Introducción a Python

Taller de Proyecto II

¿Qué es Python?

- Python es un lenguaje de programación interpretado cuya filosofía hace hincapié en una sintaxis que favorezca un código legible
- Multiparadigma: es un lenguaje orientado a objetos, pero también permite usar otros paradigmas de programación tales como programación estructurada, programación orientada a aspectos y, en menor medida, programación funcional.
- Multiplataforma
    - Sistemas operativos Linux/UNIX, MacOS, Windows, otros.

¿Qué es Python?

- Lenguaje interpretado: no se compila, se interpreta a medida que se necesita.
    - Se podría decir que es “SEMI-INTERPRETADO” ya que el código fuente se traduce a un pseudo-código de máquina llamado bytecode la primera vez que se ejecuta generando archivos .pyc o pyo (bytecode optimizado), siendo estos los que se ejecutarán en las próximas instancias.
    - Ventajas: más flexible y más portable que los lenguajes compilados.
- Tipado dinámico: una variable puede tomar valores de distinto tipo en distintos momentos
- Licencia de código abierto, denominada Python Software Foundation License

¿Por qué usar Python?

- Por su sintaxis simple, sencilla y clara
- Por la gran cantidad de librerías y módulos libres disponibles
- Fácil para desarrollar grandes proyectos de software
- Porque es multiplataforma

¿Quienes usan Python?

- Google
- Youtube
- Facebook
- Entre otros...

Un poco de sintaxis

1. Imprimir en consola
2. Funciones
3. IF
4. FOR
5. WHILE
6. Clases

Pueden probar la sintaxis utilizando la terminal de python

```
.
.
print 'Hello, World!'
```

```
def hello_world():
    .
    .
    return 'Hello, World!'
```

```
a = ['Mariano', 'Paulo', 'Martin']
for x in a:
    ...
    print x
    print len(x)
```

```
while b < 10:
    ...
```

```
x = 4;
if (x < 0):
    print 'Es menor'
elif (x == 0):
    print 'Es cero'
else:
    print 'Es mayor'
```

```
class Operador:
    ...
    def crear():
        ...
```

Instalar Python

- Versión recomendada: 2.7.6
  - Enlace de descarga: https://www.python.org/download/releases/2.7.6/

Observaciones

- WINDOWS
  - Para poder utilizar el comando “python” en la terminal CMD, es necesario agregar a PATH la url “C:\Python27”

Pip

Pip

- Pip es un sistema de gestión de paquetes utilizado para instalar y administrar paquetes de software escritos en Python.
- Los paquetes pueden ser encontrados en el Python Package Index (PyPI).
- Por línea de comando:

pip install nombre-paquete

pip uninstall nombre-paquete

- Se pueden listar los requerimientos en un archivo y desde allí instalarlos:

pip install -r requisitos.txt

Instalar Pip

- Windows
  - Descargar get.py https://bootstrap.pypa.io/get-pip.py
  - Acceder al directorio de descarga y abrir la terminal CMD
  - Ejecutar: python get-pip.py
  - Agregar a la variable de entorno PATH la url “C:\Python27\Scripts”
- Linux
  - Herramienta python2-pip
  - Ejemplo Debian, Ubuntu: sudo apt-get install python-pip

Flask

Flask

- Flask es un microframework escrito en Python que permite crear aplicaciones web de forma rápida con pocas líneas de código.
- Instalación: pip install Flask
- Flask permitirá **capturar los requerimientos** que lleguen al sitio web y, además, correr la aplicación en un puerto específico.
- Ejemplo:

```
from flask import Flask
from flask import request
app = Flask(__name__)

@app.route('/')
def hello_world():
   return 'Hello, World!'

if __name__ == "__main__":
            app.run(host='0.0.0.0', port=1111)
```

Formularios

Formularios

Es un elemento que define un formulario que va a ser usado para recolectar datos del usuario

<FORM **method** = [*"get"/"post"*]
action="*referencia*">

Dos atributos del form:

- **Method**: el método HTTP a generar en el requerimiento
- **Action**: el programa, archivo o recurso que procesará los datos

{<INPUT **type** = ["*radio*" / "*checkbox*" / "*submit*" / ...]
name = "*nombre*" **value**="*valor*"] ...>}

Y muchos *elementos* o *controles* (entradas, básicamente)

</FORM>

HTML - Formulario (ejemplo)

```
<HTML>
  <BODY>
    <FORM method="get" action="/form">
     Click <INPUT TYPE="SUBMIT" VALUE="aquí"> para ver el
resultado del script/programa
    </FORM>
  </BODY>
</HTML>
```

- Generará un HTTP GET
- Hace referencia al método del programa (ya sea en Python, Java, CGI, etc)
- Un solo control de entrada, de tipo SUBMIT y con “valor” “aquí”
- Texto, además del control de entrada
- Tanto el .html como el programa deben estar en donde se configuró

Primer programa en Python

Directorio de trabajo

Para poder trabajar con Python y Flask debemos tener el siguiente directorio de trabajo.

- /
  - /templates
    - Aquí van los archivos HTML
  - app.py
    - Aplicación Python
  - requirements.txt
    - Dependencias Python

¿Como crear un formulario?

1. Generar el HTML
2. Construir el controlador que renderiza el formulario
3. Construir el controlador que recibirá la información del formulario
4. Mostrar un resultado

```python
from flask import Flask
from flask import render_template
from flask import request

app = Flask(__name__)

@app.route('/')
def index():
   return render_template('form.html')

@app.route('/form')
def action_form(nom=None):
   nom = request.args["nombre"]
   return render_template('response.html', name=nom)

if __name__ == "__main__":
            app.run(host='localhost', port=80)
```

Este ejemplo: https://github.com/gmaron/tp2_classes/tree/master/python.ej1
Ejemplo completo (forms + flask + bootstrap): https://github.com/gmaron/tp2_classes/tree/master/python.ej

HTML - Formulario

```html
<form action="/form" method="POST">

    <label for="nombre">Nombre</label> <br>
    <input type="text"
           id="nombre"
           name="nombre"
           placeholder="Ingrese su nombre" required/> <br>
    <button type="submit">Registrarme!</button>

</form>
```

HTML - Respuesta

```html
<h1>Nombre: {{ nombre }} </h1>
```

Se obtiene la variable "*nombre*" que se envió como parámetro.

Python - Render Form

```python
@app.route('/')
def index():
    return render_template('form.html')
```

render_template: obtiene form.html y lo renderiza en pantalla.
@app.route('/'): es la raíz del dominio

Python - Render

```python
@app.route('/form', methods = ['POST'])
def action_form(nom=None):

    if request.method == 'POST':
        data = request.form
        nombre = data["nombre"]
        return render_template('response.html', nombre=nombre)
```

render_template: obtiene response.html y lo renderiza en pantalla, enviando como parámetro la variable "*nombre*".
@app.route('/form'): URL del formulario

Referencias

- https://www.python.org/
- http://docs.python.org.ar/tutorial/
- https://pip.pypa.io/en/stable/installing/
- Repositorio con ejemplos: https://github.com/gmaron/tp2_classes

¡Muchas gracias!

¿Preguntas?