STEM²D

Go girls!

CODE

like a girl

SEJAM BEM-VINDAS!

Nós somos:

Fernanda, Isabela, Letícia, Lívia, Maihara, Stéfanie Thayná

Prof **Lara**

Ajudantes:

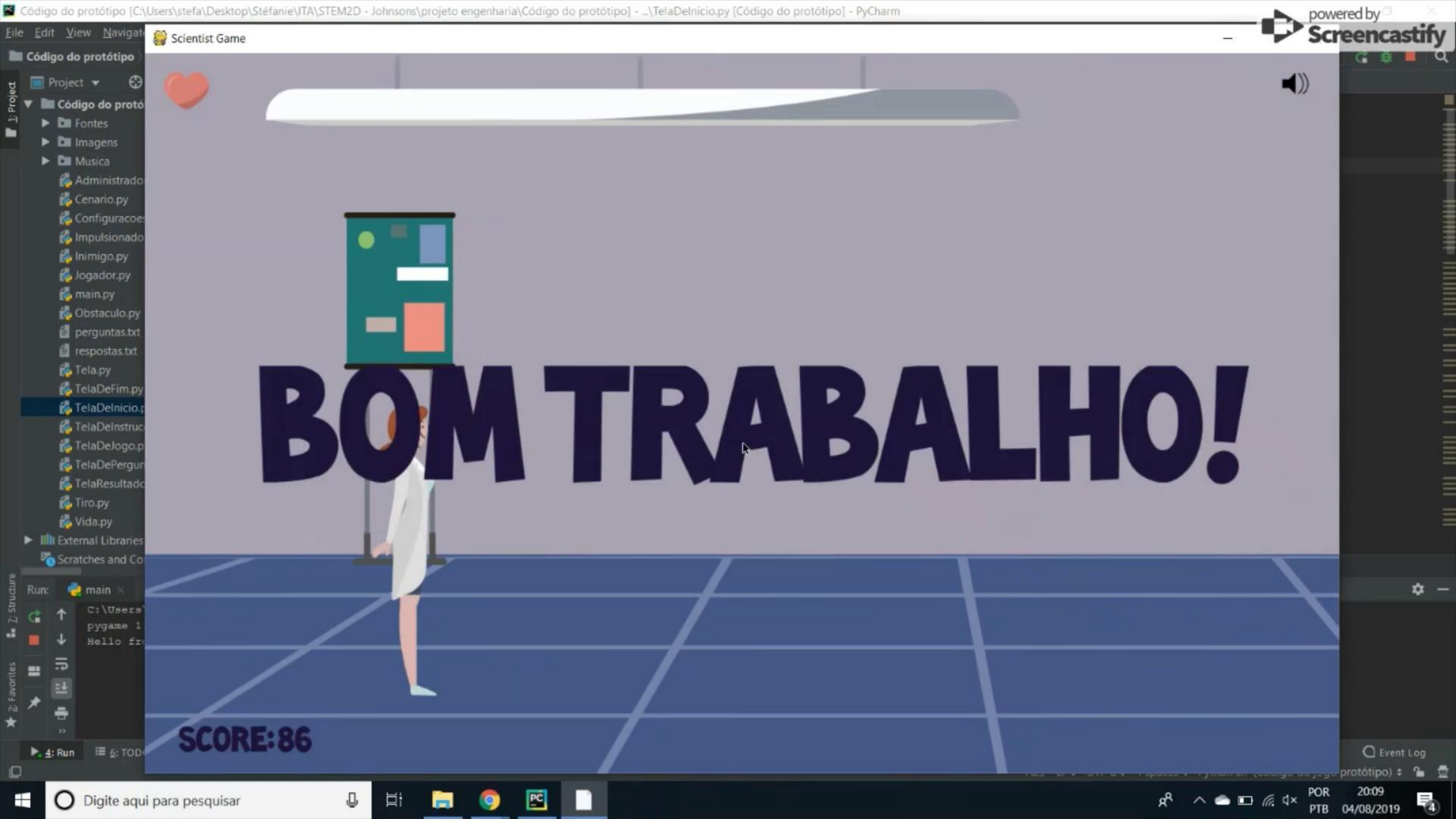
Débora

Dallyane

Letícia Silva

Raíssa

Go girls!


Código do protótipo [C:\Users\stefa\Desktop\Stéfanie\ITA\STEM2D - Johnsons\projeto engenharia\Código do protótipo] - ...\TelaDeInicio.py [Código do protótipo] - PyCharm
powered by Screencastify
Scientist Game
BOM TRABALHO!
SCORE:86
Digite aqui para pesquisar

PROGRAMAÇÃO DO CURSO

- 24/08: Introdução à Programação
- 31/08: Introdução à Programação + Desenvolvimento da tela de início e tela de fim
- 07/09: Desenvolvimento da tela de perguntas e da tela de instruções
- 28/09: Desenvolvimento da tela de jogo + Implementação do aparecimento de obstáculos na tela + Implementação do sistema de score + Desenvolvimento do modo batalha

PROGRAMAÇÃO DO CURSO

- 05/10: Implementação dos movimentos do jogador e dos inimigos
- 19/10: Implementação do movimento do tiro + Implementação da colisão do jogador com o inimigo e do jogador com o tiro
- 26/10: Implementação da colisão do jogador com a vida, do jogador com o impulsionador e do jogador com os obstáculos
- 09/11: Sonorização e correção de bugs

COMPROMISSOS

- Assiduidade

COMPROMISSOS

- Assiduidade
- Pontualidade

COMPROMISSOS

- Assiduidade
- Pontualidade
- Silêncio durante as explicações

COMPROMISSOS

- Assiduidade
- Pontualidade
- Silêncio durante as explicações
- **Não** ficar com dúvidas

COMPROMISSOS

- Assiduidade
- Pontualidade
- Silêncio durante as explicações
- **Não** ficar com dúvidas
- **Sempre acreditar que você consegue!**

O QUE É PROGRAMAÇÃO?

- É o ato de escrever códigos que instruem um computador a realizar uma tarefa.

- Essa tarefa pode ser tão simples quanto somar dois números ou tão complexa quanto enviar uma nave espacial para a órbita!

LÓGICA DE PROGRAMAÇÃO

- É a técnica de encadear ideias para atingir determinado objetivo.
- Essas ideias são representadas como **instruções**, isto é, informações que indicam a um computador uma ação elementar a executar.
- Para obtermos um resultado, precisamos colocar em prática o conjunto de todas as instruções, na ordem correta.

LÓGICA DE PROGRAMAÇÃO

- Para representar essa sequência lógica de instruções, nós fazemos uso de **algoritmos**.

- Algoritmos são uma espécie de "receita" para se executar uma tarefa ou resolver algum problema.

- Embora às vezes não percebamos, utilizamos algoritmos no nosso dia-a-dia e não sabemos. Um exemplo é o algoritmo do cinema.

LÓGICA DE PROGRAMAÇÃO

Algoritmo 1 Recepcionista

```
início
    Solicitar ao cliente o bilhete do filme.
    Conferir a data e o horário do filme no bilhete.
    se data/hora atual > data/hora do filme + 30 minutos então
        Informar ao cliente que o tempo limite para entrada foi excedido.
        Não permitir a entrada.
    senão se data/hora atual < data/hora do filme - 30 minutos então
        Informar ao cliente que a sala do filme ainda não foi liberada para entrada.
        Não permitir a entrada.
    senão
        Permitir a entrada.
        Indicar ao cliente onde fica a sala do filme.
    fim
fim
```

LÓGICA DE PROGRAMAÇÃO

- Os algoritmos são **finitos** porque, para que sejam úteis, uma hora eles tem que acabar!

- Além disso, eles devem ser **não-ambíguos** pois o computador não pode ter dúvidas do que fazer para executar cada passo. Do contrário, erros podem acontecer.

- Todas as tarefas executadas pelo computador, são baseadas em algoritmos.

LINGUAGENS DE PROGRAMAÇÃO

- É um método padronizado para expressar instruções para um computador, contendo um conjunto de **regras sintáticas e semânticas** usadas para definir um programa.

- Existem diversas linguagens de programação.

LINGUAGENS DE PROGRAMAÇÃO

- O algoritmo escrito de acordo com as regras de uma linguagem de programação, constitui o **código-fonte** de um software.

- Depois de escrevê-lo, um compilador o converte em **código de máquina**, isto é, a linguagem de nível mais baixo para um computador.

- Então, o código de máquina instrui a unidade central de processamento (**CPU**) sobre as etapas a serem seguidas, como carregar um valor ou executar alguma aritmética.

LINGUAGENS DE PROGRAMAÇÃO

- Mas se o computador interpreta apenas código de máquina, **por que não escrevemos código de máquina diretamente?**
- A razão é que o código de máquina é ilegível para humanos.

```
1  print('Hello, World!')
```

Código em Python

```
1  c7 3c 2a 3c 2a 2b 2a 5c 3c 28 5c 2a 2b 2a 5c 3c
2  28 5c 2a 2b 2a 5c 3c 28 5c 2a 2b 2a 5c 3c 28 5c
3  2a 2b 2a 5c 3c 28 5c 2a 2b 2a 5c 3c 28 5c 2a 2b
4  2a 5c 3c 28 5c 2a 2b 2a 5c 3c 28 5c 2a 2b 2a 5c
5  3c 28 5c 2a 2b 2a 5c 3c 28 5c 2a 2b 2a 5c 3c 28
6  5c 2a 2b 2a 5c 3c 28 5c 2a 2b 2a 5c 3c 28 5c 2a
7  2b 2a 00 00 01 00 00 00 00 00 00 00 00 00 00 00
8  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
9  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
10 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
11 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
12 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
13 00 00 00 00 00 00 00 64 48 65 6c 6c 6f 2c 20 57
14 6f 72 6c 64 21 00 00 00 00 00 00 00 00 00 00 00
15 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
16 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
17 Hello, World!
```

Código de máquina

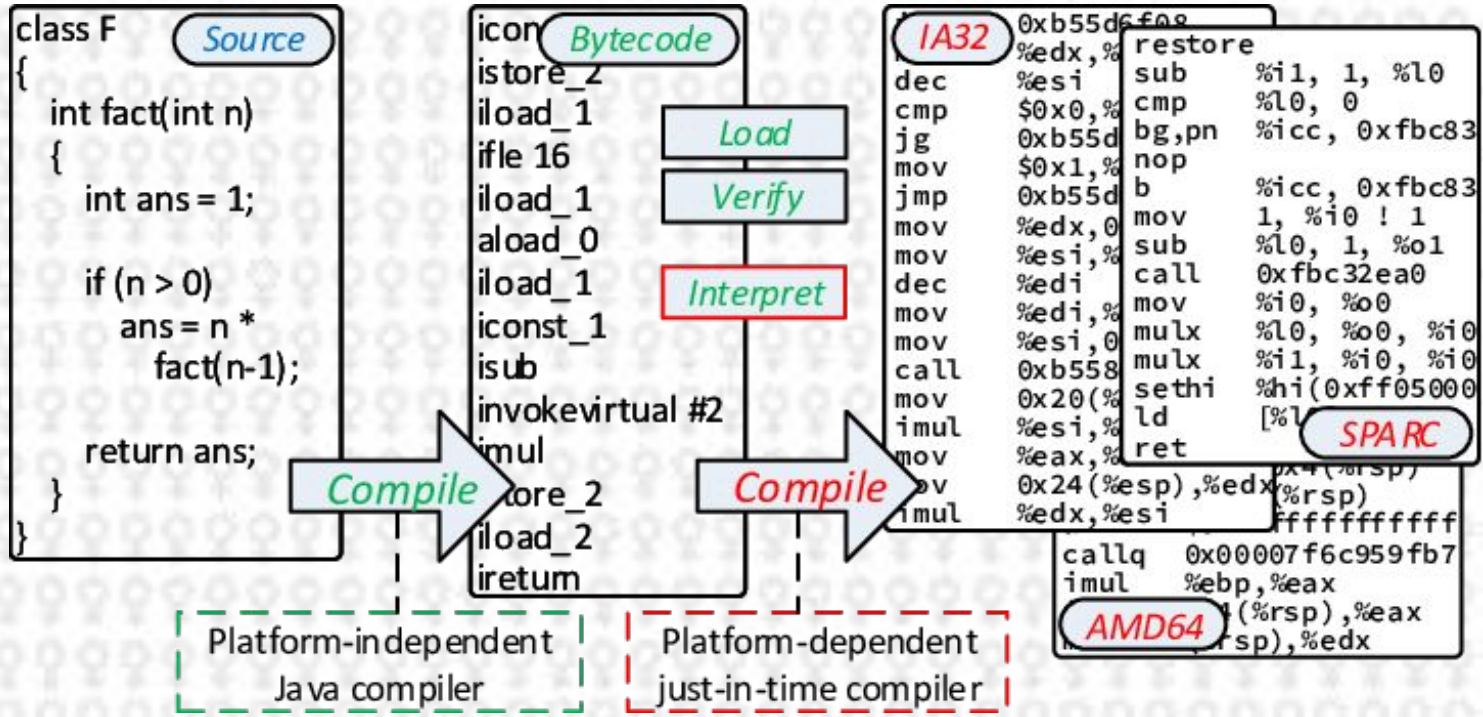
PYTHON

- Python é uma linguagem de programação criada por Guido van Rossum, em 1991 com objetivo de ser uma linguagem que proporcionasse **produtividade** e **legibilidade**

- Python tem uma **biblioteca padrão** imensa, que contém classes, métodos e funções para realizar essencialmente qualquer tarefa. Por causa de todas essas vantagens, esta será a linguagem utilizada neste curso.

PYTHON

- Um detalhe do Python é que ele é uma linguagem interpretada, isto é, você não a compila diretamente em código de máquina, ao invés disso, ele usa algo um **interpretador**.



- O interpretador é outro programa que compila o código em algo chamado Bytecode, que é traduzido para linguagem de máquina enquanto o programa é executado.

IDE

- Uma opção para desenvolver programas seria escrever o código em Python em um editor de texto qualquer e salvar o arquivo com a extensão .py e executá-lo no terminal. Porém você só irá ser notificado que algo está errado quando ele parar de funcionar durante a execução.

- Com o intuito de facilitar a vida do desenvolvedor, criou-se o IDE que é um software que contém todas as funções necessárias para o desenvolvimento de programas de computador, assim como alguns recursos que diminuem a ocorrência de erros nas linhas de código.

IDE

- Neste curso utilizaremos o *IDE PyCharm*. Ele possui uma interface muito limpa e personalizável, ideal para aqueles que estão dando os primeiros passos com a linguagem *Python*.

IDE

- As características e ferramentas mais comuns encontradas nos IDEs são:

1. **Editor**: edita o código-fonte do programa;
2. **Compilador**: compila o código-fonte do programa, e o transforma em linguagem de máquina;
3. **Linker**: liga ("linka'") os vários "pedaços" de código-fonte, compilados em linguagem de máquina, em um programa executável;
4. **Depurador** (*debugger*): auxilia no processo de encontrar e corrigir defeitos no código-fonte do programa;
5. **Testes Automatizados**: realiza testes no software de forma automatizada, gerando um relatório, e assim auxiliando na análise do impacto das alterações no código-fonte;

IDE

- As principais vantagens de se utilizar um IDE durante o desenvolvimento de código são:

1. Notifica em tempo real erros de sintaxe;
2. Faz sugestões para correção de erros;
3. Auto-completa códigos;
4. Permite "debugar" o programa;
5. Altera cores de palavras-chaves da linguagem no seu código, tornando-o mais legível;
6. O processo de execução do programa pode ser feito com o *click* de um botão e o *output* visualizado diretamente na IDE;

ERROS DE DESENVOLVIMENTO

- É natural que, durante o desenvolvimento de programas para o computador, nos deparemos com alguns erros que precisam ser corrigidos para o correto funcionamento do programa.

ERROS DE DESENVOLVIMENTO

- Em geral, os erros são causados por dois principais motivos:

1. Escrita incorreta da sintaxe da linguagem de programação;

1. Algoritmo com falhas lógicas, que fazem com que o comportamento do programa divirja do esperado.

UTILIZAÇÃO DE VARIÁVEIS

- **Atribuição de variáveis**

Assim como em outras linguagens, o *Python* pode manipular variáveis básicas como palavras ou cadeias de caracteres (**string)**, números inteiros (**int**) e números reais (**float**).

```
mensagem = 'Exemplo de mensagem!'
n = 25
pi = 3.141592653589931
```

- **Tipo de variável**

Não é necessário declarar o tipo da variável. Pode-se utilizar a função **type()** para retornar o tipo da variável.

NOMENCLATURA DE VARIÁVEIS

- As variáveis podem ser nomeadas conforme a vontade do programador (lembrando-se de utilizar nomes significativos);
- No entanto, elas devem necessariamente começar com letras **minúsculas**;
- Atentar-se para as palavras reservadas da linguagem:

False	class	finally	is	return
None	continue	for	lambda	try
True	def	from	nonlocal	while
and	del	global	not	with
as	elif	if	or	yield
assert	else	import	pass	
break	except	in	raise	

FUNÇÃO IMPRESSÃO NA TELA

- Exibir coisas na tela é, sem dúvida, a comunicação mais básica e uma das mais importantes.
- Isso é feito através da impressão de informações na tela, por meio da função **print**.
- A grosso modo a função **print** serve para imprimir os argumentos passados a ela no terminal. Basicamente, para utilizá-la, basta escrever print(< o que se deseja imprimir >).

FUNÇÃO IMPRESSÃO NA TELA

- Para escrever um texto, basta colocar entre aspas simples ou aspas duplas o que se deseja imprimir.

```
print('Ola, mundo!')
```

- **É importante ressaltar que as linguagens de programação, em geral, não funcionam com acentos ortográficos, de modo que você não deve utilizá-los em seus códigos em Python.**

STEM2D

Go girls!

Code like a girl!

EXERCÍCIO

- Faça um programa que imprima seu nome.

FUNÇÃO IMPRESSÃO NA TELA

- Por outro lado, se você quiser imprimir o valor armazenado em uma variável, deve escrever o nome desta dentro do comando **print**.

```
print(nomeDaVariavel)
```

EXERCÍCIO

- Faça um programa que guarde seu nome em uma variável e a imprima.

FUNÇÃO IMPRESSÃO NA TELA

- Se for necessário imprimir o valor de uma variável no meio de um texto, é preciso avisar ao código o que se deseja imprimir.
- Para isso, colocamos o texto entre aspas simples e, no lugar onde vai ser impresso o valor da variável, coloca-se um abrir e fechar de chaves (**{ }**).
- Depois de fechar as aspas simples do texto, coloque o método **.format()** e, como argumento, a variável que deseja imprimir.

```
print('Meu nome eh {}'.format(nome))
```

FUNÇÃO IMPRESSÃO NA TELA

- Para imprimir mais de uma variável no texto, basta colocá-las como argumento, na ordem em que elas devem aparecer.

```
print('Meu nome eh {} e eu tenho {} anos'.format(nome, idade))
```

- Se tivermos uma variável que armazena um número real (tipo **float**, por exemplo), você pode colocar dentro das chaves que indicam a variável o número de casas decimais que você deseja imprimir: **{:.(<número de casas decimais>)f}.**

```
altura = 1.6789
print('Minha altura eh {:.2f}m'.format(altura))
```

FUNÇÃO IMPRESSÃO NA TELA

- Se você quiser pular uma linha, basta colocar um **print** vazio, ou ainda, escrever a sequência de caracteres **\n** entre aspas, ou ainda, dentro de um texto.

```
print('\n')
print()
print('Vamos pular uma linha.\nE mais uma linha.\n')
```

FUNÇÃO LEITURA NA TELA

- Além de imprimir, muitas vezes precisamos que o programa leia e armazene coisas.
- Se você quiser que o usuário escreva na tela o valor a ser armazenado em uma variável, utilize algo semelhante ao código abaixo:

```
variavel = input()
```

FUNÇÃO LEITURA NA TELA

- É importante saber que o input sempre guarda uma variável do tipo string. Dessa maneira, se for desejado armazenar outro tipo de variável, é preciso fazer conversões.

```
variavel = int(input('O valor da variavel int eh: '))
```

STEM2D

Go girls!

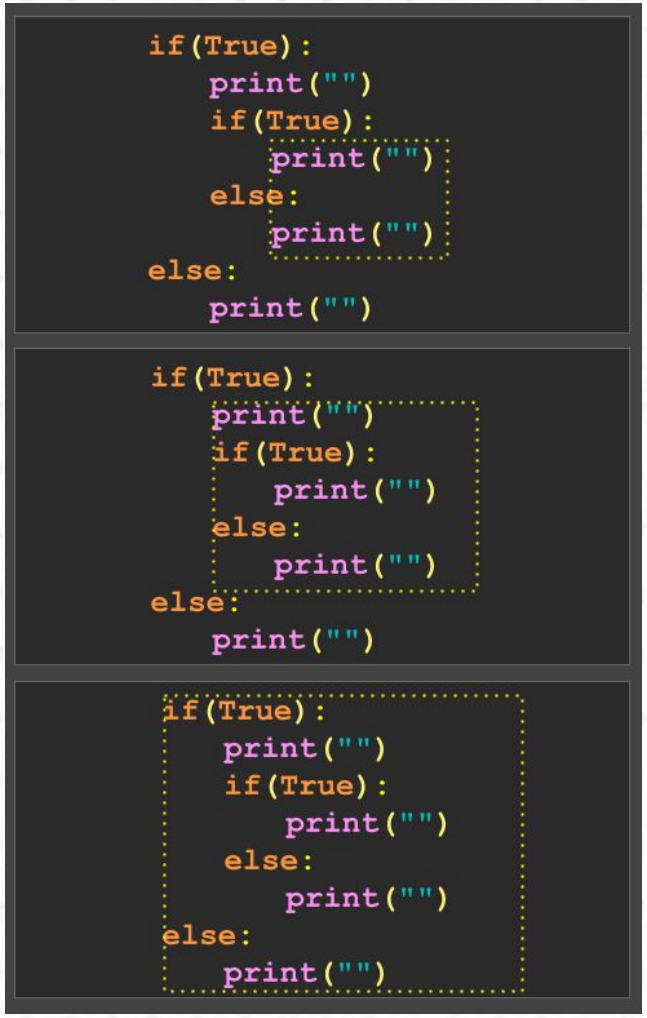
Code like a girl!

EXERCÍCIO

- Faça um programa que requisite ao usuário seu nome e responda “Bom dia, <nome da pessoa>”.

BOAS PRÁTICAS: INDENTAÇÃO

- Consiste na adição de tabulações no início de cada linha.
- O código que está na mesma posição (recuado o mesmo número de espaços da margem esquerda) é agrupado em um bloco.
- Sem uma boa indentação o código **perde a fácil visualização da hierarquia de seus comandos**.



BOAS PRÁTICAS: INDENTAÇÃO

- Em alguns comandos (**if**, **while**, **for**, ...) a indentação é obrigatória, já que substitui os identificadores de blocos.

```
if self.x > -200:
        self.x -= self.largura
    self.rect.x = self.x
    self.rect.y = self.y
```

Código indentado

```
if self.x > -200:
self.x -= self.largura
self.rect.x = self.x
self.rect.y = self.y
```

Código não indentado

- Desta forma, o código acima que não possui indentação, sequer consegue ser executado.

BOAS PRÁTICAS: NOMES SIGNIFICATIVOS

- Colocando nomes adequados e padronizados, você passará informações que ajudarão na compreensão do código. Recomenda-se:

1. Criar nomes curtos e significativos;

1. Nomear as variáveis com o padrão **camelCase**;

1. Em constantes todas as letras devem estar em maiúsculo e com as palavras separadas por sublinhado;

1. Utilizar nomes passíveis de busca ;

1. Preferir declarar nomes pronunciáveis;

BOAS PRÁTICAS: COMENTÁRIOS

- Finalidade: explicar o algoritmo ou a lógica usada.
- Basta utilizar o carácter **#** e escrever seu comentário depois disso.
- **Cuidado!** Os comentários devem ser textos curtos e relevantes: muitos comentários podem deixar o código poluído, difícil de ler e atualizar.

```
def interpretarEventos(self, game):
  game.clock.tick(game.fps)

  for evento in pygame.event.get():
      pos = pygame.mouse.get_pos()

      # checa se o usuario quer sair do jogo
      self.comportamentoBotaoDeSair(game, evento)

      # checa se o usuario quer tirar o som
      self.comportamentoBotaoDeAudio(game,
evento, pos)

      # checa se o usuario clicou no botao para abrir a
tela de instrucoes
      self.comportamentoBotaoDeInstrucoes(game,
evento, pos)

      # checa se o usuario quer jogar
      self.comportamentoBotaoDeJogar(game,
evento, pos)

      # print(``pos0: '', pos[0], `` pos1: '', pos[1])
```

OPERADORES ARITMÉTICOS

- Operadores são símbolos especiais que representam cálculos como adições e multiplicações.

```
+  : ADIÇÃO
-  : SUBTRAÇÃO
*  : MULTIPLICAÇÃO
/  : DIVISÃO
** : POTENCIAÇÃO
%  : MÓDULO
// : DIVISÃO INTEIRA
```

OPERADORES ARITMÉTICOS

- Uma expressão é uma combinação de valores, variáveis e operadores. Quando digitamos uma expressão no modo interativo, o interpretador vai calcular e imprimir o resultado.
- Podemos também utilizar variáveis nessas expressões.

```
1 + 1
2 * 3
```

```
2
6
```

```
7 // 2
7 % 3
```

```
3
1
```

```
x = 1
y = 3

x + y
x - y
x * y
x / y
x ** y
```

```
4
-2
3
0.3333333333333333
3
1
```

OPERADORES LÓGICOS

- **True** e **False** são valores booleanos que representam, respectivamente, verdadeiro e falso.
- A função **bool()** que retorna **True** quando o argumento passado é verdadeiro e retorna **False**, caso contrário.

```
2 > 1
True
2 < 1
False
bool(1 == 1)
True
bool(1 == '1')
False
```

OPERADORES LÓGICOS

- **Operadores de comparação**

Operação	Descrição
a == b	a igual a b
a != b	a diferente de b
a < b	a menor que b
a > b	a maior que b
a <= b	a menor ou igual a b
a >= b	a maior ou igual a b

- Não confundir o operador '==' com o '=' que atribui um valor a uma variável.

OPERADORES LÓGICOS

- Outros operadores que retornam valores booleanos

Operação	Descrição
a is b	True se a e b são idênticos
a is not b	True se a e b não são idênticos
a in b	True se a é membro de b
a not in b	True se a não é membro de b


STEM2D
Go girls!
Muito obrigada!