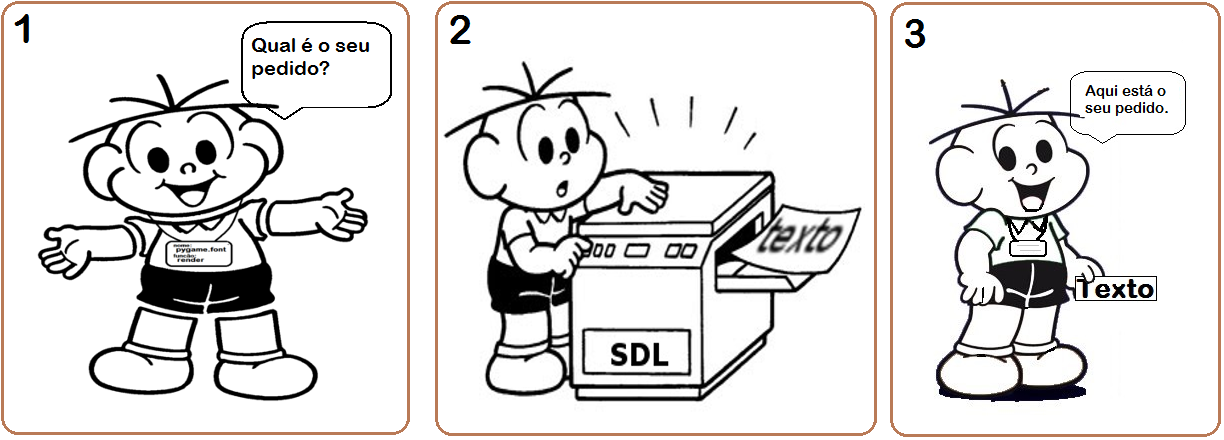
Algumas vezes é necessário imprimir alguma informação na tela do computador. Anteriormente quando nós trabalhávamos somente com o shell do interpretador Python (console), para imprimirmos alguma informação na tela usávamos o comando print. O print imprime somente informação no console não no display (janela) do pygame.

|  |
| --- |
| # importa os módulos do python e pygame necessários  import sys  import pygame  from pygame.locals import \*  pygame.init()  # cria janela  window = pygame.display.set\_mode((200,200))  # imprime uma mensagem  print "Minha mensagem de texto"  while True:  # verifica se o usuário disparou o evento de fechar a janela  for event in pygame.event.get():  if event.type == QUIT:  pygame.quit()  sys.exit()  window.fill((0,0,0))  pygame.display.update() |

A tela permanecerá vazia e sua mensagem “Minha mensagem de texto” será exibida no console, verifique. Para exibir textos no display do pygame usa-se o módulo *Font* da biblioteca do pygame.

Basicamente o que esse módulo faz é renderizar uma string para ser exibida na janela do pygame. Geralmente o termo renderizar (*rendering*) <colocar um glossário com essa palavra link> se refere a processamento gráfico ou de áudio e consiste, de forma bastante genérica, em um determinado processo digital pelo qual se obtém um produto final a partir de um processamento digital qualquer. Em outras palavras, a partir de parâmetros passados para a máquina, ela compila um resultado final em áudio e/ou vídeo. Isso quer dizer que nesse caso você fornece uma string e o pygame processa a string de modo a fornecer esse texto *renderizado* em um impresso em um *surface*.

Podemos pensar que a surface de saída contendo o texto é uma etiqueta que você precisa fabricar para colar em seu mural (que é o display da janela). Você sabe as especificações que quer com as quais você quer produzir o texto: fonte, tamanho, cor do fundo, cor do texto, negrito, itálico, sublinhado. Com isso em mente você vai a uma gráfica para produzir sua etiqueta, lá o funcionário (módulo pygame.font) pede as especificações de como você quer a fonte e isso obviamente deve ser informado **antes** de ele confeccionar sua etiqueta, ou seja, antes dele renderizar sua etiqueta [1]. A função desse funcionário na gráfica chama *render* (renderizador em tradução livre) isso porque ele o que ele faz é pegar suas especificações e passar para a impressora para que seja produzida a etiqueta. A impressora (SDL) imprime sua etiqueta de acordo com o que você pediu nada mais, nada a menos [2]. Então o funcionário pega a etiqueta impressa e te entrega [3]. O que você fará com ela é da sua conta, não mais dele!



Esse é o processo resume como é feito a construção do texto que você quer renderizar. Agora você tem a etiqueta e se não pegar ela e colar no seu mural ela não vai se colar sozinha... Nesse caso você deve pegar e colar sua etiqueta (função surface.blit()) e pronto, lá está ela.

Se você tentou imprimir algum texto com mais de uma linha você pode ver que não é possível simplesmente usando a string com /n para pular a linha.

USAR O LIVRO BEGGINNING GAME DEVELOPPING PARA CONTINUAR AQUI

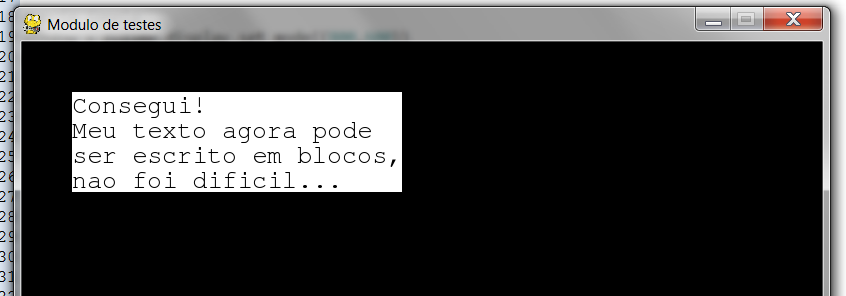
DESCREVER CADA METODO DO MODULO FONT E PENSAR COMO COLOCAR OS EXEMPLOS

EXERCICIOS

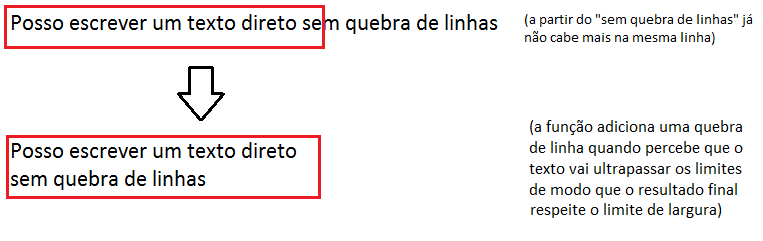
1. Faça um box para inserir texto de modo que fique parecido com esse usando somente o modulo fonte e manipulando a string de entrada para renderização. (<vai precisar do modulo de event pensar nisso>)
2. Carregue a fonte \*\*\* renderize um texto em negrito e sublinhado imprima o texto na tela e imprima também um texto somente em negrito usando A MESMA FONTE sem carrega-la novamente.
3. Escreva uma função letreiro com seu texto fazendo e movimentar lateralmente na tela continuamente, ou seja, ele se movimenta da direita pra esquerda e ao atingir a lateral esquerda da tela recomeça saindo da extremidade direita da tela. Dica: desenhe seu texto na extremidade direita da tela e faça se movimentar para a esquerda quando a posição do texto for zero (alcançar a extremidade esquerda da tela) então renderize novamente um novo texto (com mesmo conteúdo) na extremidade direita e repita o ciclo indefinidamente.
4. Crie um texto e faça ele se movimentar com trajetórias diferentes e não lineares. Tente movimentar seu texto em uma trajetória circular e elíptica por exemplo. Tente outras trajetórias usando o conceito da aula de <movimentando coisas> para testar as possibilidades. Dica: para isso use os vetores e use as funções trigonométricas ou as equações paramétricas da trajetória que você escolher.
5. Tendo em vista que não é possível renderizar textos com mais de uma linha, crie uma função que gera blocos de textos, ou seja, passado uma string texto com mais de uma linha (string com \n) a função gera um superfície bloco\_texto contendo as frases separadas em linhas de acordo com a string de entrada. A função deve ser dessa forma.

|  |
| --- |
| import sys  import pygame  from pygame.locals import \*  pygame.init()  window = pygame.display.set\_mode((800,600))  # SUA FUNCAO  def gera\_bloco\_texto(texto, tamanho = 20, espacamento = 0, cor = (0, 0, 0),  fundo = (255,255,255), tipo\_fonte = 'monospace'):  """  texto (string): texto a ser renderizado em bloco de texto  tamanho (inteiro): tamanho da fonte a ser carregada (default 20)  espaçamento (inteiro): espaçamento desejado entre as linhas (default 0)  cor (tupla): cor do texto a ser renderizado (default preto)  fundo (tupla): cor de fundo do bloco de texto (default branco)  tipo\_fonte (string): tipo de fonte (nome) a ser renderizado  """  # separe o texto de entrada por frases (pense como separar as frases)  # carregue a fonte, defina uma altura e largura para o bloco de texto  # crie uma superficie bloco\_texto para colar esses texto nela  # preencha a superficie com a cor de fundo  # renderize cada frase e cole-as uma por vez na superficie  return bloco\_texto # retorna surface com o texto  #--------------------------------------------------  while True:  for event in pygame.event.get():  if event.type == QUIT:  pygame.quit()  sys.exit() window.fill((0,0,0))  window.blit(gera\_bloco\_texto("Consegui!\nMeu texto agora pode\nser escrito em blocos\nnao foi dificil..."),(100,100))  pygame.display.update() |

A saída deve ser parecida com essa:



1. Crie uma função para formatar um texto dentro de um retângulo com uma largura e altura fixa passada como parâmetro da sua função. O objetivo aqui é passada uma string quebra de textos (\n) e a função a partir da largura fixa a função decide onde deve quebrar o texto em frases para que não ultrapasse o limite lateral (largura) que você definiu. Dicas: Aproveite a função escrita no exercício anterior para adaptá-la a esse novo objetivo. Tente iterar palavra a palavra para isso pode separar o texto em palavras e a cada palavra adicionada aquela linha verifica se ultrapassa ou não os limites de largura se ultrapassar adiciona uma quebra de linhas (\n) antes dessa palavra; O retorno da função deve ser uma **string** com as quebras de linha nos lugares adequados a sua necessidade; tente imprimir o retorno da função no console para saber se a função está adicionando as quebras de linha conforme esperado.

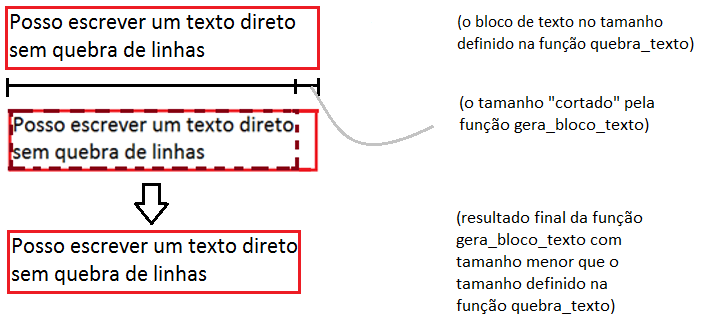


|  |
| --- |
| # SUA FUNCAO  def quebra\_texto(texto, largura, tamanho = 20, tipo\_fonte = 'monospace'):  """  texto (string): texto a ser renderizado em bloco de texto  largura (inteiro): define a largura (fixa) para o bloco de texto  tamanho (inteiro): tamanho da fonte a ser carregada (default 20)  tipo\_fonte (string): tipo de fonte (nome) a ser renderizado  """  # carregue a fonte  # separe o texto por linhas (pense como separar a string) iterando sobre as palavras da string texto e testando se, considerando o tamanho da palavra, se vai ultrapassar os limites de largura definido, se ultrapassar adiciona uma quebra de linhas antes dessa palavra  return texto # retorna a string texto separada  #--------------------------------------------------  print gera\_bloco\_texto2("Posso escrever um texto direto sem quebra de linhas e a minha funcao faz a quebra de linhas para mim, sem esquentar com isso.", 300) |

<Colocar aqui o resultado que a função deve gerar um print do console do python>

Pense nesta função como quando você copia (Ctrl+c) um texto da internet, por exemplo, e cola (Ctrl+v) para o seu editor de textos (Microsoft Word, por exemplo). Toda vez que faz isso basicamente o sistema operacional copia um texto sem quebra de linhas, armazena em uma variável e ao colar o sistema operacional passa essa variável (string) para o editor que o insere em uma função que vai pegar essa string sem quebra de linhas e quebrar as linhas para caber dentro dos limites da página do editor. Lembre-se que a quantidade de linhas dependerá da demanda, ou seja, o programa criará quantas linhas forem necessárias para exibir o texto.

1. Utilize a função para escrita no exercício anterior para gerar a string que será passada para a função escrita no exercício 3 e veja se o resultado é o esperado. Possivelmente a superfície gerada pelo pela função do exercício 3 pode ter uma largura menor que a largura passada como parâmetro da função do exercício anterior, não se preocupe com isso porque a largura passada para a função do exercício anterior é a largura **máxima** que o bloco de texto pode ter portanto não trará problemas para o resultado final.



Fique a vontade se quiser adaptar seu código para que a surface tenha exatamente a largura passada como parâmetro da função do exercício 4.

1. Carregue um texto de um arquivo .txt (bloco de notas) e use-o como entrada da função escrita no exercício 5 (anterior) e repita o processo do exercício 5 para imprimir esse texto.