Lógica de Programação Orientada a Objetos

Bem-vindos!

Objetivos

- Argumentos
- Algumas bibliotecas

ARGUMENTOS POSICIONAIS

- arg1, arg2 e arg3 são argumentos posicionais
- Quando invoca func, exatamente 3 argumentos devem ser passados, então, o número errado de argumentos resulta em um TypeError
- A ordem da chamada determina que arg eles estão vinculados

```
def func(arg1, arg2, arg3):
    pass

func(a, b, c)
```

ARGUMENTOS NOMEADOS

- Argumentos nomeados podem ser passados fora de ordem

```
def say(arg1, named1, named2):
    print(arg1, named1, named2)

say('make', named2='day', named1='my')

# output
make my day
```

MUTAÇÃO DOS ARGUMENTOS PADRÃO

- Os argumentos padrão são avaliados quando a função é definida
- Em todas as chamadas, o objeto que a expressão foi avaliada será usado.
- Se o padrão é mutável, as atualizações seguem um efeito de chamadas
- def func(a=[]) deverá mudar o padrão em cada chamada
- Use None como padrão para evitar mutação

```
def func(a=None):
    a = a or []
```

LET'S CODE

*ARGS

- Uma número de variáveis de argumentos posicionais que podem ser especificadas
- Deve usar qualquer identificador, mas args é convencional
- args é uma tupla de 0 ou mais objetos

```
def func(arg1, *args):
    print(args)
func(1, 2, 3, 4)

# output:
(2, 3, 4)
```

**KWARGS

- Use **kwargs no final
- Deve usar qualquer identificador, mas kwargs é convencional
- kwargs é um dicionário de strings para valores
- As chaves do kwargs são os nomes dos argumentos

```
def foo(arg1, **kwargs):
    print(kwargs)

foo(1,two=2, three=3)

# output
{'two': 2, 'three': 3}
```

LET'S CODE

* EM CHAMADAS DE FUNÇÃO

- l é um iterável
- Isso é pego descompactado, como argumentos posicionais de bar

```
def bar(arg1, arg2, arg3):
    print(arg1+arg2+arg3)

l = [1, 2, 3]
bar(*l)

# output
6
```

** EM CHAMADAS DE FUNÇÃO

- person deve ser um dicionário de forma {'string': val, ...}
- Ele é pego descompactado como argumentos palavra-chave de print_version

```
def print_person(name, age):
    print('{} is {} years old'.format(name, age))

person = {'name': 'Mike', 'age': 28}
print_person(**person)

# output
Mike is 28 years old
```

TIME

- time.time retorna o tempo em segundos desde a época como um numéro de ponto flutuante.

```
import time
t0 = time.time()
for i in range(10000000):
    pass
t1 = time.time()
print(t1 - t0)

# saida:
1.1572110652923584
```

PROPAGANDO EXCEÇÕES

- time.sleep suspende execução para determinado número de segundos.

```
import time
print('Processando, por favor aguarde ...')
time.sleep(2)
print('Pronto.')

# Saida:
Processando, por favor aguarde ...
Pronto.
```

LOGGING

- Em programas grandes e de longa execução, nós precisamos de impressões mais sofisticadas.
- O pacote logging habilita para facilmente efetuar log do estado corrente e o tempo exato no seu programa

```
logging.debug('Alltems operational')
logging.info('Airspeed knots')
logging.warn('Lowfuel')
logging.error('Nol. Trying to glide.')
logging.critical('Glide attempt failed. About to crash.')

# Saida:
WARNING:root:Lowfuel
ERROR:root:Nol. Trying to glide.
CRITICAL:root:Glide attempt failed. About to crash.
```

OS

- os.listdir retorna uma lista contendo os nomes das entradas em um diretório dado um caminho:

```
import os

for filename in os.listdir('.'):
    print(filename)

# saida
The-standard-library.ipynb
example.log
unit-tests.ipynb
```

- os.path.join concatena caminhos (de acordo com o OS):

```
import os

home = '/home/user'
os.path.join(home, 'Downloads')

# output
'/home/user/Downloads'
```

- os.path.splitext separa o arquivo dentro da raiz, da extensão:

```
os.path.splitext('/home/noam/Downloads/xom.csv')

# saida
('/home/noam/Downloads/xom', '.csv')
```

- os.path.getsize retorna o tamanho do arquivo em bytes
- os.path.isdir verifica se o caminho passado é um diretório

```
os.path.getsize('The-standard-library.ipynb')

# saida
8440
```

```
os.path.isdir('The-standard-library.ipynb')

# saida
False
```

SYS

- sys.argv[0] contêm o nome do arquivo
- sys.argv[1:] contêm argumentos (se tiver)

Arquivo test.py:

```python
import sys

def main(argv):
    a = int(argv[1])
    b = int(argv[2])
    return a + b

if __name__ == '__main__':
    print(main(sys.argv))
```

Linha de comando:

```
$ python test.py 5 10
15
```

LET'S CODE

EXERCÍCIO

- https://www.hackerrank.com/challenges/class-2-find-the-torsional-angle/problem
- https://www.hackerrank.com/challenges/python-time-delta/problem

LET'S CODE