Secretaria de Educação e Esportes

GOVERNO DE PERNAMBUCO
ESTADO DE MUDANÇA

Programação com Python

Orientações para Novas Oportunidades de Aprendizagem

Elaboração

Gabriel Pimenta Carneiro Campelo

Equipe de coordenação

Janine Furtunato Queiroga Maciel
Gerente de Políticas Educacionais do Ensino Médio (GGPEM/SEMP)

Rômulo Guedes e Silva
Gestor de Formação e Currículo (GGPEM/SEMP)

Andreza Shirlene Figueiredo de Souza
Chefe da Unidade de Formação e Currículo do Ensino Médio (GGPEM/SEMP)

Revisão

Andreza Shirlene Figueiredo de Souza
Ana Caroline Borba Filgueira Pacheco

Sumário

Este material foi produzido a partir do Material de Apoio a ação Docente, disponível em: Programação-com-Python
Autor: Gabriel Pimenta.

Introdução

Olá **estudante**.

Este caderno foi escrito especialmente para você, estudante do Ensino Médio. Aqui você encontrará uma abordagem sobre a unidade curricular ***Programação com Python***, com atividades e formas de discussão das temáticas de maneira mais próxima, mediada por este caderno. Dúvidas podem ser tiradas com seus professores na escola.

A Unidade Curricular ***Programação com Python*** - presente na *Trilha* **Tecnologias Digitais**, no Novo Ensino Médio da Rede Pública Estadual de Pernambuco - tem o objetivo de aprofundar conhecimentos que você já estudou na Formação Geral Básica (FGB), do nosso currículo.

Dessa forma, uma parte das atividades deste material são desplugadas, ou seja, não exige internet nem computador. No entanto, sugerimos que havendo a possibilidade de realizar os exemplos usando um computador, a aprendizagem tende a ser mais significativa. Há também a possibilidade de usar aplicativo gratuito para *smarthphones/tablet* com sistemas operacionais Android©.

Vamos iniciar nossos estudos para trilhar os caminhos do conhecimento, valorizando as questões referentes à esta Unidade Curricular.

Tecendo Conhecimento 1

O que é um Algoritmo?

Inúmeras atividades do nosso cotidiano são feitas de forma automática. Ao decidir andar de um lugar para o outro, não nos preocupamos em indicar ao nosso cérebro quais músculos devem ser usados, simplesmente damos a instrução e a partir daí, a ação acontece. Numa analogia simples, podemos dizer que seus neurônios executam uma sequência de instruções organizadas e precisas que resultam no ato de caminhar.

Sempre que um processo puder ser descrito por instruções claras e sequenciadas chamaremos de **Algoritmo**. Pense numa receita de bolo. Cada etapa é descrita separadamente e na ordem fixa. Se por acaso mudarmos o ordenamento das instruções, corremos o risco de não acertar a receita. Então, ao conceber um algoritmo pense na sequência de eventos e qual a sucessão entre eles.

Uma forma de representar Algoritmos é através de Descrições Narrativas. Escrevemos cada comando com frases curtas ocupando linhas distintas, uma depois da outra. Sempre começando de cima para baixo. Veja os exemplos:

TOMAR BANHO

1 Despir-se;
2 Ligar o chuveiro;
3 Molhar-se;
4 Desligar o chuveiro;
5 Ensaboar-se;
6 Ligar o chuveiro;
7 Enxaguar-se;
8 Fechar o chuveiro;
9 Enxugar-se;
10 Vestir-se.

FAZER UMA MÉDIA (duas notas)

1 Receber as duas notas;
2 Somar as duas notas;
3 Dividir o resultado da soma por 2;
4 Se a média for maior que 7;
5 Escreva “APROVADO/A”
6 Se não;
7 Escreva “RECUPERAÇÃO”;

Fig.1: Exemplos de Algoritmos em Descrição Narrativa
Fonte elaborada pelo autor.

O que é Linguagem de Programação?

Por incrível que pareça, um computador é burro. A verdade dos fatos é que toda a informação que está contida num computador pode ser representada por sequências que só comportam dois valores: Zero e Um. Então, tudo que digitamos, lemos, imprimimos e demais tarefas são executadas com conjuntos de bilhões de zeros e uns, escritos na ordem correta. Esta forma de expressão é denominada código binário.

Para que possamos descrever a um computador aquilo que gostaríamos que ele fizesse, teríamos que ter a capacidade de organizar bilhões de zeros e uns em sequência e sem errar a posição de nenhum deles. Isso é humanamente impossível! Para conseguir programar, utilizamos uma **Linguagem de Programação** que vai aceitar que escrevemos em palavras corriqueiras (linguagem de alto nível) e vai converter para o código binário (linguagem de baixo nível). Essa transcrição é feita usando a lógica de um Algoritmo que neste caso passa a ser chamado de código ou programa.

Fig.2: Imagem gerada pela inteligência artificial DALL-E 3.

Roteiro de Atividades 1

1. De acordo com o conceito de Algoritmo discutido, ordene as seguintes diretivas na forma mais adequada para obter o resultado esperado:

Item 1.

	Instruções	Ordem Correta
1.	Acrescente sal;	
2.	Sirva num prato limpo;	
3.	Quebre a casca do ovo;	
4.	Derreta a manteiga no fundo da frigideira;	
5.	Espere até que fique firme;	
6.	Despeje o ovo na frigideira;	
7.	Coloque a frigideira na chama do fogão;	

Item 2.

	Instruções	Ordem Correta
1.	Coloque o macaco;	
2.	Erga o carro;	
3.	Retire as ferramentas, step e triângulo da mala do carro;	
4.	Coloque as porcas;	

5.	Faça o macaco descer o carro;	
6.	Arme o triângulo atrás do carro;	
7.	Desarme o triângulo;	
8.	Afrouxe as porcas;	
9.	Aperte as porcas em, no mínimo, 30º;	
10.	Coloque o step;	
11.	Guarde o pneu, as ferramentas e o triângulo na mala do carro;	
12.	Retire o pneu furado;	
13.	Retire as porcas;	

1.1. Analise os algoritmos abaixo e diga o que será impresso na tela ao serem executados:

(A)

A ← 10
B ← 20
Escreve B
B ← 5
Escreve A, B

(B)

A ← 30
B ← 20
C ← A + B
Escreve C
B ← 10
Escreve B, C
C ← A + B
Escreve A, B, C

(C)

A ← 10
B ← 20
C ← A
B ← C
A ← B
Escreve A, B, C

(D)

A ← 10
B ← A + 1
A ← B + 1
B ← A + 1
Escreve A
A ← B + 1
Escreve A, B

Use o texto para responder as três próximas questões:

"Robô Rob é um robô que adora explorar. Hoje, ele vai explorar uma floresta misteriosa. O Robô Rob encontra três obstáculos diferentes na floresta. Para cada obstáculo, explique como Robô Rob usa sua lógica de programação para superar o desafio."

1.2. Robô Rob encontra um rio que precisa atravessar. Ele decide usar um "loop" para construir uma ponte. O que Robô Rob deve fazer?

a) Usar uma "condicional" para verificar a profundidade do rio.
b) Colocar tábuas até que a ponte esteja completa.
c) Criar uma "função" para medir a largura do rio.
d) Usar variáveis para armazenar a quantidade de tábuas.

1.3. Ao encontrar um tronco caído, Robô Rob decide usar uma "condicional" para decidir o que fazer para continuar no trajeto. O que ele deve verificar?

a) Se o tronco está oco ou não.
b) Se há pássaros cantando no tronco.
c) Se ele pode passar por cima ou deve desviar.
d) Voltar para o ponto de origem.

Tecendo Conhecimento 2

AMBIENTE DE PROGRAMAÇÃO (IDLE PYTHON)
O que é Ambiente de Programação?

Se lhe for pedido para enviar uma mensagem de texto para o celular de um amigo, quantas opções você terá para fazer essa tarefa? De certo, mais de quatro. Independente da sua escolha, será feito um texto com as informações necessárias que serão enviadas de um aparelho até o outro. Pois bem, imagine que o texto é um "código" que você desenvolveu e o aplicativo usado para enviar é um **ambiente de programação**. Um programa de computador que nos permite criar, editar e executar nossos próprios programas de computador.

Assim como a mensagem de texto, temos vários ambientes de programação para a linguagem de programação Python. Alguns são gratuitos e outros pagos. A referência para este material será a IDLE Python que tem distribuição sem custos através do site oficial (https://www.python.org/) .

Primeiros passos

Após baixar e instalar a IDLE Python no computador, inicie um novo ambiente de desenvolvimento. É preciso abrir o IDLE e em seguida clicar em **File > New File** ou aperte **Ctrl+N**. Nesta nova página que se abriu, temos um ambiente que comporta o código em desenvolvimento que deve ser escrito, e só ao final será executado por um comando dado pelo/a usuário/a. Para visualizar o resultado dos comandos é preciso clicar em **Run > Run Module** ou apertar **F5**. Com isso, vai ser solicitado que o programa seja salvo antes de ser executado.

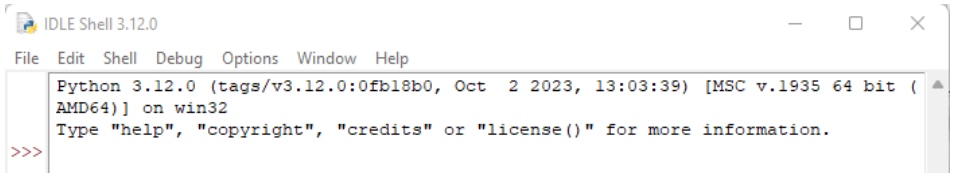

Fig.2: Tela de desenvolvimento da IDLE Python.
Fonte adaptada pelo autor.

Pydroid

Uma alternativa ao uso de um computador é a substituição por um *smartphone* ou *tablet* com sistema operacional Android©. Basta instalar o *app* Pydroid na loja de aplicativos. A versão gratuita permite editar códigos e executá-los.

Fig.3: Tela de desenvolvimento do Pydroid.
Fonte: adaptada pelo autor.

"Alô Mundo" no Python

Seguindo as instruções do capítulo anterior, faça seu primeiro código usando a IDLE Python. Utilize o modelo abaixo:

Faça um código que exibe a frase "Alô, mundo!" na tela do IDLE	Saída na IDLE do *Python*:
`print ("Alô, mundo!")`	`'Alô, mundo!'`

Sintaxe

Para desenvolver uma Linguagem de Programação é preciso seguir regras gramaticais de escrita. Alguns desses comandos têm significados próprios e acumulam ações. Vamos nos dedicar a entender como usar as palavras certas, nos lugares certos.

A linguagem de programação Python utiliza Objeto, Método e Módulo para estruturar sua sintaxe e ordenar as instruções.

- **Objeto:** alocação de memória que armazena letras, palavras, frases, listas, valores e tudo aquilo que se deseja referenciar no código desenvolvido.
- **Método:** comando que acumula um conjunto de atividades pré-programadas que utiliza o objeto como parâmetro/atributo.
- **Módulo:** são algumas rotinas mais comuns que já estão organizadas na linguagem Python. Um bom exemplo é o módulo "math" que implicitamente tem operações mais complexas como raiz quadrada, fatorial dentre outras. Dentro dos módulos, podemos ter **funções** (conjuntos de códigos que têm um nome: média, soma, por exemplo).

Exemplos:

Instrução	Sintaxe
Atribuir a palavra "Einstein" ao objeto "nome".	`nome = "Einstein"` Objeto Atributo
Incluir ao final do objeto "lista" o valor 25. Obs.: dentro dos parênteses pode ser um objeto também.	`lista.append(25)` Objeto Método
Utilizar o módulo math para extrair a raiz quadrada do valor 16. Obs.: dentro dos parênteses pode ser um objeto também.	`math.sqrt(16)` Módulo Função

Tab. 1: Aplicações de Objeto, Método e Módulo. Fonte: LabTIME (2017, p.09).

Outra regra muito importante da diagramação de programas em Python é o uso obrigatório de "indentação". Se houver a necessidade de utilizar um bloco dentro do seu programa principal, é preciso avançar a tabulação desde sub código. Observando a figura 1 da página 2 vemos que o algoritmo que usamos para adquirir a média de duas notas tinha duas estruturas de decisão. Uma para valores maiores que 7 outra para menores. Então, cada decisão ocupa uma indentação diferente do código principal.

Na figura 3 temos um diagrama que ilustra como aplicar a indentação de forma coerente num ambiente de desenvolvimento Python. Ao indicar o início de um bloco à parte, sinalizamos com o uso de ":" e o final dar-se pelo retorno do parágrafo que avançamos.

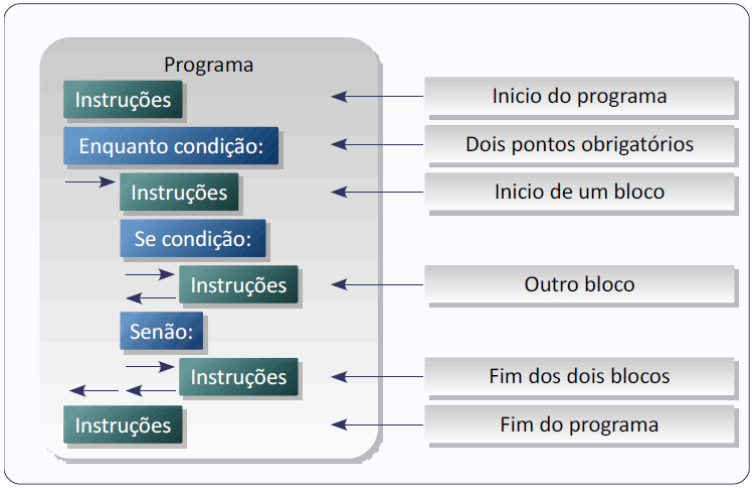

Fig.3:Estrutura de indentação. Fonte: FRANCO, João (2010, p.06)

Declaração de variáveis

Variável é um termo que adotamos no código com objetivo de alocar uma certa quantidade de memória do computador para armazenar a informação. Na linguagem Python temos 4 tipos de dados:

- **Interge:** expressa quantidades numéricas exatas. Não aceitando números decimais.
- **Float:** armazena valores decimais.
- **String:** usado para agrupamentos de letras.
- **Booleanos:** armazena estado como falso ou verdadeiro.

OBS.: Existem outros formatos para declaração de variáveis. Para os próximos passos vamos nos concentrar nesses quatro.

A melhor estratégia é ter consciência de qual conteúdo da informação que se deseja armazenar pois, quanto menos memória usar, mais fluido será seu programa e, por tanto, melhor.

Por exemplo, se em algum momento for solicitada a idade de um usuário não é necessário armazenar como um *float* pois vai ocupar um tamanho de memória muito maior que o necessário. Idade são sempre números inteiros. A melhor escolha é usar a declaração de números inteiros (*interge*). Da mesma forma, ao se questionar o peso de um usuário, provavelmente o resultado será valores decimais e quanto mais exato, melhor. Aí sim, cabe o uso de variáveis do tipo *float*.

Para que uma variável receba do usuário uma informação, usamos a sintaxe *input( )* atribuindo a variável, usando o sinal gráfico de igualdade (“=”). O compilador IDLE vai aguardar até que a informação seja inserida no console e só então, dará continuidade ao programa.

Sempre que for necessário exibir na tela a informação armazenada numa variável, basta escrever *print* (NOME_DA_VARIÁVEL) numa das linhas do código.

Exemplos:

Código para solicitar e exibir o nome de uma pessoa:	Saída na IDLE do *Python*:
`nome = input ("Digite o seu nome: ")` `print (nome)`	`Digite seu nome: Albert Einstein` `Albert Einstein`

Fonte elaborada pelo autor.

É possível direcionar o tipo de dado que o usuário pode inserir no código. Para isso, antes da declaração *input( )*, devemos direcionar o tipo aguardado usando *int( )*, para inteiros, *float( )* para decimais. Por padrão, *input( )* sempre devolve *string*, se não for direcionado.

Código para solicitar a altura de uma pessoa:	Saída na IDLE do *Python*:
`altura = float (input ("Digite a sua altura: "))` `print (altura)`	`Digite sua altura: 1.71` `1.71`
Código para solicitar a idade de uma pessoa:	**Saída na IDLE do *Python*:**
`idade = int (input ("Digite a sua idade: "))` `print (idade)`	`Digite sua idade: 78` `78`

Fonte elaborada pelo autor.

Roteiro de Atividades 2

2. De acordo com o que foi discutido, junte os conceitos da coluna I com o texto da coluna II ligando uma bola a um quadrado.

Ambiente de Programação ○ □ Alocação de memória que armazena letras, palavras, frases, listas, valores e tudo aquilo que se deseja

Objeto	○	□	referenciar no código desenvolvido.
Método	○	□	Expressa quantidades numéricas exatas. Não aceitando números decimais.
Interge	○	□	Um programa de computador que nos permite criar, editar e executar nossos próprios programas de computador.
			Comando que acumula um conjunto de atividades pré-programadas que utiliza o objeto como parâmetro/atributo.

2.1. Para armazenar o ano de nascimento de um usuário qual o melhor tipo de variável:
a) interge.
b) float.
c) booleano.
d) string.

2.2. Para armazenar o peso de um de um usuário qual o melhor tipo de variável:
a) interge.
b) float.
c) booleano.
d) string.

2.3. Para armazenar cidade natal de um usuário qual o melhor tipo de variável:
a) interge.
b) float.
c) booleano.
d) string.

Tecendo Conhecimento 3

OPERADORES ARITMÉTICOS E LÓGICOS

A linguagem de programação Python já possui uma série de funções designadas para realizar operações lógicas matemáticas básicas. Porém é preciso declarar de forma correta sua sintaxe para obter o resultado desejado.

Operação	Símbolo
Soma	+
Subtração	-
Multiplicação	*
Divisão	/

Operação	Símbolo
Potenciação/ Radiciação	**
Resto da divisão	%

Tab.2: Operadores aritméticos. Fonte elaborada pelo autor.

Para fazer comparações entre objetos, utilizamos operadores matemáticos:

Descrição	Símbolo
Menor que	<
Maior que	>
Menor ou igual	<=

Descrição	Símbolo
Maior ou igual	>=
Igual (comparação)	==
Diferente	<>

Tab.3:Operadores matemáticos. Fonte elaborada pelo autor.

Os operadores lógicos são utilizados para combinar expressões e realizar operações lógicas. Eles nos permitem avaliar múltiplas condições ao mesmo tempo.

Descrição	Sintaxe
Negação	not
Aditivo	and
Comutativo	or

Tab.4:Operadores Lógicos. Fonte elaborada pelo autor.

Segundo o que as diretivas preconizam, seguem os resultados das comparações usando operadores lógicos.

Diretiva	Análise
VERDADEIRO and VERDADEIRO	Verdadeiro
VERDADEIRO and FALSO	Falso
FALSO and VERDADEIRO	Falso
FALSO and FALSO	Falso
VERDADEIRO or VERDADEIRO	Verdadeiro
VERDADEIRO or FALSO	Verdadeiro
FALSO or VERDADEIRO	Verdadeiro
FALSO or FALSO	Falso
not (2 == 1)	Verdadeiro
not (2 == 2)	Falso

Tab.5: Análise de comparadores lógicos e aritméticos. Fonte elaborada pelo autor.

CONCATENAÇÃO E INTERPOLAÇÃO DE STRINGS

Uma técnica de apresentação de resultados e relatórios na tela da IDLE Python e através do uso de Concatenação e Interpolação de string.

- Concatenação: consiste na alternância entre texto e variáveis na tela de exibição do console da IDLE.

Um código que solicita e exibe o nome e a data de nascimento daquela pessoa:	Saída na IDLE do *Python*:

`nome = input ("Digite o seu nome: ")`
`nascimento = input ("Digite sua data de nascimento: ")`
`print ("O sr. ou sra. "+ nome + " nasceu no dia: "+ nascimento)`
	`Digite seu nome: Albert Einstein`
	`Digite a data de nascimento: 14/03/1879`
	`O sr. ou sra. Albert Einstein nasceu no dia 14/03/1879`

Fonte elaborada pelo autor.

- Interpolação: usa o operador "%" para atribuir a exibição da informação contida numa variável que deve ser corretamente declarada de acordo com o seu tipo após a solicitação.

Um código que solicita e exibe o nome e a altura de uma pessoa:	Saída na IDLE do *Python*:

`nome = input ("Digite o seu nome: ")`
`altura = float (input ("Digite a sua altura: "))`
`print ("O/A sr./sra. %s tem a altura de %.2f metros." %(nome, altura))`
	`Digite seu nome: Albert Einstein`
	`Digite sua altura: 1.7`
	`O sr. ou sra. Albert Einstein tem a altura de 1.70 metros.`

Fonte elaborada pelo autor.

O uso de interpolação é mais indicado para a economia da alocação de memória pelo programa em *Python*.

Símbolos usados na interpolação:

- %s: string.
- %d: inteiro.
- %o: octal.
- %f: real.

OBS. No código usado para fazer a interpolação de dados com escrita foi acrescentado ao operador "%f" o valor ".2" ficando "%.2f". Com isso limitamos a exibição para duas casas decimais após à vírgula.

Roteiro de Atividades 3

3. Observe o seguinte código e informe o que se pede:

```
A = 10
B = 30
C = (A**2) + B
D = B/A
print ("Valor A é %d, de B é %d e de C é %d" %(A,B,C))
```

Fonte elaborada pelo autor.

(A) O valor exibido para a variável A é : _____
(B) O valor exibido para a variável B é : _____
(C) O valor exibido para a variável C é : _____
(D) O valor exibido para a variável D é : _____

3.1 Observe o seguinte código e informe na tabela se a diretiva resulta num estado falso ou verdadeiro:

```
A = 10
B = 30
C = (A**2) + B
D = B/A
print ("Valor A é %d, de B é %d e de C é %d" %(A,B,C))
```

Fonte elaborada pelo autor.

	Diretiva	Análise (Verdadeiro ou falso)
a)	`B >= A and D <= C`	
b)	`B == A or D <= C`	
c)	`not B == C`	
d)	`not C == 130`	

Fonte elaborada pelo autor.

Atividade prática

3.2 Faça um Programa que peça o raio de um círculo, calcule e mostre a sua área e o comprimento.

3.3 Escreva um programa que recebe dois valores do tipo inteiro (x e y), e calcule o valor de z com três casas decimais de precisão: (Obs.: x ≠ y)

$$z = \frac{(x^2+y^2)}{(x-y)^2}$$

Tecendo Conhecimento 4

COLEÇÃO DE VARIÁVEIS E OBJETOS

Uma das formas de melhorar a automação de processos é a utilização de conjuntos de variáveis e objetos que facilitem a organização das informações. Em Python existem dois jeitos de fazer isso. Usando Tuplas ou Listas.

Tuplas

Uma vez que uma Tupla é feita, não pode ser alterada. É um conjunto de informações imutáveis, porém, acessíveis. Ainda é possível excluir itens de uma Tupla, mas não acrescentar. Só consultar.

Sintaxe:

```
nome_da_tupla = ('valor1', 'valor2', ...,
                 'valorN')
```

Uma forma muito comum de usar Tuplas é através da construção de **Dicionários**. Neste formato a Tupla terá uma chave associada a um objeto/valor. Pense numa lista telefônica de um celular. Nela conta o nome (chave) e o telefone (objeto).

Sintaxe:

```
nome_do_dicionário = {chave1: valor1,
                      chave2: valor2,
                           ...
                      chaveN: valorN}
```

Exemplo:

Um dicionário de produtos e respectivos preços que exibe o preço da farinha:	Saída na IDLE do *Python*:

`preco = {"arroz": 6.90, "feijão": 12.39, "farinha": 7.95, "macarrão": 5.86, "biscoito": 2.30}`
`print (preco)`
`print (preco["farinha"])`
	`{'arroz': 6.9, 'feijão': 12.39, 'farinha': 7.95, 'biscoito': 2.3}`
	`7.95`

Fonte elaborada pelo autor.

Outros métodos podem ser executados com um objeto no formato de Tuplas e Dicionários. São elas:

Método	Descrição	Exemplo	
del	Exclui item	Del.preco{"biscoito"}	
in	Verifica a existência da chave	"batata" in preco False	"feijão" in preco True
keys()	Exibe as chaves	preco.keys() dict_keys(['arroz', 'feijão', 'farinha', 'macarrão', 'biscoito'])	
values()	Exibe os valores	preco.values() dict_values([6.9, 12.39, 7.95, 5.86, 2.3])	
items()	Retornar uma lista contendo pares de tuplas (o primeiro elemento será a chave do e o segundo elemento o valor).	preco.items() dict_items([('arroz', 6.9), ('feijão', 12.39), ('farinha', 7.95), ('macarrão', 5.86), ('biscoito', 2.3)])	

Tab. 6: Operadores para manipular dicionários. Fonte elaborada pelo autor.

Lista

A principal diferença entre uma Tupla e uma Lista é que a segunda pode ser manipulada a qualquer momento.

Sintaxe:

```
nome_da_lista = ['valor1', 'valor2', 'valor3',..., 'valorN']
```

Exemplo:

Uma lista das capitais do Nordeste do Brasil:	Saída na IDLE do *Python*:
`capitais = ['Recife', 'João Pessoa', 'Maceió', 'Salvador', 'Natal', 'Teresina', 'São Luís', 'Fortaleza', 'Aracajú']` `print (capitais)`	`['Recife', 'João Pessoa', 'Maceió', 'Salvador', 'Natal', 'Teresina', 'São Luís', 'Fortaleza', 'Aracaju']`

Fonte elaborada pelo autor.

Diretivas que instruem Listas:

Função	Descrição	Exemplo
len	retorna o tamanho da lista.	L = [1, 2, 3, 4] len(L) → 4
min	retorna o menor valor da lista.	L = [10, 40, 30, 20] min(L) → 10
max	retorna o maior valor da lista.	L = [10, 40, 30, 20] max(L) → 40
sum	retorna a soma dos elementos da lista.	L = [10, 20, 30] sum(L) → 60
append	adiciona um novo valor no final da lista.	L = [1, 2, 3] L.append(100) L → [1, 2, 3, 100]
insert	insere um novo valor numa posição específica da lista	L = [0, 1, 2] L.insert(1,A) L → [0, A, 1, 2]
del	remove um elemento da lista, dado seu índice.	L = [1,2,3,4] del L[1]

		L → [1, 3, 4]
in	verifica se um valor pertence à lista.	L = [1, 2 , 3, 4] 3 in L → True
sort()	ordena em ordem crescente de valor.	L = [3, 5, 2, 4, 1, 0] L.sort() L → [0, 1, 2, 3, 4, 5]
reverse( )	inverte os elementos de uma lista.	L = [0, 1, 2, 3, 4, 5] L.reverse() L → [5, 4, 3, 2, 1, 0]
list(i-1)	acessar o enésimo termo de uma lista. (O primeiro termo de uma lista ocupa a posição zero.)	L = [0, 1, 2, 3, 4, 5] L.list (3) 3

Tab. 7: Operadores para manipular Listas. Fonte elaborada pelo autor.

Roteiro de Atividades 4

4. Marque a alternativa que reflete corretamente um Dicionário da tabela de preços de uma lanchonete

Lanchonete	
Produtos	**Preços R$**
Salgado	R$ 4.50
Lanche	R$ 6.50
Suco	R$ 3.00
Refrigerante	R$ 3.50
Doce	R$ 1.00

(A)
```
lanchonete = {4.50, 6.50, 3.00, 3.50, 1.00}
```
(B)
```
lanchonete = [“Salgado”: 4.50, “Lanche”: 6.50, “Suco”: 3.00, “Refrigerante”: 3.50, “Doce”: 1.00]
```
(C)
```
lanchonete = {“Salgado”: 4.50, “Lanche”: 6.50, “Suco”: 3.00, “Refrigerante”: 3.50, “Doce”: 1.00}
```
(D)
```
lanchonete = {“Salgado”, “Lanche”, “Suco”, “Refrigerante”, “Doce”}
```

4.1 Dada a lista:
```
L = [7, 6, 8, 9, 3, 5]
```
Se executar a diretiva:
```
L.sort()
```
Como fica a nova lista?
(A) `L = [7, 6, 8, 9, 3, 5]` (B) `L = [5, 3, 9, 8, 6, 7]`
(C) `L = [3, 5, 6, 7, 8, 9]` (D) `L = [9, 8, 7, 6, 5, 3]`

4.2 O que acontece quando tentamos modificar o terceiro elemento da tupla frutas para "uva"?
```
frutas = ("maçã", "banana", "laranja")
```
a) O elemento é modificado com sucesso.
b) Nada acontece.
c) Ocorre um erro TypeError.
d) O elemento é removido da tupla.

4.3 Como adicionamos "lápis" com a quantidade 15 ao inventário?
```
inventario = {"canetas": 10,
   "cadernos": 5,
   "borrachas": 7 }
```

a) `inventario.adicionar("lápis", 15)`
b) `inventario.append("lápis", 15)`
c) `inventario["lápis"] = 15`
d) `inventario.inserir("lápis", 15)`

4.4 O que acontece quando tentamos acessar uma chave que não existe no dicionário?
a) Nada acontece.
b) A chave é adicionada com valor None.
c) Ocorre um erro KeyError.
d) O dicionário é impresso.

Atividade prática

4.5 Dada a lista L = [5, 7, 2, 9, 4, 1, 3], escreva um programa que imprima as seguintes informações:
a) quantidade de itens;
b) maior valor da lista;
c) menor valor da lista;
d) soma de todos os elementos da lista;
e) listar em ordem crescente;
f) listar em ordem decrescente.

4.6 Crie uma lista com os nomes dos super-heróis que devem participar da Iniciativa Vingadores seguindo a ordem:
• Homem de Ferro.
• Capitão América.
• Thor.
• Hulk.
• Viúva Negra.
• Gavião Arqueiro.

4.7 No exercício anterior, inclua o Homem-Aranha no final da lista e imprima em qual posição está o Thor.

Tecendo Conhecimento 5

CONDICIONAIS PARA CONTROLE DE FLUXO I

Por que um computador foi inventado? De forma simplista, para fazer coisas repetitivas. Inicialmente operações matemáticas, mas com o passar do tempo, várias outras tarefas também cíclicas puderam ser feitas em computadores.

Para controlar o fim das repetições de tarefas na linguagem de programação são feitas **estruturas de controle de fluxo**, ou simplesmente, **laços de repetição**.

Laço de repetição "if"

Para usar o "if" é preciso fazer uma comparação lógica. Só dois resultados são qualificados no final: Verdadeiro (true), se o parâmetro for de acordo com a lógica aplicada ou Falso, (false) se não cumprir o que foi usado como comparação.

Passada a etapa da comparação, se o resultado for verdadeiro uma sub rotina vai ser executada (que fica dentro do "if"). Uma vez que a lógica apresentar resultado falso, volta ao código principal (pula o "if").

Sintaxe:

```
if "condição":
    "código"
```

Exemplo:

Um programa que identifica se o usuário pode dirigir de acordo com a sua idade:	Saída na IDLE do *Python*:

`idade = int (input ("Qual a sua idade? "))`
`if idade < 18:`
`  print ("Não pode dirigir")`
	`Qual a sua idade? 15`
	`Não pode dirigir`

Fonte elaborada pelo autor.

Laço de repetição if...else

Semelhante ao "if", essa estrutura executa um código quando parâmetro assume o estado de verdadeiro, porém há também um código caso o critério atinja o estado de falso.

Sintaxe:

```
if "condição1"
```

```
        "código1"
else:
        "código2"
```

Exemplo:

Um programa que identifica se o usuário pode ou não dirigir de acordo com a sua idade:	Saída na IDLE do *Python*:

`idade = int (input ("Qual a sua idade? "))`
`if idade < 18:`
`  print ("Não pode dirigir")`
`else:`
`  print ("Você é o cara")`
	`Qual a sua idade?`
	`20`
	`Você é o cara`

Fonte elaborada pelo autor.

Laço de repetição if...elif...else.

A linguagem de programação Python oferece um recurso para ter vários "if...else". Seria acrescentar "elif" que recebe novos parâmetros e refina as possibilidades de análise da variável de entrada.

Sintaxe:

```
if "condição1":
        "código1"
elif "condição2":
        "código2"
elif "condiçãoN":
        "códigoN"
else:
        "código de default"
```

Exemplo:

Um programa que identifica se o usuário pode ou dirigir e votar de acordo com a sua idade:	Saída na IDLE do *Python*:

`idade = int (input ("Qual a sua idade? "))`
`if idade < 16:`
`  print ("Não pode dirigir e não pode votar")`
`elif idade >=16 and idade < 18:`
`  print ("Pode votar mas não pode dirigir")`
`elif idade >= 18`
`  print ("Pode votar e pode dirigir")`
	`Qual a sua idade? 17`
	`Pode votar, mas não pode dirigir`

Fonte elaborada pelo autor.

Roteiro de Atividades 5

5. Analise o seguinte código que afere duas notas e apresenta um resultado para o usuário.

```
nota1 = float (input("Primeira nota: "))
nota2 = float (input("Segunda nota: "))
media = (nota1+nota2)/2
if media >= 6:
        print("Aprovado")
else:
        print("Reprovado")
```

Marque como verdadeiro (V) ou falso (F):

( ) Ambas as notas só podem ser números inteiros;
( ) Um aluno que teve nota 4 e 8 respectivamente, a média apresentada pelo software será 6.
( ) A nota máxima de um aluno pode ser acima de 10.
( ) Todos/as os/as alunos/as que obtiverem pelo menos uma das notas sendo 6, alcança o status de "Aprovado/a".

5.1. O que será impresso quando o código abaixo for executado?

```
numero = 5
if numero > 3:
print("O número é maior que 3")
```

a) O número é maior que 3.
b) O número é menor que 3.
c) Nada será impresso.

d) O código dará erro.

5.2. Qual das opções abaixo mostra a forma correta de verificar se um número é par ou ímpar?

a
```
if numero % 2 == 0:
print("O número é par")
else:
print("O número é ímpar")
```

b
```
if numero % 2 = 0:
    print("O número é par")
else:
    print("O número é ímpar")
```

c
```
if numero / 2 == 0:
    print("O número é par")
else:
    print("O número é ímpar")
```

d
```
if numero % 2 == 0:
    print("O número é par")
else:
    print("O número é ímpar")
```

5.3. O que será impresso quando o código abaixo for executado?
```
nota = 75
if nota >= 90:
    print("A")
elif nota >= 80:
    print("B")
elif nota >= 70:
    print("C")
else:
    print("D")
```
a) A b) B c) C d) D

Atividade prática

5.4. Faça um programa que pergunte o preço de três produtos e informe qual produto você deve comprar, sabendo que a decisão é sempre pelo mais barato.

5.5. Faça um Programa que peça os 3 lados de um triângulo. O programa deverá informar se os valores podem ser um triângulo. Indique, caso os lados formem um triângulo, se o mesmo é: equilátero, isósceles ou escaleno.
Dicas: Três lados formam um triângulo quando a soma de quaisquer dois lados for maior que o terceiro;
Triângulo Equilátero: três lados iguais;
Triângulo Isósceles: quaisquer dois lados iguais;
Triângulo Escaleno: três lados diferentes;

Tecendo Conhecimento 6

CONDICIONAIS PARA CONTROLE DE FLUXO II

Laço de repetição while

Imagine que você precisa fazer um processo de repetição no qual você não conhece quantas ocorrências vão acontecer. Para isso usamos a condicionante “while”.

Sintaxe:
```
while “condição”:
        “código”
```
Exemplo:

Um código de acesso mediante a cadastro e reconhecimento de senha:	Saída na IDLE do *Python*:

`senha = input (“Cadastre a senha: “)`
`leitura = input (“Digite a senha: “)`
`while leitura != senha:`
`  leitura = input (“Digite a senha correta: ”)`
`  if leitura == senha:`
	`Cadastre a senha:`
	`123456`
	`Digite a senha correta:`
	`abcde`
	`Senha Incorreta`

`print ("Acesso liberado")`
`else:`
`print ("Senha Incorreta")`
	`Digite a senha correta:`
	`654321`
	`Senha Incorreta`
	`Digite a senha correta:`
	`123456`
	`Acesso liberado`

Fonte elaborada pelo autor.

Laço de repetição for

Usamos o controle de fluxo "for" para iterar (buscar dentro de uma coleção de forma sequenciada) um resultado ou valor. Então imagine a sua agenda telefônica do celular. Ao buscar um contato específico usando um laço "for" uma variável vai percorrer todas as células da lista em busca de uma chave igual ao nome desejado.

Sintaxe:

```
for "condição" in "coleção":
    "código"
```

Exemplo:

Um código que verifica se um valor inteiro está presente numa lista	Saída na IDLE do *Python*:

`lista = {1,3,5,7}`
`numero = int(input("Número: "))`
`for i in lista:`
`if i == numero:`
`print("Tem")`
	`Número: 7`
	`Tem`

Fonte elaborada pelo autor.

Função range

A função range(m, n, p) é muito útil em laços pois retorna uma lista de inteiros começando em m, menores que n e em passos de comprimento p. Essa função é muito útil para estabelecer listas e sequências organizadas de forma fácil e acessível.

Sintaxe:

```
range ("menor_valor", "maior_valor", "passo")
```

Exemplo:

Programa que exibe múltiplos de 3 entre 6 e 25:	Saída na IDLE do *Python*:

`i = 0`
`for i in range (6, 25, 3):`
`print (i)`
	`6`
	`9`
	`12`
	`15`
	`18`
	`21`
	`24`

Fonte elaborada pelo autor.

Roteiro de Atividades 6

6. Qual o resultado do código abaixo se o usuário inserir o número 5?

```
numero = int(input("Insira um número inteiro positivo: "))
contador = 1
while contador <= numero:
    print(contador)
    contador += 1
```

a) 0 1 2 3 4 5
b) 1 2 3 4 5 6
c) 1 2 3 4 5
d) 5 4 3 2 1

6.1. Qual será a saída do seguinte código Python?

```
for i in range(1, 6):
print(i * 2)
```

a) 2 4 6 8
b) 1 2 3 4 5
c) 1 3 5 7 9
d) 2 4 6 8 10

6.2. Qual será a saída do programa abaixo se o usuário inserir o número 4?

```
numero = int(input("Insira um número inteiro positivo: "))
soma = 0
for i in range(1, numero + 1):
      soma += i
print("A soma dos números de 1 até", numero, "é", soma)
```

a) A soma dos números de 1 até 4 é 6
b) A soma dos números de 1 até 4 é 10
c) A soma dos números de 1 até 4 é 15
d) A soma dos números de 1 até 4 é 20

Atividade prática

6.3. Faça um programa que peça uma nota, entre zero e dez. Mostre uma mensagem caso o valor seja inválido e continue pedindo até que o usuário informe um valor válido.

6.4. Supondo que a população de um país A seja da ordem de 80.000 habitantes com uma taxa anual de crescimento de 3% e que a população de B seja 200.000 habitantes com uma taxa de crescimento de 1.5%. Faça um programa que calcule e escreva o número de anos necessários para que a população do país A iguale a população do país B, mantidas as taxas de crescimento.

Tecendo Conhecimento 7

FUNÇÃO

Um código pode ser idealizado para que uma parte dele seja repetida várias vezes. Por exemplo, deseja-se verificar a temperatura diversas vezes ao longo do dia para se ter uma média. No lugar de reproduzir dentro do escopo a mesma instrução diversas vezes, é possível fazer uma função que contém esses comandos e chamá-la por meio de uma instrução que ocupa apenas uma linha. E assim, elaborar um programa mais elegante e fácil de ser entendido.

Sintaxe:

```
def "nome_da_função" ("parâmetro"):
    "Bloco de códigos"
    return "objeto"
```

Exemplo:

Um código que recebe dois valores e devolve a soma deles (usando função):	Saída na IDLE do *Python*:

`x = float(input("Digite um número: "))`
`y = float(input("Digite outro número: "))`
 	
`def soma (x,y):`
`  total = x + y`
`  return total`
 	
`print ("Soma = %.2f" %(soma(x,y)))`
	`Digite um número: 8`
	`Digite outro número: 4`
	`Soma = 12.00`

Fonte elaborada pelo autor.

Roteiro de Atividades 7

7.1. O que é uma função em Python?

a) Um bloco de código que executa uma tarefa específica e pode ser reutilizado.
b) Um tipo de variável que armazena números.
c) Um comando para desenhar na tela.
d) Um laço de repetição que executa código várias vezes.

7.2. Considere a função abaixo:

```
def diga_ola():
    print("Olá, mundo!")
```

O que acontece quando você chama diga_ola() no seu código?

a) Retorna o valor 10.
b) Desenha uma linha na tela.
c) Exibe "Olá, mundo!" na tela.
d) Nada acontece.

7.3. Observe a função abaixo:

```
def somar(a, b):
    return a + b
```

Qual será o resultado de somar(3, 4)?

a) Erro no código
b) 34
c) 12
d) 7

Referências Bibliográficas

Python Software Foundation. Python. Disponível em: https://www.python.org. Acesso em: 30 nov. 2023.

Python para Desenvolvedores / Luiz Eduardo Borges. Rio de Janeiro, Edição do Autor, 2010.

Franco, João L. . INTRODUÇÃO À PROGRAMAÇÃO COM PYTHON, Programa de Educação Tutorial. Grupo PET - ADS IFSP - Câmpus São Carlos. 2010.

Botcity. "Bibliotecas Python: uma introdução para desenvolvedores". Blog Botcity, 15 de janeiro de 2024. Disponível em: https://blog.botcity.dev/pt-br/2024/01/15/bibliotecas-python/.

Acesso em: 16/04/2024;

Python Brasil. "Games - Python Brasil". Python Brasil, [s.d.]. Disponível em: https://python.org.br/games/. Acesso em: 16/04/2024.

Curso de Introdução à Programação de Jogos em Python, LabTIME, UFG, 2017.

Araújo, Stenio L. Exercícios resolvidos utilizando Python. Universidade Estadual do Sudoeste da Bahia. Série de Textos Didáticos. Edições UESB. Julho, 2023. Disponível em: https://drive.google.com/file/d/1S-ecrRhdVO28mdAWJ7qe4aN_sluqrBMi/view?usp=sh aring

Oxford Languages. Google Dictionary. Disponível em: https://languages.oup.com/google-dictionary-pt. Acesso em: 19 dez. 2023.

Alura. Trabalhando com o dicionário no Python. Disponível em: https://www.alura.com.br/artigos/trabalhando-com-o-dicionario-no-python?utm_term=&utm_campaign=%5BSearch%5D+%5BPerformance%5D+-+Dynamic+Search+Ads+-+Artigos+e+Conteúdos&utm_source=adwords&utm_medium=ppc&hsa_acc=7964138385&hsa_cam=11384329873&hsa_grp=111087461203&hsa_ad=662261158752&hsa_src=g&hsa_tgt=aud-574826424850:dsa-843358956400&hsa_kw=&hsa_mt=&hsa_net=adwords&hsa_ver=3&gad_source=1&gclid=CjwKCAiAu9yqBhBmEiwAHTx5pyuXphAw8VWVMA--mFIeIhXntDR6ty5JW0ldiDt-IKUXc7TEa4Z8ihoCy0EQAvD_BwE. Acesso em: 17 nov. 2023.

Python Academy. A função range() do Python. Disponível em: https://pythonacademy.com.br/blog/a-funcao-range-do-python. Acesso em: 20 nov. 2023.

Linux Console. Disponível em: https://pt.linux-console.net/?p=26498.

PEREIRA, Fernando. Exercícios resolvidos de algoritmos. Março de 2007. Disponível em:

https://fit.faccat.br/~fpereira/apostilas/exerc_resp_alg_mar2007.pdf. Acesso em: 26 jul. 2024.

UNIVERSIDADE ESTADUAL DE CAMPINAS. Olimpíada Brasileira de Informática. Curso introdutório de JavaScript. Disponível em: https://olimpiada.ic.unicamp.br/saci/cursos/intro_js/1/. Acesso em: 26 jul. 2024.

https://fit.faccat.br/~fpereira/apostilas/exerc_resp_alg_mar2007.pdf

Secretaria de Educação e Esportes
GOVERNO DE PERNAMBUCO
ESTADO DE MUDANÇA