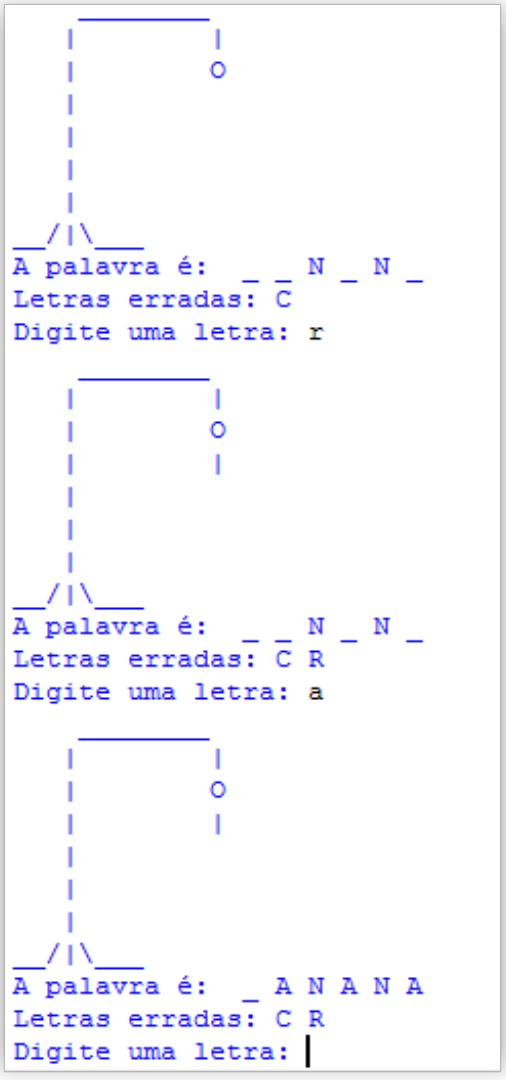
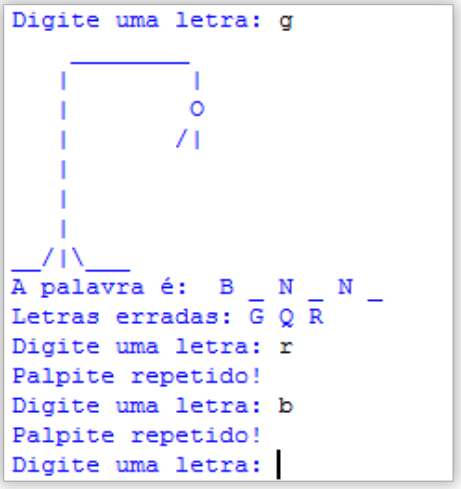
# Jogo da Forca

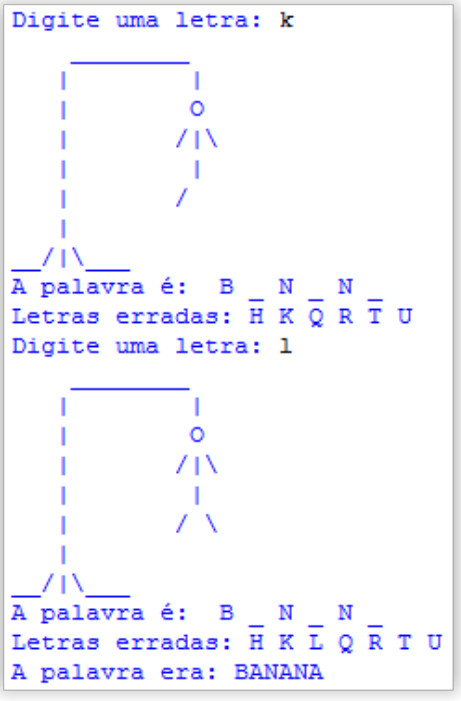
O jogo da forca deve:

- Receber a palavra a ser adivinhada.
- Imprimir a forca vazia.
- Receber cada palpite de letra e imprimir a forca atualizada com as letras acertadas, guardando as letras erradas em uma lista e imprimido-as antes de receber o próximo palpite.
- Informar ao jogador quando ele digitar uma letra repetida (seja ela certa ou errada)
- Finalizar o jogo quando o jogador acertar a palavra ou quando o bonequinho for enforcado. Se este for o caso, o jogo deve dizer qual era a palavra correta.
- O jogador terá 7 tentativas: Cabeça, braço, corpo (peito), outro braço, corpo (barriga), perna, outra perna.

Exemplos:



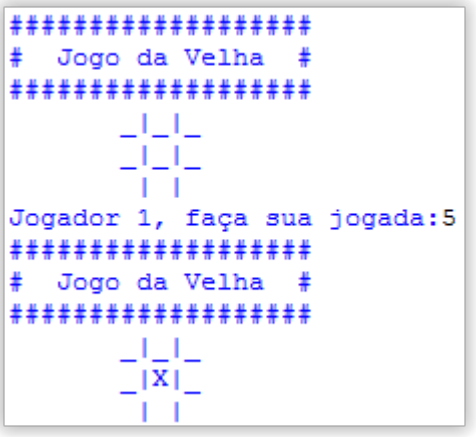



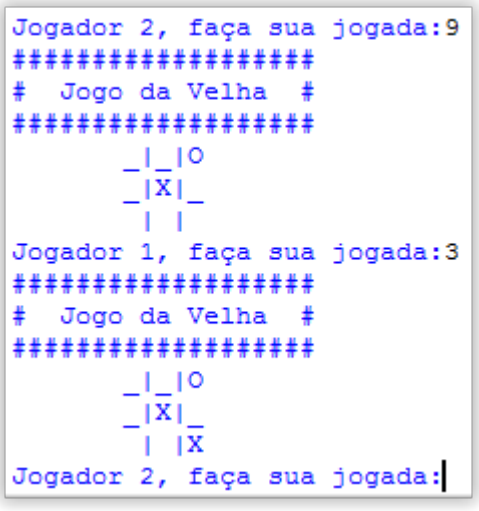
# Jogo da Velha

O jogo da velha deve:

- Imprimir o tabuleiro vazio;
- Receber alternadamente as jogadas do jogador 1 e 2, sendo que o jogador 1 será sempre o ‘X’ e o jogador 2 sempre o ‘O’;
- As jogadas devem ser feitas com os números de 1 a 9, na mesma ordem do teclado numérico, isto é, o 1 é a casa inferior esquerda, o 5 a do centro, o 9 a superior direita, etc. Isso aumenta a jogabilidade pois é mais intuitivo.
- Imprimir o tabuleiro com a jogada feita, caso ela seja válida, e receber a jogada.
- Quando uma jogada for inválida, o jogo deve pedir a jogada desse mesmo jogador novamente. Uma jogada pode ser inválida por dois motivos:
    - O jogador tenta jogar numa casa que já está ocupada, ou
    - O jogador joga algo que seja diferente dos números de 1 a 9.
- O jogo termina quando um dos jogadores alcançar uma sequência de três símbolos iguais:
    - em qualquer horizontal;
    - em qualquer vertical;
    - ou em uma das diagonais principais.
- Ou quando o tabuleiro estiver completo, o que caracteriza empate. Se o jogo terminar em:
    - Vitória -> imprimir uma mensagem parabenizando o jogador vencedor;
    - Empate -> imprimir “Deu velha!”

Exemplos:



# Dicas

Lembrem-se das funções vistas em aula para trabalhar com sequências:

- contar as ocorrências de um dado elemento;
- verificar se contém um dado elemento;
- recuperar um ou mais elementos;
- substituir um elemento por outro (não funciona em strings);
- concatenar;
- adicionar elementos a uma lista (append e extend não funcionam em strings);
- etc.

Métodos em strings:

- Unir uma lista de caracteres em uma string --> `str.join(lst)`:

```
>>> separador = ‘ ’                    # string com um espaço
>>> lista = [‘a’, ‘b’, ‘c’]
>>> s = separador.join(lista)
>>> s
‘a b c’
```

- Transformar todas letras em maiúsculas --> `str.upper()`:

```
>>> s = ‘Banana’
>>> s = s.upper()
>>> s
‘BANANA’
```

Métodos em listas:

Não é necessário colocar a lista de palavras erradas em ordem alfabética, mas quem quiser, basta usar o lista.sort():

```
>>> lista = [‘c’, ‘a’, ‘r’, ‘m’]
>>> lista.sort()
>>> lista
[‘a’, ‘c’, ‘m’, ‘r’]
```

# Criação dos arquivos dos jogos

O jogo deve obrigatoriamente ser dividido em três arquivos, que devem ser nomeados da seguinte forma:

- jogo.py
- velha.py
- forca.py

Os aquivos dos jogos (forca.py e velha.py) devem conter apenas funções e devem rodar o jogo uma única vez, para isso, o código que seria a parte principal do jogo deve ser colocado dentro de uma função obrigatoriamente chamada `main`, que não recebe nenhum parâmetro e não tem nenhum retorno. Ou seja, vocês só precisam pegar o código principal que vocês já tem e colocar dentro de uma função chamada main, como no exemplo a seguir:

Código SEM o main()

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

def subtrai(a,b):
    return a-b

def print_menu():
    print('Digite + para soma')
    print('ou - para subtração')

print_menu()
operacao = input('Digite a operação: ')
n1 = float(input('Digite n1:'))
n2 = float(input('Digite n2:'))
if operacao == '+':
    res = soma(n1,n2)
elif operacao == '-':
    res = subtrai(n1,n2)
else:
    res = 'operação inválida'
print(res)
```

Código COM o main()

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

def subtrai(a,b):
    return a-b

def print_menu():
    print('Digite + para soma')
    print('ou - para subtração')

def main():
    print_menu()
    operacao = input('Digite a operação: ')
    n1 = float(input('Digite n1:'))
    n2 = float(input('Digite n2:'))
    if operacao == '+':
        res = soma(n1,n2)
    elif operacao == '-':
        res = subtrai(n1,n2)
    else:
        res = 'operação inválida'
    print(res)
```

Podemos então importar as funções desse arquivo para um outro e basta chamar a função main() neste outro arquivo para rodar o código. Caso o nome desse arquivo seja `teste.py`, fazemos:

```
>>> import teste
>>> teste.main()
```

E isso irá rodar o código do exemplo corretamente. Para que o código funcione também quando rodarmos ele sozinho, direto do editor, precisamos adicionar uma chamada `main()` ao final dele.

Mas isso faria ele rodar também quando for importado. Portanto adicionamos a estrutura de seleção que é apresentada abaixo **AO FINAL DE TODOS OS ARQUIVOS**.

O arquivo jogo.py terá apenas uma função que será o menu que pergunta qual jogo o usuário quer jogar e chama o arquivo correspondente. Os comandos de importação no arquivo jogo.py ficarão da seguinte maneira (os arquivos forca.py e velha.py precisam necessariamente estar na mesma pasta que jogo.py):

código do arquivo jogo.py

```
########################################
import forca, velha
def main():
     <bloco de código para perguntar em um laço qual o jogo, com menu, opções
     para escolher o jogo, sair, etc.>
     <se jogo da velha, chamar --> velha.main()>
     <se jogo da forca, chamar --> forca.main()>

if __name__ == '__main__':
     main()
########################################
```

Para importar arquivos direto da shell do Python, deve-se adicionar o diretório no qual os arquivos estão salvos à lista de diretórios em que o Python busca os módulos na hora de importá-los.
Esta lista pode ser acessada em `sys.path`, da seguinte maneira:

```
>>> import sys
>>> sys.path
```

E para adicionar o caminho do diretório que contém os seus arquivos, faça:

```
>>> sys.path.append(<caminho>)
```

Onde `<caminho>` é o endereço completo do diretório, lembrando que devemos ou adicionar um `r` antes da string ou uma `\` antes de cada `\` do nome do diretório. Exemplos:

```
>>> sys.path.append(r'C:\Users\rafael.ribeiro\Documents')
```

**OU**

```
>>> sys.path.append('C:\\Users\\rafael.ribeiro\\Documents')
```

Então, basta fazer:

```
>>> import jogo, velha, forca
```

Notem que não é necessário adicionar a extensão `.py`

# Importando arquivos no Python

Vimos em aula que para importar módulos em Python usamos o comando `import.` Imagine que tenhamos criado um arquivo chamado teste.py e dentro desse arquivo definimos 4 funções f1(), f2(), f3() e f4(). Agora queremos importar essas funções para outro arquivo (ou para a shell do Python), podemos usar:

```
1. import teste
2. import teste as t
3. from teste import f1
4. from teste import f1, f2, f3
5. from teste import *
6. form teste import f1 as g1
```

```
1) import teste
```

Importa o arquivo teste e todas as funções que ele contém, criando um objeto chamado teste. Para usarmos cada função, devemos acessá-las a partir do objeto teste fazendo `teste.f1(),` `teste.f2()`, etc.

```
2) import teste as t
```

Faz o mesmo que 1), mas agora o objeto criado tem o nome `t,` e para acessar as funções fazemos: `t.f1(), t.f2(),` etc.

```
3) from teste import f1
```

Importa a partir do arquivo `teste.py` apenas a função `f1,` e podemos acessá-la diretamente fazendo `f1().` As demais funções não são importadas e não estão disponíveis nessa situação.

```
4) from teste import f1, f2, f3
```

Faz o mesmo que 3), só que importa as funções `f1, f2` e `f3`. A função f4 não é importada e não está disponível.

```
5) from teste import *
```

Faz o mesmo que 3) e 4), mas importa todas as funções que tiverem sido definidas no arquivo teste.py, e cada uma pode ser acessada diretamente, fazendo `f1(), f2(),` etc.

```
6) from teste import f1 as g1
```

Faz o mesmo que 3) e 2) combinados. Importa apenas a função `f1` e salva ela como `g1`. A única função acessível dessa forma é a função `g1`, que é uma cópia da função `f1`, e pode ser chamada fazendo `g1()`.