (0) e o fim é o índice final, com **intervalo aberto**, ou seja, não está incluído. O passo é o salto ou diferença entre os índices.
Na questão, a função **range** formará os números de 10 até 2 (**o 1 será excluído**) com passo -1, ou seja, decrescente. **X = 10 9 8 7 6 5 4 3 2**.
O comando print vai exibir os itens da lista a partir da posição de índice 3 até o fim **[7, 6,5,4,3,2]**, sendo a alternativa D correta.
Letra d.

033. (IESES/IFC-SC/INFORMÁTICA/PROGRAMAÇÃO WEB E DISPOSITIVOS MÓVEIS/2015) Analise o seguinte trecho de código em Python 3.3.2 e assinale a alternativa correta:

```
class Cachorro:
        def tipo(self):
                return "10"

class Basset(Cachorro):
        def tipo(self):
                return "100"

cachorro1 = Cachorro()
cachorro2 = Basset()

print(cachorro1.tipo())
print(cachorro2.tipo())
```

a) Ocorrerá um erro de execução na linha "def tipo(self):" logo abaixo de "class Basset(Cachorro):".
b) A execução de "print(cachorro2.tipo())" mostrará o valor "10" na saída padrão.
c) A execução de "print(cachorro2.tipo())" mostrará o valor "100" na saída padrão.
d) Ocorrerá um erro de execução na linha "print(cachorro2.tipo())".

No trecho de código da questão, nota-se uma relação de herança entre classes. Ou seja, a classe Basset herda da classe Cachorro. É possível observar ainda o método tipo da classe Cachorro sendo redefinido na classe filha Basset. Deste modo, se um objeto for instanciado a partir da classe filha Basset ele irá se comportar conforme sua redefinição.
Nas últimas duas linhas do trecho de código da questão serão impressos, respectivamente, os valores "10" e "100". Deste modo, a alternativa correta é **letra C**. O objeto cachorro2 é instanciado a partir da classe Basset e, portanto, a saída é 100.
Letra c.

---

034. (PAQTCPB/UEPB/ANALISTA DE SISTEMAS/2012) No comando de atribuição em Python valor = raw_input("Digite um valor:"), qual o tipo da variável valor?
a) str
b) bool
c) int
d) float
e) file

Em versões anteriores do Python (2.0), para a entrada de dados do tipo String era necessário usar **raw_input**. Porém, em versões mais atuais da linguagem (3.0 e superior), o comando **input** captura os dados como string também.
Apesar da questão não ter citado a versão da linguagem Python, **raw_input** é específico para lidar com valores do tipo **String** e a alternativa correta é A (str).
Letra a.

035. (FCC/CNMP/DESENVOLVIMENTO DE SISTEMAS/2015) Considere os fragmentos de programas Phyton a seguir:

Fragmento 1:

```
for n in range(2, 10):
  for x in range(2, n):
      if n % x == 0:
        print n, '=', x, '*', n/x
          break
  else:
     print n, 'é um número primo'
```

Fragmento 2:

```
a = ['Casa', 'Mala', 'Prova']
for x in a:
      print x, len(x)
```

É correto afirmar que:

a) o Fragmento 1 está incorreto, pois laços não podem ter uma cláusula else.

b) no Fragmento 2, a instrução for está incorreta, pois ela não pode iterar sobre a.

c) o Fragmento 1 está incorreto, pois não é possível iterar sobre sequências numéricas utilizando a função range.

d) no Fragmento 1 é verificado se o quociente da divisão de n por x corresponde a 0.

e) os dois fragmentos de código estão corretos.

a) Errada. Em Python, **estruturas condicionais (if..else)** podem ser usadas em qualquer parte do código, bem como entre laços de repetição. Portanto, não há erros no fragmento 1.

b) Errada. No fragmento 2, "a" é um vetor ou lista de Strings e é possível "navegar" ou iterar usando uma instrução "for". Deste modo, não há erros no fragmento 2.

c) Errada. A função **range**() retorna uma série numérica no intervalo enviado como argumento, sendo possível iterar sobre ela. Assim, não há erros na utilização da função range().

d) Errada. O fragmento 1 está correto, a descrição na alternativa não condiz com a funcionalidade do código. O operador % retorna o resto da divisão e não o quociente.

e) Certa. Conforme visto nos comentários anteriores, os dois fragmentos estão corretos e a alternativa E está correta.

Letra e.

---

036. (FGV/TJ-BA/ANALISTA JUDICIÁRIO/2015) Analise o trecho de programa Python, na versão 2.7, apresentado a seguir.

```
L=[1,2,3,4,5,6,7,8]
print L[::-1]
```

Ao ser executado, o resultado exibido é:
a) [1, 2, 3, 4, 5, 6, 7, 8]
b) [8]
c) [ ]
d) [8, 7, 6, 5, 4, 3, 2, 1]
e) [1]

Em python, os elementos de uma **lista** são **definidos por colchetes e separados por vírgula**. Uma forma simplificada de imprimir os elementos é por meio do comando PRINT com a seguinte regra: **[ElementoInicial:ElementoFinal:Salto].**
Na questão, somente é especificado o salto que é -1. Deste modo, o Python entende que deve percorrer imprimindo os elementos de trás para frente (passo -1), sendo a alternativa correta a letra D.
Letra d.

---

037. (UERJ/UERJ/ANALISTA DE SISTEMAS/GRID/2015) Considere o trecho do programa Python abaixo:

```
def dobra (y):
x = y + y
return x
x = 5
dobra(x)
dobra(x)
print x
```

O valor impresso ao executarmos o programa é:
a) 5
b) 10
c) 15
d) 25

Nesta questão na **linguagem Python** há a definição de uma **função** chamada "dobra" que recebe um argumento de entrada e realiza uma operação soma (o dobro do valor), armazena o resultado em uma variável e retorna este valor (*retorn x*).
Nas quatro últimas linhas há um programa que chama a função "dobra". Primeiramente é atribuído à variável x o valor de 5. Em seguida, a função "dobra" é chamada duas vezes, consecutivamente, passando como argumento a variável x. Entretanto, o valor de x não é alterado, porque não há atribuição à variável x, permanecendo seu valor inalterada. Ao final, o valor exibido para x será o mesmo, ou seja, 5.
Letra a.

038. (CESPE/CORREIOS/ANALISTA DE CORREIOS/ANALISTA DE SISTEMAS – DESENVOLVIMENTO DE SISTEMAS/2011) Com relação aos sistemas de suporte a decisão e gestão de conteúdo, julgue os seguintes itens.
A linguagem Python e seu interpretador estão disponíveis para as mais diversas plataformas. Para que seja usado em determinado sistema operacional não suportado, é possível gerar o Python a partir do programa fonte utilizando um compilador C. Nesse caso, o código fonte é traduzido para o formato bytecode, que é multiplataforma e pode ser distribuído de forma independente.

A linguagem e seu interpretador estão disponíveis para as mais diversas plataformas, desde Unix (Linux, FreeBSD, Solaris, MacOS X, etc.), Windows,.NET, versões antigas de MacOS até consoles de jogos eletrônicos ou mesmo alguns celulares e palmtops (WIKIPEDIA, 2020).
Para algum sistema operacional não suportado, basta que exista um compilador C disponível e gerar o Python a partir do seu código fonte. O código fonte é traduzido pelo interpretador para o formato **bytecode**, que é multiplataforma e pode ser executado e distribuído sem fonte original (WIKIPEDIA, 2020).
Certo.

---

039. (FGV/TÉCNICO DA PROCURADORIA (PGE RO)/TECNOLOGIA DA INFORMAÇÃO/2015) Na linguagem **Python 2.7**, os comandos
L=range(0,12)
print L
produzem:
a) [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
b) [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]
c) [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]
d) [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 12]
e) uma mensagem de erro

Vamos ao detalhamento da função **range**:

class range(start, stop[, step])	**start:** número de início. Nesse caso, conta **de start (incluso) até stop (não-incluso)**. Por padrão, o *step* tem valor igual a **1. Se *step* for fornecido, faz a iteração de *step* em *step***.
class range(stop)	Se o argumento *start* for omitido, o valor padrão é 0.

Em **range(0, 12)**, a função **range** cria **uma** <u>lista</u> **de números que se seguem um após o outro** (os números foram dados como parâmetros). Mas, note que o segundo desses dois números não está incluído na lista que o Python mostrará. Assim, em range(0, 12), conta

de 0 a 11, e o 12 não é incluído, porque o intervalo é semiaberto, o que significa que ele inclui o primeiro valor, mas não o último. Como resposta, teremos o intervalo **[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]**.

Letra b.

---

040. (CESPE/SERPRO/ANALISTA – SUPORTE TÉCNICO/2010) Acerca das linguagens de programação, julgue os itens subsequentes.

As linguagens Perl e Python são dinamicamente tipadas, ou seja, o tipo da variável é definido em tempo de execução. As linguagens C e Java são estaticamente tipadas, ou seja, o tipo de variável é definido em tempo de compilação.

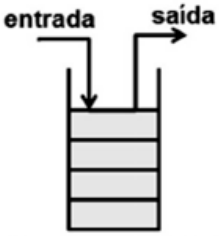
Em Python, as variáveis não precisam ser previamente declaradas e também não precisam ser de um tipo particular, isto é, não precisam ser definidas para representar números, caracteres, datas, etc. Dizemos que Python é uma linguagem dinamicamente tipada, pois o tipo da variável é redefinido sempre que ocorrer uma nova atribuição de valor.

Certo.

LÓGICA DE PROGRAMAÇÃO

041. (FUNCAB/PRODAM – PROCESSAMENTO DE DADOS AMAZONAS S.A /PROGRAMADOR-DESENVOLVEDOR/2014) Observe as figuras I e II, que representam duas estruturas de dados.

I - FIFO – "First In First Out"



II - LIFO – "Last In First Out"

Essas estruturas de dados são denominadas respectivamente:

a) fila e pilha.

b) vetor e lista.

c) fila e vetor.

d) vetor e pilha.

e) fila e lista.

Questão aborda a definição de duas estruturas de dados bem conhecidas. A primeira é a **Fila**, **que se baseia no princípio FIFO (*First in, first out* – Primeiro a entrar, primeiro a sair)**. Na fila são realizadas duas operações básicas: adicionar um elemento ao final da fila (ENQUEUE), e remover o elemento no início da fila (DEQUEUE).
Já a **pilha** **baseia-se no princípio LIFO (*LAST in, FIRST out* – Último a entrar, primeiro a sair).** Na pilha existem duas funções básicas: PUSH(Empilha), que insere um dado no topo da pilha, e POP(Desempilha), que remove o item no topo da pilha.
Outras estruturas de dados relevantes são:
Vetores – compostos por um número fixo (finito) de elementos de um determinado tipo de dados;
Lista – compostas por nós que apontam para o próximo, podendo ser encadeadas de formas diferentes; e
Árvores – cada elemento tem um ou mais elementos associados.
Letra a.

---

042. (IBFC/PC-RJ/PERITO CRIMINAL/ENGENHARIA DA COMPUTAÇÃO/2013) Marque a opção que determina uma lista de procedimentos bem definida, que pega algum valor, ou conjunto de valores como entrada, e produz algum valor ou conjunto de valores como saída.
a) Algoritmo.
b) Código.
c) Chave
d) Inserção.
e) Loop.

Dá-se o nome de **algoritmo** **a uma sequência lógica finita de instruções cujo objetivo é a execução de uma determinada tarefa**. Trata-se de uma **lista** de procedimentos bem definida, que pega algum valor, ou conjunto de valores como entrada, e produz algum valor ou conjunto de valores como saída.
Letra a.

---

043. (VUNESP/CTA/TÉCNICO EM INFORMÁTICA/2013) Em linguagem de programação, uma variável é:
a) o resultado de uma expressão aritmética.
b) o nome dado às informações salvas no disco.
c) um número, uma letra ou um ponto-flutuante.
d) uma posição de memória identificada.
e) uma palavra especial utilizada pela linguagem para identificar suas instruções de controle.

Em linguagem de programação, **variáveis** são **recipientes** (endereços de memória) que **armazenam informações** de um **determinado tipo** para que seja possível a manipulação delas pelos programas.

Letra d.

---

044. (FCC/TRT – 9ª REGIÃO (PR)/ANALISTA JUDICIÁRIO – TECNOLOGIA DA INFORMAÇÃO/2013) Considere o algoritmo em pseudo linguagem:

```
início
caracter: nome, sexo;
    imprima ("Qual é o seu nome? ");
    leia(nome);
    imprima ("Qual é o seu sexo? (f/m) ");
    leia (sexo);

    se (sexo = 'f' E sexo = 'F')
    então imprima ("Você é do sexo feminino. ");
    senão
        se (sexo = 'm' E sexo = 'M')
            então imprima ("Você é do sexo masculino. ");
        senão
            imprima ("Você digitou um valor de sexo invalido ");
        fim se;
    fim se;
fim.
```

Sobre o algoritmo acima é correto afirmar que

a) a lógica do algoritmo está comprometida pela falta de um comando de repetição.

b) em vez de utilizar comandos de decisão **se** aninhados, deveria ter sido usado um único comando de seleção múltipla, por isso a lógica ficou comprometida.

c) se for digitada uma letra maiúscula 'F' ou minúscula 'f' será impresso **Você é do sexo feminino.**

d) somente se a letra digitada para o sexo for diferente de 'F', 'f', 'M', 'm' é que a frase **Você digitou um valor de sexo inválido** será impressa.

e) não importa o valor digitado no sexo, pois sempre será impresso **Você digitou um valor de sexo inválido**.

O algoritmo está quase correto. No entanto, mesmo que o usuário informe 'F' ou 'M', ele não identificaria corretamente o seu sexo. Isso ocorre porque foi utilizado o **conector lógico E nas condições**. Então, nunca um caracter poderá ser igual a 'F' e 'f' ao mesmo tempo. O mesmo ocorre também com 'M' e 'm'. Sendo assim, as condições retornariam falsas e a execução sempre cairia no último senão.

Letra e.

REFERÊNCIAS

ARAÚJO, R. **Conversões e Casting de Tipos de Dados em Python.** 2022. Disponível em: <https://blog.grancursosonline.com.br/conversoes-e-casting-de-tipos-de-dados-em-python/>. Acesso em: 23 jan. 2023.

DevMedia. **Application Programming Interface: Desenvolvendo APIs de Software.** 2018. Disponível em: <https://www.devmedia.com.br/application-programming-interface-desenvolvendo-apis-de-software/30548>. Acesso em: 06 ago. 2020.

_____. **Python.** Disponível em: <https://www.devmedia.com.br/python-tutorial/33274>. Acesso em: 06 ago. 2020.

*JDOODLE. **Online Compiler And Editor***. https://www.jdoodle.com/. Acesso em: 05 ago. 2020.

Python.org. **Python Software Foundation (PSF)**. Disponível em: <https://www.python.org/>. Acesso em: 05 ago. 2020.

_____. Página da comunidade no Brasil. Disponível em: <https://python.org.br/>. Acesso em: 05 ago. 2020.

SEBESTA, R. W. **Conceitos de Linguagens de Programação** – 11.ed. Porto Alegre: Bookman, 2018.

TREINAWEB. **Guia da linguagem Python.** 2022. Disponível em: <https://www.treinaweb.com.br/>. Acesso em 20 jan. 2023.

Universidade da Tecnologia. Python: Características, Noções e Guia de Estudo. Disponível em: <https://universidadedatecnologia.com.br/estudo-linguagem-python-2018/>. Acesso em: 05 ago. 2020.

_____. Ranking de Linguagens UTec 2018. Disponível em: <https://universidadedatecnologia.com.br/ranking-de-linguagens-utec-2018/>. Acesso em: 06 ago. 2020.

_____. Linguagem de Programação: Classificações. Disponível em: <https://universidadedatecnologia.com.br/linguagem-de-programacao-classificacoes/>. Acesso em: 05 ago. 2020.

Wikipedia. **Python.** Disponível em: <https://pt.wikipedia.org/wiki/Python>. Acesso em: 08 ago. 2020.

Abra

caminhos

crie

futuros

gran.com.br