Strings

fmasanori@gmail.com

Aspas de vários tipos

- Posso usar aspas simples, duplas ou triplas

```
>>> x = 'abacate'
>>> x
'abacate'
>>> y = "MacDonald's"
>>> y
"MacDonald's"
>>> form = '''
<html>
        <head>
            <title> Teste </title>
        </head>
        <body>
            <p>Testando</p>
        </body>
</html>'''
```

Fatiamento

- Fatia do primeiro índice até o anterior do segundo

```
>>> x = "0123456789"
>>> print(x[0:2])
01
>>> print(x[1:2])
1
>>> print(x[2:4])
23
>>> print(x[0:5])
01234
>>> print(x[1:8])
1234567
```

Fatiamento

- Podemos omitir índices, substituindo pelo extremo correspondente e também podemos ter índices negativos: -1 último, -2 penúltimo

```
>>> print(x[:2])
01
>>> print(x[4:])
456789
>>> print(x[4:-1])
45678
>>> print(x[-4:-1])
678
>>> print(x[:])
0123456789
```

Incremento no fatiamento

- Posso usar um incremento ao fatiar a string

```
>>> texto = 'batatinha quando nasce'
>>> texto[::2]
'bttnaqad ac'
>>> texto[::-1]
'ecsan odnauq ahnitatab'
```

Incremento no fatiamento

- Verifique se uma palavra é palíndrome

```
palavra = input('Palavra: ')
palíndrome = palavra == palavra[::-1]
print (f'{palavra} é palíndrome?')
print (palíndrome)
```

Strings são imutáveis

```
>>> texto = 'Alô Mundo'
>>> texto[0] = '@'
Traceback (most recent call last):
  File "<pyshell#1>", line 1, in <module>
    texto[0] = '@'
TypeError: 'str' object does not support
item assignment
```

Posso criar novas strings

- Usando concatenação resolvemos esse problema

```
>>> texto = '@' + texto[1:]
>>> print (texto)
@lô Mundo
```

Concatenação

- Faça um programa que leia uma palavra e troque as vogais por “*”

```
palavra = input('Palavra: ')
k = 0
troca = ''
while k < len(palavra):
   if palavra[k] in 'aeiou':
      troca = troca + '*'
   else:
      troca = troca + palavra[k]
   k = k + 1
print (troca)
```

Verificação parcial de strings

```
>>> arquivo = 'prog.py'
>>> arquivo.startswith ('p')
True
>>> arquivo.endswith ('py')
True
>>> resposta = "Sim"
>>> resposta.lower()
'sim'
>>> resposta.upper()
'SIM'
>>> resposta.lower() in 'sim não yes no'
True
```

find and replace

```
>>> s = 'um tigre, dois tigres, três tigres'
>>> s.find('tigre')
3
>>> s.find('tigre', 4)
15
>>> s.find('tigre', 16)
28
>>> s.replace('tigre', 'gato')
'um gato, dois gatos, três gatos'
>>> s
'um tigre, dois tigres, três tigres'
>>> s = s.replace('tigre', 'gato')
>>> s
'um gato, dois gatos, três gatos'
```

split and join

```
>>> txt = 'batatinha quando nasce'
>>> txt.split()
['batatinha', 'quando', 'nasce']
>>> data = '21/02/2011'
>>> data.split('/')
['21', '02', '2011']
>>> ip = '198.188.10.144'
>>> ip.split('.')
['198', '188', '10', '144']

>>> times = ['Palmeiras', 'Santos', 'Corintians']
>>> '/'.join(times)
'Palmeiras/Santos/Corintians'
```

Exercício

- Faça um programa que solicite a data de nascimento (dd/mm/aaaa) e imprima com o nome do mês por extenso

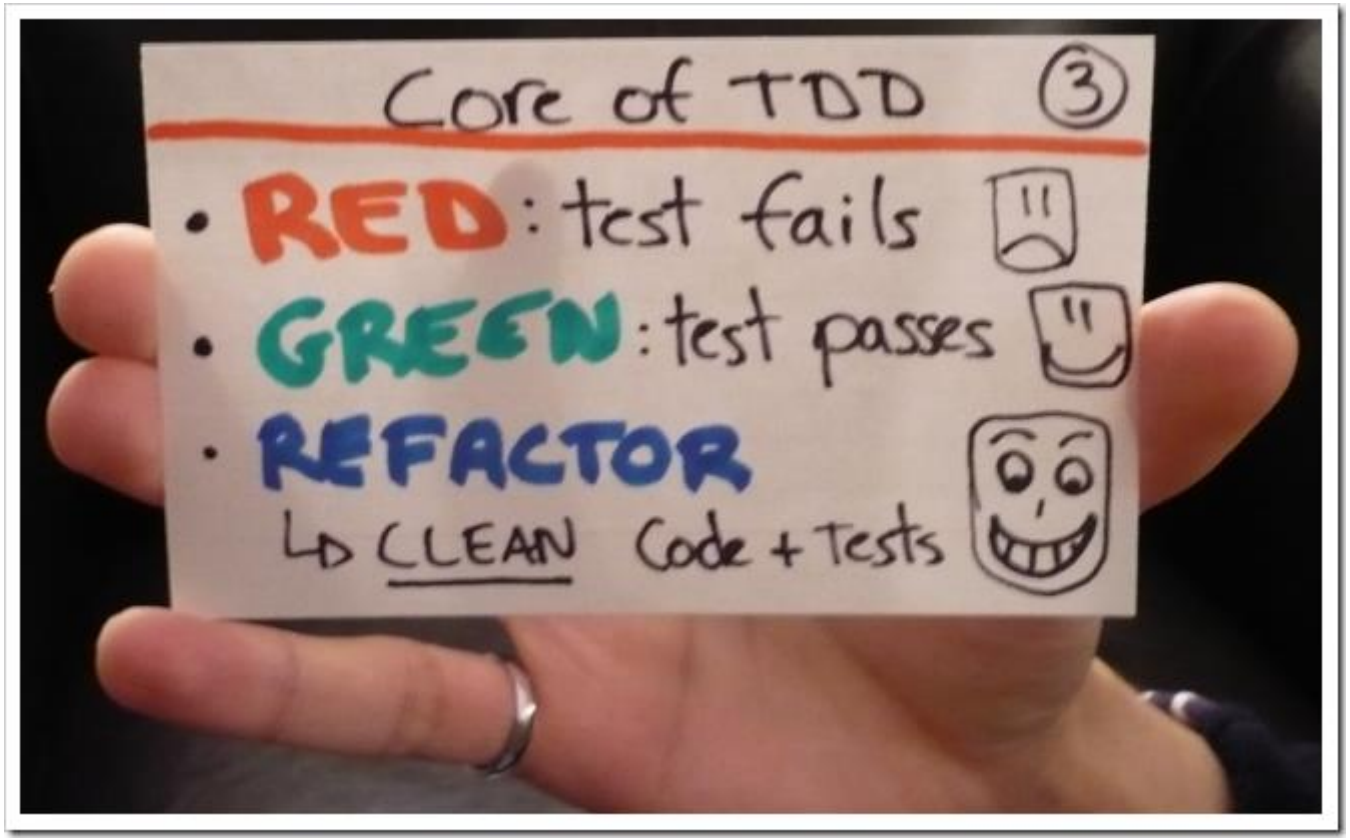
```
mes = '''janeiro fevereiro março
    abril maio junho julho agosto
    setembro outubro novembro
    dezembro'''.split()

d,m,a=input('dd/mm/aaaa: ').split('/')

print(f'{d} de {mes[int(m)-1]} de {a}')
```

Coding Dojo

Coding Dojo

- Desenvolvimento guiado por testes
- Passos de bebê
- Pair programming

Desenvolvimento guiado por testes



Baby Steps

Pair Programming