Lección 0 - Introducción

Contenidos

Bienvenidos a la introducción a python.

A lo largo de este curso aprenderán a utilizar el versátil y poderoso lenguaje de programación Python: su sintaxis, utilidades, modos de operación, entre otros.

En esta lección introductioria, veremos: - Section **??** - Section **??** - Section **??** - Section **??** - Section 5

1 £Qué es Python?

Python es un lenguaje de programación multiparadigma, el cual es sumamente poderoso gracias a su simplicidad.

Python esta diseñado para ser legible, con un fuerte enfoque en productividad. Esto lo hace una excelente opcion como primer lenguaje de programación.

Python es usado para un sinnúmero de aplicaciones. Debajo solo algunas menciones que les serán familiares:

- Youtube estuvo en un momento, escrito mayormente en python.
- Dropbox, la compañía de almacenaje en la nube, esta hecho con python.
- EVE Online, un juego en línea, multijugador, esta escrito en una versión modificada de python.

- En la industria de cine, Industrial Light & Magic, Pixar, Dreamworks, todos usan python en alguna parte de su proceso de producción.
- La NSA utiliza python para criptografía y análisis de debilidades.
- Netflix utiliza python en su plataforma.
- Boston Dynamics utiliza python para impartir comandos a sus robots.
- D-Link utiliza python para escribir los sistemas que manejan algunos de sus dispositivos.

2 £Qué puedo lograr con python?

Python es un lenguaje que sirve para llevar a cabo tareas del mundo real, el tipo de tareas que los programadores hacen todo el tiempo.

Python es usado en una amplia variedad de dominios, ya sea como una herramienta de control sobre otros programas, o para crear programas en sí. A continuación una breve lista, con unos simples ejemplos, de lo que se puede llevar a cabo con python.

2.1 Programación de sistemas

Python cuenta con interfaces a servicios sistema operativo que lo hacen ideal para crear sistemas de control y administración del sistema operativo. Los programas de python pueden inicializar tareas del sistema operativo, monitorear su estado, lanzar otros programas, coordinar varios programas, etc.

Python cuenta con una serie de interfaces de bajo nivel, que le permiten, de ser necesario, interactuar con el hardware, y con otros interfaces del sistema, para tener aún más control sobre el sistema.

El pequeño script debajo hace un listado de los dispositivos USB conectados al equipo.

[1]:
```
import re
import subprocess
import sys

def list_usb_devices():
    device_re = re.compile("Bus\s+(?P<bus>\d+)\s+Device\s+(?P<device>\d+).+ID\s(?
 ↪P<id>\w+:\w+)\s(?P<tag>.+)$", re.I)
    df = subprocess.check_output("lsusb")
    devices = []
    for i in df.decode("utf-8").split('\n'):
        if i:
            info = device_re.match(i)
            if info:
                dinfo = info.groupdict()
                dinfo['device'] = '/dev/bus/usb/%s/%s' % (dinfo.pop('bus'), dinfo.
 ↪pop('device'))
                devices.append(dinfo)

    return devices
```

```
if __name__ == "__main__":
    f = "{id}\t{device}\t{tag}\n"
    for d in list_usb_devices():
        sys.stdout.write(f.format(**d))
```

```
8087:8000       /dev/bus/usb/003/002    Intel Corp. Integrated Rate Matching Hub
1d6b:0002       /dev/bus/usb/003/001    Linux Foundation 2.0 root hub
8087:8008       /dev/bus/usb/001/002    Intel Corp. Integrated Rate Matching Hub
1d6b:0002       /dev/bus/usb/001/001    Linux Foundation 2.0 root hub
1d6b:0003       /dev/bus/usb/004/001    Linux Foundation 3.0 root hub
0bda:b002       /dev/bus/usb/002/004    Realtek Semiconductor Corp. Bluetooth
Radio
1c7a:0603       /dev/bus/usb/002/003    LighTuning Technology Inc. ES603 Swipe
Fingerprint Sensor
046d:c534       /dev/bus/usb/002/002    Logitech, Inc. Unifying Receiver
5986:055c       /dev/bus/usb/002/005    Acer, Inc BisonCam, NB Pro
1d6b:0002       /dev/bus/usb/002/001    Linux Foundation 2.0 root hub
```

2.2 Interfaces de usuario

La librería estándar de Python cuenta con una librería de interfaz de usuario, llamada `tkinter`, que a su vez no es más que un interfaz con una extensa libreria llamada `TK`.

Tkinter corre sin importar el sistema operativo, ya que el mismo no es más que un iterfaz a utilidades nativas.

Existen otras librerías que ofrecen soluciones más completas que `tkinter`, debajo solo hay algunas:

- wxPython https://www.wxpython.org/
- GTK con PyGTK
- QT con PyQT

El script debajo inicializa un sencillo cronómetro usando `tkinter`

```
#!/usr/bin/python3
import time
from tkinter import (
    Frame,
    Tk,
    StringVar,
    Label,
    X,
    NO,
    TOP,
    LEFT,
    Button,
)
```

```
class StopWatch(Frame):
    """Implementación de un cronómetro."""
    def __init__(self, parent=None, **kw):
        Frame.__init__(self, parent, kw)
        self._start = 0.0
        self._elapsedtime = 0.0
        self._running = 0
        self.timestr = StringVar()
        self.make_widgets()

    def make_widgets(self):
        """Crea el reloj."""
        l = Label(self, textvariable=self.timestr)
        self._set_time(self._elapsedtime)
        l.pack(fill=X, expand=NO, pady=2, padx=2)

    def _update(self):
        """Actualiza el reloj con el tiempo."""
        self._elapsedtime = time.time() - self._start
        self._set_time(self._elapsedtime)
        self._timer = self.after(50, self._update)

    def _set_time(self, elap):
        """Se encarga de dar formato al reloj en minutors:segundos:centesimas."""
        minutes = int(elap/60)
        seconds = int(elap - minutes*60.0)
        hseconds = int((elap - minutes*60.0 - seconds)*100)
        self.timestr.set('%02d:%02d:%02d' % (minutes, seconds, hseconds))

    def start(self):
        """Arranca el cronómetro."""
        if not self._running:
            self._start = time.time() - self._elapsedtime
            self._update()
            self._running = 1

    def stop(self):
        """Detiene el cronómetro."""
        if self._running:
            self.after_cancel(self._timer)
            self._elapsedtime = time.time() - self._start
            self._set_time(self._elapsedtime)
            self._running = 0

    def reset(self):
        """Detiene el cronómetro."""
        self._start = time.time()
        self._elapsedtime = 0.0
```

```
        self._set_time(self._elapsedtime)


def main():
    root = Tk()
    sw = StopWatch(root)
    sw.pack(side=TOP)

    Button(root, text='Iniciar', command=sw.start).pack(side=LEFT)
    Button(root, text='Detener', command=sw.stop).pack(side=LEFT)
    Button(root, text='Reinicio', command=sw.reset).pack(side=LEFT)
    Button(root, text='Cerrar', command=root.quit).pack(side=LEFT)

    root.mainloop()

if __name__ == '__main__':
    main()
```

2.3 Scripting

Python cuenta con muchas utilidades para manipulación de datos, desde manejo de archivos hasta protocolos como HTTP, Websocket, RPC.

Python es muy utilizado para automatización de infraestructura: tareas orientadas a inicialización y mantenimiento de sistemas; en ocasiones estos sistemas se encuentran distribuidos en una nube.

Adicionalmente, python cuenta con marcos de desarrollo completos para aplicaciones, los cuales permiten crear aplicaciones interactivas con impresionante rapidez.

[2]:
```
import sys
import requests
import bs4 # BeautifulSoup

URL = 'https://es.wikipedia.org/wiki/Anexo:Comparaci%C3%B3n_de_procesadores_Intel'

def get_table():
    res = requests.get(URL)
    soup = bs4.BeautifulSoup(res.text, features="html5lib")

    table = soup.select("table")
    headers = table[1].select("th")
    data = table[1].select("td")

    return headers, data


def filter_columns(headers, data, want=[]):
```

```
    idx = []
    for w in want:
        for h in headers:
            if w.lower() in h.getText().lower():
                idx.append(headers.index(h))
    new_headers = []
    for i in idx:
        new_headers.append(headers[i].getText())

    new_data = []
    for k in range(0, len(data), len(headers)):
        d = []
        for i in idx:
            d.append(data[k + i].getText())
        new_data.append(d)


    return new_headers, new_data

if __name__ == "__main__":
    headers, data = get_table()
    headers, data = filter_columns(headers, data, ["procesador", "nombre del␣
 ↪código", "frecuencia"])
    for h in headers:
        sys.stdout.write("{}\t".format(h))
    sys.stdout.write("\n"+"-"*64+"\n")
    sys.stdout.write("\n")
    for d in data:
        sys.stdout.write("{}\t{}\t{}\n".format(*d))
```

```
Procesador      Nombre del código       Frecuencia del reloj
----------------------------------------------------------------

Intel Pentium   P5, P54C, P54CTB, P54CS 60 MHz - 200 MHz
Intel Pentium MMX       P55C, Tillamook 120 MHz - 300 MHz
Intel Atom      Diamondville, Pineview, Silverthorne, Lincroft, Cedarview,
Medfield, Clover Trail       800 MHz - 2.13 GHz
Intel Celeron   Banias, Cedar Mill, Conroe, Coppermine, Covington, Dothan ,
Mendocino, Northwood, Prescott, Tualatin, Willamette, Yonah 266 MHz - 3,6 GHz
Intel Pentium Pro       P6      150 MHz - 200 MHz
Intel Pentium II        Klamath, Deschutes, Tonga, Dixon        233 MHz - 450
MHz
Intel Xeon      Allendale, Cascades, Clovertown, Conroe, Cranford, Dempsey,
Drake, Dunnington, Foster, Gainestown, Gallatin, Harpertown, Irwindale,
Kentsfield, Nocona, Paxville, Potomac, Prestonia, Sossaman, Tanner, Tigerton,
Tulsa, Wolfdale, Woodcrest    400 MHz - 4,4 GHz
Pentium 4       Cedar Mill, Northwood, Prescott, Willamette     1.3 GHz - 3.8
```

```
GHz
Pentium 4 Extreme Edition       Gallatin. Prescott 2M   3.2 GHz - 3,73 GHz
Pentium M       Banias, Dothan  800 MHz - 2.266 GHz
Pentium D / EE  Smithfield, Presler     2.66 GHz - 3.73 GHz
Intel Pentium Dual-Core Allendale, Penryn, Wolfdale, Yonah      1.6 GHz - 2.93
GHz
Intel Pentium New       Penryn, Wolfdale, Clarkdale, Sandy Bridge ,     1.2 GHz
- 3.33 GHz
Intel Core      Yonah   1.06 GHz - 2.33 GHz
Intel Core 2    Allendale, Conroe, Merom, Penryn, Kentsfield, Wolfdale,
Yorkfield       1.06 GHz - 3.33 GHz
Intel Core i3   Arrandale, Clarkdale, Sandy Bridge, Ivy Bridge  2,4 GHz - 3,4
GHz
Intel Core i5   Arrandale, Clarkdale, Clarksfield, Lynnfield, Sandy Bridge, Ivy
Bridge  1,06 GHz - 3,46 GHz
Intel Core i7   Bloomfield, Nehalem, Clarksfield, Clarksfield XM, Lynnfield,
Sandy Bridge, Sandy Bridge-E, Ivy Bridge, Haswell  1.6 GHz - 3.6 GHz
Intel Core i9   Gulftown, Sandy Bridge-E        3.2 GHz - 3,46 GHz
```

3 Instalación de python

Es probable que ya tenga alguna version de python instalada en su sistema. En este curso estaremos usando `python3`.

Puede descargar e instalar la última versión de https://python.org/downloads/.

Los métodos más abajo se refieren a los diferentes package manager de cada sistema. Si ya descargó python del enlace de arriba, puede saltar esta sección.

3.1 Mac OS X (>10.9)

- Instale `Homebrew`. Desde la terminal:

  ```
  ~$ /bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/instal
  ```

- Luego instale `python3` usando `Homebrew`:

  ```
  ~$ brew install python
  ```

3.2 Windows

Descarge e instale `chocolatey` de https://chocolatey.org/install#individual, luego, desde `powershell`:

```
choco install python3
```

3.3 Linux

- Utilice su package manager para instalar python:

- Ubuntu / debian:

```
~$ sudo add-apt-repository ppa:deadsnakes/ppa
~$ sudo apt get update
~$ sudo apt get python3.9
```

- Arch:

```
~$ sudo pacman -S python
```

- Fedora linux:

```
~$ sudo dnf install python3.9
```

- RedHat / Centos / RHEL

```
~$ sudo yum install python3
```

4 Intérprete

Es posible iniciar el intérprete de `python` e inmediatamente interactuar con el.

- En Windows, abra el "Windows Run" con `Win+R`, e ingrese "python" en la barra.
- En MacOS o Linux, abra una terminal y escriba `python3`.

Python es un lenguaje de programación, en el fondo, igual que todos los lenguajes de programación, son instrucciones para hablar con el computador.

```
~$ python
```

```
Python 3.8.6 (default, Sep 30 2020, 04:00:38)
[GCC 10.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> 1+1
2
>>> 5-2
3
>>> 3*4
12
>>> respuesta = input("¿Cual es tu nombre?")
¿Cual es tu nombre?Python 3!
>>> respuesta
'Python 3!'
>>>
```

En el ejemplo anterior, usamos la funcion nativa `input` para capturar texto del usuario. Esta función guarda su resultado en la variable `respuesta` que creamos a la izquierda.

Una variable, al igual que en matemáticas, es objeto cuyo valor puede ser cualquier cosa. En este caso, es el `string` que uno le escribe como respuesta al programa.

De igual forma, uno puede asignar valores a las variable, y realizar cálculos con las mismas.

```
>>> x = 2
>>> y = x**2 # ** es potenciación
>>> print(y)
4
>>>
```

Lo que sigue después del numeral (#) es un comentario, `python` ignora todo lo que le sigue, permitiendo hacer anotaciones sobre el código.

El intérprete es util para experimentar y evaluar valores en vivo. Sin embargo, el mayor uso que se le dará a python será a través de scripts.

5 Scripts

Lo que se llama un "script" no es mas que una serie de instrucciones en secuencia, las cuales serán ejecutadas por `python` al llamar el archivo.

Abra su editor favorito, cree un archivo nuevo llamado `tarjeta.py` e ingrese el código siguiente:

[3]:
```
#!/usr/bin/python3

nombre = input("Ingrese su nombre: ")
edad = input("Ingrese su edad: ")
profesion = input("Ingrese su profesión: ")

print("""
Nombre     {}
Edad:      {}
Profesión: {}
""".format(nombre, edad, profesion))
```

```
---------------------------------------------------------------------------

StdinNotImplementedError                  Traceback (most recent call last)

<ipython-input-3-231905278a0e> in <module>
  1 #!/usr/bin/python3
  2
----> 3 nombre = input("Ingrese su nombre: ")
  4 edad = input("Ingrese su edad: ")
  5 profesion = input("Ingrese su profesión: ")


~/go/src/github.com/djangulo/introducción-a-python/.env/lib/python3.8/
↪site-packages/ipykernel/kernelbase.py in raw_input(self, prompt)
  851         """
  852         if not self._allow_stdin:
--> 853             raise StdinNotImplementedError(
```

```
      854                 "raw_input was called, but this frontend does not␣
↪support input requests."
      855             )


      StdinNotImplementedError: raw_input was called, but this frontend does not␣
↪support input requests.
```

Vamos a desglosar linea por linea:

```
1: #!/usr/bin/python3
```

Esto se conoce como "shebang", y le indica al sistema operativo que programa utilizar para ejecutar el archivo. Debe de indicar la ruta hacia el ejecutable del programa que lo correrá (en este caso, python3). Tenga en cuenta que si está en Windows, debe de escapar los backslash (), su ruta se vería, por ejemplo, así: `#!C:Program Files\\python\\python3.exe`. De no tener el shebang, para correr el script es necesario usar `python` para directamente ejecutar el archivo:

```
~$ python3 tarjeta.py
```

Ya que este lo tiene, el script se puede ejecutar como un programa

```
~$ ./tarjeta.py
```

```
3: nombre = input("Ingrese su nombre: ")
4: edad = input("Ingrese su edad: ")
5: profesion = input("Ingrese su profesión: ")
```

Las líneas 3-5 declaran variables, que capturan la información escrita por el usuario.

```
 7: print("""
 8: Nombre     {}
 9: Edad:      {}
10: Profesión: {}
11: """.format(nombre, edad, profesion))
```

Las líneas 7-11 ejecutan un comando de impresión y formato. En `python`, los `str`ings representan cadenas de caracteres: texto, en español. Un string en `python` se puede ingresar entre comillas simples `'texto'`, comillas dobles `"texto"`. Se puedes usar comillas triples (`"""texto"""` o `'''texto'''`) para escribir `str`ings de múltiples líneas.

Las llaves (`{}`) le indican al método `format` donde colocar sus argumentos, y usan lo que se conoce como interpoación para popular la "plantilla".