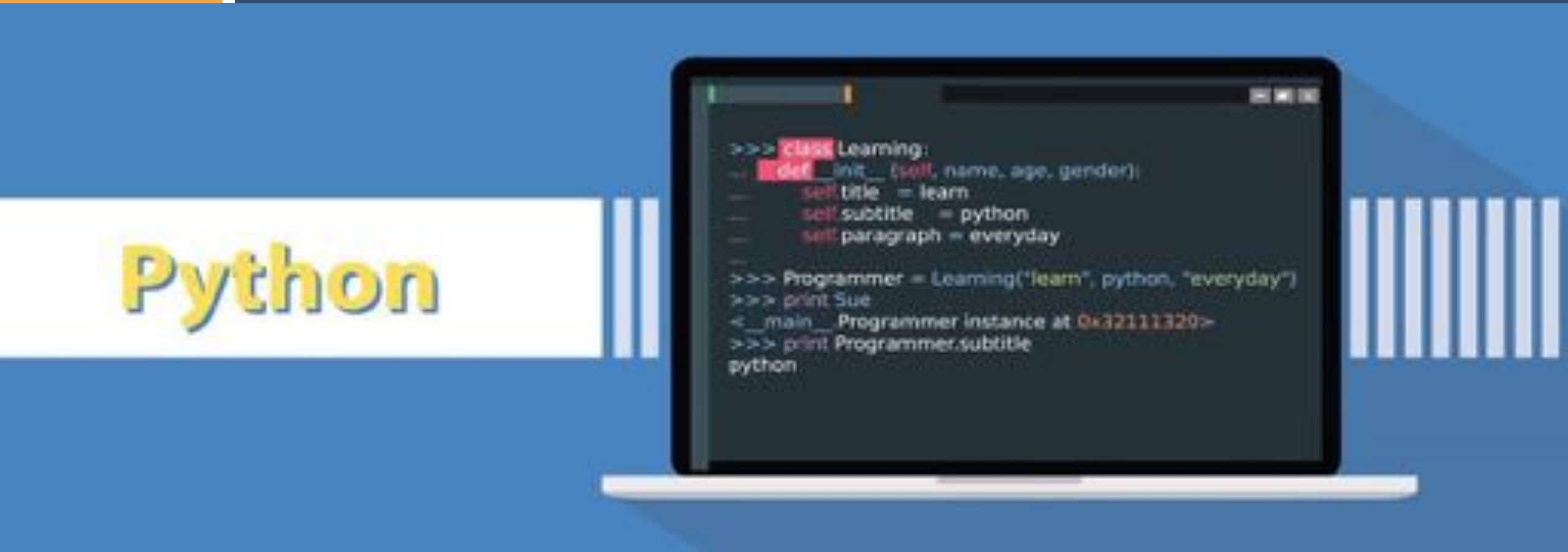
UNIVERSIDADE CATÓLICA DE PETRÓPOLIS
CENTRO DE ENGENHARIA E COMPUTAÇÃO
TECNÓLOGO EM REDES DE COMPUTADORES

[PCD] PRINCÍPIO DE COMUNICAÇÃO DE DADOS
|PARTE 2 | AULA 03 | Python | Introdução|

VERSÃO: 1.0 - Março de 2020

Professor: Luís Rodrigo
E-mail: luis.goncalves@ucp.br
Site: http://lrodrigo.sgs.lncc.br

[3] SOCKETS | Python



[3.1]

SOCKETS | Python

Conceitos Básicos

Dinamicamente tipado

Multi-paradigma

Sintaxe intuitiva

Interpretado

Tipos de dados de alto nível

Características da Linguagem:

- Por padrão ela é Interpretada
- Implementação mais comum → CPython
  - Combina Interpretação e Compilação
  - Primeiro o código fonte é compilado para byte code
  - Depois o byte code é interpretado
- O Código fonte podem ser diretamente executado

Usado pelas empresas:

- Google, Facebook(Instagram),
- Microsoft, Dropbox, Globo.com, etc.

Áreas:

- web, data science, devops,
- automação, IA e muito mais.

Guido Van-Rossum

- **The Story of Python, by Its Creator, Guido van Rossum** [https://www.youtube.com/watch?v=J0Aq44Pze-w]

"Foi lançada por Guido van Rossum em 1991. Atualmente possui um modelo de desenvolvimento comunitário, aberto e gerenciado pela organização sem fins lucrativos Python Software Foundation."
Wikipedia

PEPs - Python Enhancement Proposals

- **[https://www.python.org/dev/peps/]**

As PEPs são documentos que formalizam as funcionalidade, propósitos, procedimentos e ambientes. Eles funcionam como um guia para orientar no uso da linguagem.

The Zen of Python

- **[https://www.python.org/dev/peps/pep-0020/]**

Bonito é melhor que feio
Explícito é melhor do que implícito
Simples é melhor do que complexo
Complexo é melhor do que complicado
Legibilidade conta

Complexidade do Código – Python x Java

```
public class Hello{
        public static void main(String[] args){
                System.out.println("Hello, world!");
        }
}
```

Complexidade do Código – Python x C++

```
#include <iostream>
int main(){
	std::cout << "Hello World!" << std::endl;
	return 0;
}
```

Complexidade do Código – Python

```
print('Hello World!’)
```

Python v2 e v3 - https://www.python.org/doc/versions/

- **12/1989**: Guido Van Rossum inicia a implementação do Python
- **01/1994**: Versão 1.0 lançado
- **10/2000**: Versão 2.0 lançado
- **12/2008**: Versão 3.0 lançado
- **06/2009**: Versão 3.1 lançado
- **07/2010**: Versão 2.7 lançado com correções de segurança
- 02/2020: Versões atual do Python são 2.7.17 and 3.8.1

Python v3

- Tentativa de melhorar algumas partes do “design” da linguagem, principalmente a parte de encoding de strings
- Financiado pelo Google.
- print e exec passam a ser funções
- Toda string passou a ser Unicode
- Melhor suporta a programação assíncrona (async/await)
- O virtualenv passou a fazer parte da “Standard Library”

[3.2]

SOCKETS | Python

Executando e Testando o Código

REPL

iPython

Arquivos .py

REPL - Read Evaluate Print Loop

```
$ python
Python 2.7.17 (default, Nov  7 2019, 10:07:09)
[GCC 7.4.0] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>>print('Ola Mundo !')
Ola Mundo !
```

- Para sair use:
  - ctrl + d on *nix
  - ctrl + z on windows.

REPL - Read Evaluate Print Loop

- Pode ser utilizado para testar códigos simples

```
>>> 10 + 10
20
>>> 50 * 2
100
>>> 10 + 20 * 3
70
>>> (10 + 20) * 3
90
```

iPython

- Semelhante ao REPL, mas com alguns melhoramentos
- Use o tab para autocomplete.
- Acrescente um ? no fim da variável, função, classe para obter mais informações.
- Executa comandos do shell: !ls

iPython

- Teste os comandos abaixo e veja o resultado
  - %magic
  - %history
  - %save
  - %pastebin

iPython – Ubutnu 18.04

```
$ sudo apt update
$ sudo apt install ipython ipython3
```

Executando e testando o código

iPython

```
Python 2.7.17 (default, Nov  7 2019, 10:07:09)
Type "copyright", "credits" or "license" for more information.

IPython 5.5.0 -- An enhanced Interactive Python.
?         -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help      -> Python's own help system.
object?   -> Details about 'object', use 'object??' for extra details.

In [1]: print ("Ola Mundo !")
Ola Mundo !

In [2]: "Ola Mundo !" * 3
Out[2]: 'Ola Mundo !Ola Mundo !Ola Mundo !'

In [3]:
Do you really want to exit ([y]/n)?
```

Arquivos .py

- São os arquivos com os códigos fonte dos programa
- Eles são interpretados, logo não precisam ser compilados
- No *nix, precisamos alterar sua permissão de forma que possam ser executados

```
$ chmod +x [nome_do_arquivo].py
```

Executando e testando o código

Exemplo: ~/Documents/Aula02/exemplo01.py

```
#!/usr/bin/python
print (‘Ola, mundo !!’);
```

Execute no *Shell* do Linux

```
$ python ~/Documents/Aula02/exemplo01.py
Ola, Mundo !!
```

[3.3]

SOCKETS | Python

PEP8 – Estilo de Programação

O PEP-8 é um manual de boas práticas para escrita de código Python que visa padronizar o de código/ comentários. Tornado o código mais legível. Várias empresas adotam o PEP-8 em suas equipes para facilitar o trabalho em grupo.

- Raissa Camelo

- Forte ênfase na legibilidade
- PEP 8 é um guia de estilo para o código Python.
  - PEP 8 -- Style Guide for Python Code
  - PEP 8 - Guia de Estilo Para Python

- **Indentação:** 4 espaços por nível; alguns códigos antigos ainda estão com 8 espaços;
- **Tabulações e espaços:** nunca misture tabulação com espaço; O uso de espaço é o mais indicado;
- **Comprimento máximo de linhas:** limitar para 80 colunas; Utilizar a barra invertida (\) para gerar uma quebra de linhas
- Não deve-se utilizar múltiplos comandos em uma única linha.

- **Import:** devem ser feitos em linhas separadas; devem ser colocados no topo do arquivo; logo depois dos comentários ou "docstrings"; antes de constantes e variáveis globais; Devem ser agrupados na seguinte ordem:
  - Módulos da Biblioteca padrão
  - Módulos grandes relacionados entre sí (todos os módulos relacionados a secokets)
  - Módulos específicos da aplicação.

- **Não deve-se utilizar espaços:**
  - antes e após parênteses, colchetes ou chaves;
  - logo após uma virgula, ponto-e-vírgula ou dois-pontos;
  - mais de um em volta de algum atribuidor, para alinhar os operandos;
  - ao redor do sinal de igual (=) quando usando para indicar um valor padrão de um argumento

[3.4]

SOCKETS | Python

Variáveis e Tipos de dados

Em Python tudo é objeto.

Até uma “String” tem seus próprio métodos.

Não é preciso especificar o tipo da variável quando ela á declarada.

SOKECTS | Python

Variáveis e tipos de dados

- As variáveis não precisam ser previamente declaradas.
- São criadas e tipadas quando lhes atribuímos algum valor.
- As variáveis são fortemente tipadas, e possuem um conjunto de definido de operações;
- Todas a variáveis são objetos (possuem métodos e atributos)

Variáveis - Atribuições

- A atribuição de valores é realizada via símbolo de “=“

```
>>> var = 15
>>> var
15

>>> var = 15.02
>>> var
15.02
```

- A atribuição aumentada

```
>>> var = 15.02
>>> var
15.02
>>> var = 15.2
>>> var +=10
>>> var
25.2
>>> var -= 5.2
>>> var
20.0
>>> var *= 2
>>> var
40.0
>>> var /= 4
>>> var
10.0
```

- A atribuição aumentada

```
>>> var = 11.2
>>> var
11.2
>>> var //= 2
>>> var
5.0
>>> var **= 2
>>> var
25.0
>>> var %= 2
>>> var
1.0
```

Variáveis - Nomes

- Letras minúsculas;
- Palavras separadas por underline (_)

```
cli_ip.    = “192.0.2.15”
srv_ip     = “192.0.2.30”
srv_porta = 61502
```

Variáveis - Comparações

Operação	Significado
<	Menos que
<=	Menos ou igual que
>	Maior que
>=	Maior ou igual que
==	Igual
!=	diferente
is	objeto é idêntico
is not	o objeto não é idêntico

- Pode assumir – True ou False

```
verdadeiro =  True
falso      =  False
```

- Valores que representam False

```
falso_1       =  None
falso_2       =  0
falso_3          =  []
```

- False pode ser qualquer string vazia

Variáveis – Operações Booleanas

Operação	Significado
x or y	Se x é False; então y; senão x
x and y	Se x é False; então x; senão y
not x	Se x é False; então True; senão False

[3.4.1]

SOCKETS | Python | Variáveis

String

As variáveis do tipo **str** são **imutáveis**

Ou sejam não podem ser modificadas

- Literais de String – Equivalentes

```
"Hello World!"
'Hello World!'
"""Hello World!"""
'''Hello World!'''
```

- Utilizando tipos diferentes de aspas

```
"It's a very nice day."
'The sign says "Hello World!".'
```

Strings multilinha

- Aspas duplas

```
"""Shopping List:
Cheese
Apples
Bread"""
```

- Aspas simples

```
'''ABC
DEF
GHI'''
```

String | Sequencias de Escape

Operação	Significado
\n	Nova linha
\t	Tabulação
\’ e \”	Aspas simples e Aspas duplas
\\	Barra
\x68	Caracter ASCII 104 (68 Hex <-> 104 Dec)

SOKECTS | Python | Variáveis

- Escape habilitado

```
>>> print('c:\windows\newstuff\todo')
c:\windows
ewstuff odo
```

- Escape desabilitado

```
>>> print(r'c:\windows\newstuff\todo')
c:\windows\newstuff\todo
```

SOKECTS | Python | Variáveis

- endswith, startswith

```
'hello world'.startswith('he')  # -> True
```

- isalnum, isalpha, isdigit, islower, isupper, isspace

```
'123'.isdigit()  # -> True
'Hello World'.islower()  # -> False
```

SOKECTS | Python | Variáveis

- count

```
'hello world'.count('l')  # -> 3
```

- find

```
'hello world'.find('l')  # -> 2
'hello world'.find('t')  # -> -1
```

SOKECTS | Python | Variáveis

- lower, upper , title , capitalize , swapcase

```
'hello world'.title()  # -> 'Hello World'
'hello world'.capitalize()  # -> 'Hello world'
```

- replace

```
'hello world'.replace('world', 'john')  # -> 'hello john'
```

SOKECTS | Python | Variáveis

- strip, rstrip, lstrip - remove espaços e nova linha

```
'     hello!    \n'.strip() # -> 'hello!'
```

- + e *

```
'hello ' + 'world'  # -> 'hello world'

'hello ' * 3  # -> 'hello hello hello '
```

SOKECTS | Python | Variáveis

- input(“Mensagem”) → Python 3

```
>>> nome=input("Digite seu nome : ")
Digite seu nome : Seu nome
>>> nome
'Seu nome'
```

- raw_input(”Mensagem") → Python 2

```
>>> nome=raw_input("Digite seu nome : ")
Digite seu nome : Luis Rodrigo
>>> nome
'Luis Rodrigo'
```

SOKECTS | Python | Variáveis | Strings | Formatando

- Variável

```
name = 'Batata'
```

- Estilo Ruim

```
print('Olá, ' + name + '!' )
Olá, Batata!
```

SOKECTS | Python | Variáveis

- Estilo Antigo

```
>>> print('Olá, %s!' % name)
Olá, Batata!
```

- Estilo Novo

```
>>> print('Olá, {}!'.format(name))
Olá, Batata!
```

SOKECTS | Python | Variáveis | Strings | Formatando

- **Variáveis**

```
>>> name = 'Batata'
>>> idade=16
```

- **Estilo Novo**

```
>>> print('Olá, {} você tem {} anos!'.format(name,idade))
Olá, Batata você tem 16 anos!
```

- **Formatação Nomeada**

```
>>> TMPL = 'Você obteve um erro no arquivo {file} na linha {line}'
>>> print(TMPL.format(file='a.py', line=5))
Você obteve um erro no arquivo a.py na linha 5
```

- Formatação Posicionada

```
>>> print('{0}, {0} e {1}'.format('repete','não repete'))
repete, repete e não repete
```

SOKECTS | Python | Variáveis

- Definindo o tamanho (x) de caracteres {:x}

```
>>> TEST_RESULTS_TMPL = '{nome:40} {status:10}'

>>> print(TEST_RESULTS_TMPL.format(nome='NDU', status='Failed'))
NDU                                      Failed

>>> print(TEST_RESULTS_TMPL.format(nome='Cluster expansion', status='Succeed'))
Cluster expansion                        Succeed
```

- Formatando números como binário {:b}

```
>>> print('{:b}'.format(15))
1111
```

- Formatando números como hexadecinal {:x}

```
>>> print('{:x}'.format(15))
f
```

- Formatando números como octal {:o}

```
>>> print('{:o}'.format(15))
17
```

- Para mais informações consulte:
  - Especificação para mini-linguagem de formatação
  - Exemplos
  - Casos mais comuns