
FIAP
MBA

Introdução ao Python

Dheny R. Fernandes

Agenda

1. **Funções**
2. **Pacotes**
3. **Arquivos e I/O**
4. **Tempo**
5. **Regex**

Funções

Funções são blocos de código identificados por um nome, que podem receber parâmetros pré-determinados.

Características:

- Podem ou não retornar objeto
- Aceitam *Doc Strings*
- Tem *namespace* próprio
- Aceitam parâmetros opcionais

Doc Strings são documentações de uma estrutura

```
def func(parametro1, parametro2=padrao):
    """Doc String
    """
    <bloco de código>
    return valor
```

```python
#implementação recursiva
def fatorial(num):
    """
    Esta função calcula o fatorial de um número
    usando implementação recursiva

    args: num (int) - número a partir do qual
    será calculado o fatorial

    return: o fatorial de num
    """
    if (num <= 1):
        return 1
    else:
        return(num * fatorial(num - 1))
```

Quanto mais pythonica a implementação de uma função, melhor sua performance:

```python
def numero_par(num):
    resto = num % 2
    if (resto == 0):
        return True
    else:
        return False
```

```python
def numero_par_otimizada(num):
    if((num % 2) == 0):
        return True
    else:
        return False
```

```python
def numero_par_mais_otimizada(num):
    return True if ((num % 2) ==0) else False
```

Resolver o exercício no notebook

Pacotes

Para o Python, pacotes são arquivos que podem ser importados para um programa.

Eles são compilados quando importados pela primeira vez e armazenados em arquivo, possuem namespace próprio e aceitam Doc Strings.

São objetos Singleton (é carregada somente uma instância em memória, que fica disponível de forma global para o programa)

Podemos importar pacotes que, geralmente, estão disponíveis no ambiente Anaconda, assim como também criar nossos próprios pacotes .py e importá-los em nossos notebooks.

Quando um pacote não está disponível, podemos instalá-lo via pip ou conda.

Para importar pacotes, fazemos:

1. import pandas
2. from collections import Counter
3. from collections import * (não recomendado)

Criar um arquivo, nomeá-lo como calc.py e escrever a seguinte função:

```
1  def media(lista):
2      return float(sum(lista)) / len(lista)
```

Importar esse arquivo no jupyter, criar uma lista de inteiros e usar a função media para retornar a média dos valores presentes na lista:

```
import calc
l = [23, 54, 31, 77, 12, 34]
print (calc.media(l))
```

Arquivos e I/O

Os arquivos no Python são representados por objetos do tipo *file*, que oferecem métodos para diversas operações de arquivos

Arquivos podem ser abertos para leitura ('r', que é o default), gravação ('w') ou adição ('a'), em modo texto ou binário('b')

Em Python:

- sys.stdin representa a entrada padrão.
- sys.stdout representa a saída padrão.
- sys.stderr representa a saída de erro padrão.

Exemplo de manipulação de arquivo:

```
import sys
# Criando um objeto do tipo file
temp = open('temp.txt', 'w')
# Escrevendo no arquivo
for i in range(100):
    temp.write('%03d\n' % i)
# Fechando
temp.close()
temp = open('temp.txt')
# Escrevendo no terminal
for x in temp:
    # Escrever em sys.stdout envia o texto para a saída padrão
    sys.stdout.write(x)
temp.close()
```

Exemplo de leitura:

```
import sys
import os.path
# input() retorna a string digitada
fn = input('Nome do arquivo: ').strip() #remove espaço em branco extra do nome do arquivo
if not os.path.exists(fn):
    print ('Tente outra vez...')
    sys.exit()
# Numerando as linhas
for i, s in enumerate(open(fn)):
    print (i + 1, s,)
```

É possível imprimir uma lista contendo as linhas de um arquivo:

```
# Imprime uma lista contendo linhas do arquivo
print (open('temp.txt').readlines())
```

Tempo

O python possui dois módulos para lidar com tempo:

time: implementa funções que permitem utilizar o tempo gerado pelo sistema.

datetime: implementa tipos de alto nível para realizar operações de data e hora.

Tempo


```
import time
print (time.localtime()) #imprime como uma estrutura
print (time.asctime()) #imprime como string
for x in range(5):
    time.sleep(3)
    print('Espera 3 segundos e imprime na tela\n')
```

O pacote time também pode ser utilizado para calcular o tempo decorrido ao se executar um loop.

```
start_time = time.time()
x = 0
for i in range(0,1000000):
    x += i
end_time = time.time()

print(x)
print('tempo decorrido: ',end_time-start_time, 'segundos')
```

Em *datetime*, estão definidos quatro tipos para representar o tempo:

- *datetime*: data e hora.
- *date*: apenas data.
- *time*: apenas hora.
- *timedelta*: diferença entre tempos.

Tempo


```python
import datetime
from datetime import timedelta

dt = datetime.datetime(2020, 12, 31, 23, 59, 59) #criando uma data com horário

data = dt.date() #retorna a data
hora = dt.time() #retorna o horário

dd = dt - dt.today() #cálculo de tempo
print ('Data:', data)
print ('Hora:', hora)
print ('Quanto tempo falta para 31/12/2020:', dd)
print ('1 ano a partir de agora será: '+ str(datetime.datetime.now() + timedelta(days=365)))
```

Exemplo:

	hora	quantidade
0	February 1, 2019, Hour 0	415
1	February 1, 2019, Hour 1	231
2	February 1, 2019, Hour 2	118
3	February 1, 2019, Hour 3	64
4	February 1, 2019, Hour 4	107

```
hora['hora'] = [hora['hora'][x].replace('Hour',' ') for x in range(len(hora))]
hora['hora'] = [re.sub(' +', ' ', hora['hora'][x]) for x in range(len(hora))]
hora['hora'] = [re.sub(',','',hora['hora'][x]) for x in range(len(hora))]
hora['hora'] = [datetime.datetime.strptime(hora['hora'][x], '%B %d %Y %H').strftime('%m/%d/%Y %H:%M:%S') for x in range
```

```
hora.tail()
```

	hora	quantidade
667	02/28/2019 19:00:00	1114
668	02/28/2019 20:00:00	1102
669	02/28/2019 21:00:00	1111
670	02/28/2019 22:00:00	937
671	02/28/2019 23:00:00	660

Regex

Expressão regular é uma maneira de identificar padrões em sequências de caracteres. No Python, o módulo re provê um analisador sintático que permite o uso de tais expressões. Os padrões definidos através de caracteres que tem significado especial para o analisador.

Principais caracteres:

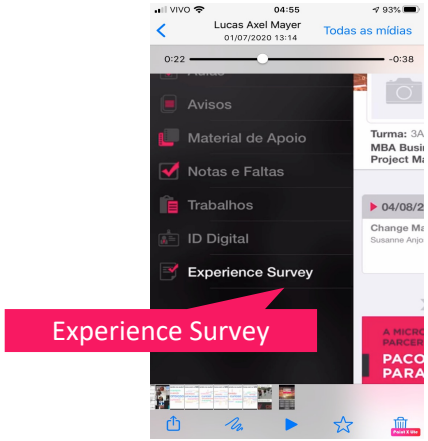
- Ponto (.): Em modo padrão, significa qualquer caractere, menos o de nova linha.
- Circunflexo (^): Em modo padrão, significa inicio da *string*.
- Cifrão ($): Em modo padrão, significa fim da *string*.
- Contra-barra (\): Caractere de escape, permite usar caracteres especiais como se fossem comuns.
- Colchetes ([]): Qualquer caractere dos listados entre os colchetes.
- Asterisco (*): Zero ou mais ocorrências da expressão anterior.
- Mais (+): Uma ou mais ocorrências da expressão anterior.
- Interrogação (?): Zero ou uma ocorrência da expressão anterior.
- Chaves ({n}): n ocorrências da expressão anterior.
- Barra vertical (|): "ou" lógico.
- Parenteses (()): Delimitam um grupo de expressões.
- \d: Dígito. Equivale a [0-9].
- \D: Não dígito. Equivale a [^0-9].
- \s: Qualquer caractere de espaçamento ([\t\n\r\f\v]).
- \S: Qualquer caractere que não seja de espaçamento.([^ \t\n\r\f\v]).
- \w: Caractere alfanumérico ou sublinhado ([a-zA-Z0-9_]).
- \W: Caractere que não seja alfanumérico ou sublinhado ([^a-zA-Z0-9_]).

```
import re
rex = re.compile('\w+') #qualquer caracter alfanumérico - compilado
bandas = 'Queen, Aerosmith & Beatles'
print (bandas, '->', rex.findall(bandas))
phone = "2004-959-559 # This is Phone Number"
num = re.sub('#.*$', "", phone) #elimina tudo após #
print ("Phone Num : ", num)
num = re.sub(r'\D', "", phone)# só deixa número
print ("Phone Num : ", num)
```

Resolver o exercício no notebook

O que você achou da aula de hoje?

Entrar no aplicativo FIAPP, e no menu clicar em Experience Survey



Ou pelo link: https://fiap.me/Pesquisa-MBA

Obrigado!

profdheny.fernandes@fiap.com.br

/dhenyfernandes


FIAP