Aprendendo com a Tartaruga

Pensamento computacional com módulo turtle do Python

Pensamento computacional

Uma abordagem para solucionar problemas de maneira eficaz e criativa a partir de fundamentos da computação.

Pilares do Pensamento computacional:

1. Decomposição - Dividir para conquistar;
2. Padrões - Identificar semelhanças e regularidade nos problemas;
3. Abstração - Simplificar os problemas, focando no que importa;
4. **Algoritmos** - (O passo a passo para resolver o problema) Sequências de passos lógicos para resolver um problema.

É possível olhar para situações diferentes e aplicar os mesmos princípios para encontrar uma solução.

O que são Algoritmos?

Algoritmos são sequências lógicas e finitas de instruções, geralmente aplicadas para resolver um problema ou realizar uma tarefa específica. Eles são a base da programação de computadores e são usados para descrever a lógica de um processo passo a passo.

Um algoritmo pode ser comparado a uma receita de cozinha: ele fornece uma série de instruções claras e ordenadas para realizar uma tarefa específica. No contexto da computação, os algoritmos são essenciais para processar informações e executar operações em dados.

Algoritmos

Algoritmo: Bolo de Chocolate

- **Passo1 – Receber os ingredientes:**
- 2 xícaras de açúcar;
- 3 ovos;
- 250g de margarina;
- 3 xícaras de farinha de trigo;
- 1 e ½ colher de fermento;
- 1 xícara de leite.
- **Passo 2: aqueça o forno a 180 graus;**
- **Passo 3: bata as claras em neve e reserve;**
- **Passo 4: em uma travessa, bata o açúcar, a manteiga e as gemas;**

```
e > anamaria > Documentos > Coding > Python > exercicios >
nums = []
par = []
impar = []

for i in range(20):
    x = int(input('informe o número: '))
    nums.append(x)

for num in nums:
    if (num % 2) == 0 or num == 0:
        par.append(num)
    else:
        impar.append(num)

print(f'Números: {nums}, números pares: {par}
```

Linguagem de Programação

É uma linguagem formal que os programadores usam para criar instruções que um computador pode entender e executar. É um meio de comunicação entre seres humanos e máquinas, permitindo que os desenvolvedores escrevam código que descreve as ações que um computador deve realizar.

Algoritmos são escritos usando linguagens de programação

Linguagens de Alto Nível : são foram projetadas para ser mais compreensível e fácil de usar para os programadores, essas linguagens são mais abstratas e permitem que os desenvolvedores escrevam código de forma mais próxima da linguagem humana, facilitando o entendimento e a manutenção do código.

Linguagem de Baixo Nível (Linguagem de máquina): o computador só consegue executar programas escritos em linguagens de baixo nível. Deste modo, programas escritos em linguagens de alto nível precisam ser **processados** antes que possam rodar.

Traduzir para a linguagem de máquina

Interpretadas

Compiladas

A linguagem de programação Python

- Alto Nível;
- Sintaxe simples e Legível;
- Interpretada e Tipagem Dinâmica;
- Multiplataforma;
- Comunidade Ativa!

Bora codar!

Variáveis e tipos de dados

Tipos de Dados

Um *valor* é uma das coisas fundamentais — como uma palavra ou número — que um programa manipula.

TIPOS: int, string, float (ponto flutuante), booleano.

```
integer_value = 42
negative_integer = -10
zero = 0

pi = 3.14
negative_float = -0.5
floating_point = 2.0

hello_string = "Hello, World!"
name = 'Alice'

is_true = True
is_false = False
```

Variáveis

Uma variável é um nome que se refere a um valor ou um espaço reservado na memória.

Atribuição de uma variável:

```
3   mensagem = "Que hora isso termina?"
4   n = 13
5   pi = 3.14159
```

O operador de atribuição, =, não deve ser confundido com *igualdade*, para a qual usamos ==. O comando de atribuição associa o *nome*, que está à esquerda do operador, como o *valor*, que está à direita.

Comandos e Expressões

Um **comando** (*statement*) é uma instrução que o interpretador Python pode executar. Até agora só vimos o comando de atribuição. Outros tipos de comando que veremos em breve são o comando *while*, o comando *for*, o comando *if* e o comando import.

Uma **expressão** (*expression*) é uma combinação de valores, variáveis, operadores e chamadas de funções. Expressões necessitam ser calculadas.

Operadores

Operadores são símbolos especiais que representam computações como adição, multiplicação e divisão.

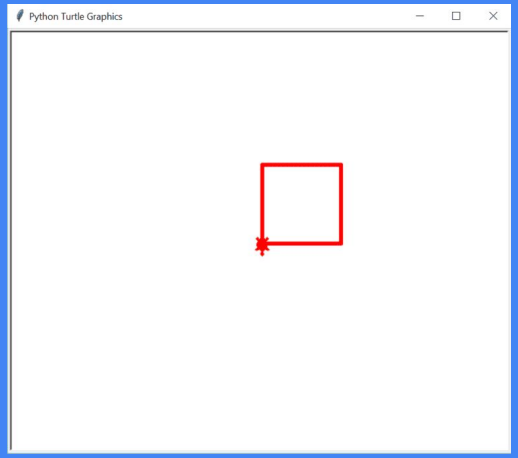
Operadores aritméticos

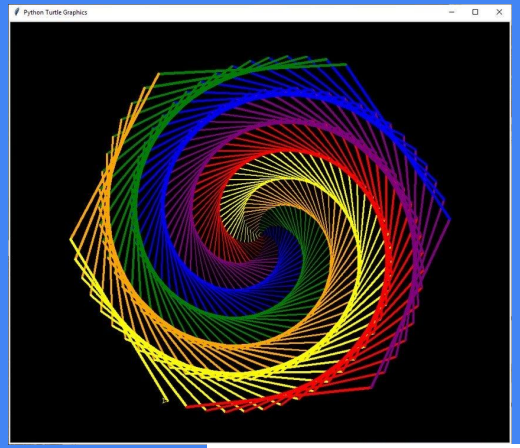
OPERADOR	OPERAÇÃO
+	Adição
-	Subtração
*	Multiplicação
**	Exponenciação
/	Divisão

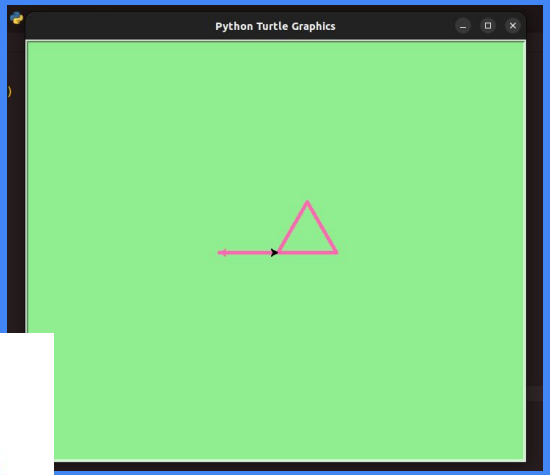
Cadê a tartaruga, mermã?!

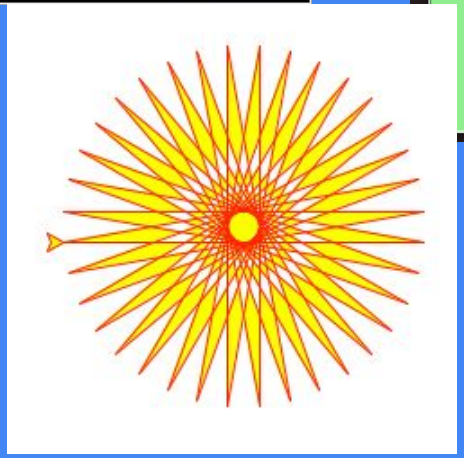
módulo turtle

um módulo que nos permite criar um objeto chamado turtle que pode ser usado para desenhar figuras (gráfico de tartarugas).


Python Turtle Graphics


Python Turtle Graphics


Python Turtle Graphics



Minha primeira tartaruga

CRIANDO E DANDO FORMA

turtle.Screen() - cria a janela

turtle.st() - mostra a tartaruga

turtle.shape([*forma*]) : 'arrow', 'turtle',
'circle', 'square',
'triangle', 'classic'

MOVIMENTANDO A NOSSA TARTARUGA

Pra frente

- turtle.forward( )

O quanto sua tartaruga deve se mover
deve ser um valor inteiro

Pra trás

- turtle.backward( )
- turtle.back( )
- turtle.bk( )

Girando a tartaruga

turtle.Screen() - cria a janela

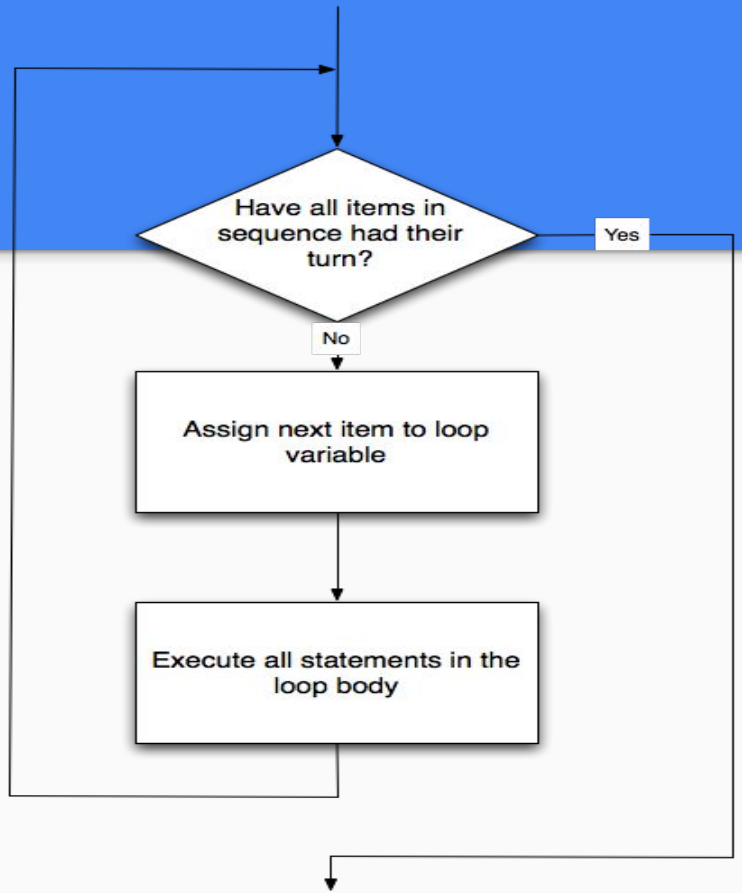
turtle.st() - mostra a tartaruga

turtle.shape([*forma*]) : 'arrow', 'turtle',
'circle', 'square',
'triangle', 'classic'

Comandos de repetição

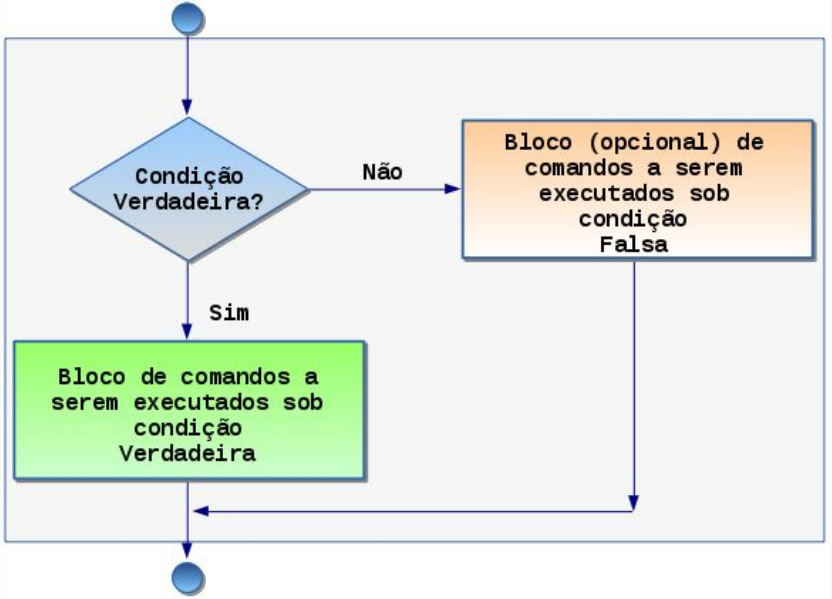
O laço for

o comando for nos permite escrever programas que implementam iterações



Comando IF

```
if <condição>:
    <comando1>
else:
    <comando2>
```



REFERÊNCIAS

https://docs.python.org/pt-br/3/library/turtle.html#compound-shapes

https://panda.ime.usp.br/pensepy/static/pensepy/03-PythonTurtle/olatartaruga.html