



Máster Blockchain y Big Data con Python

UNIDAD 2 Python Parte 1

Objetivos Generales del Curso:

- **Semana 1:**
 - Introducción General. Aprendiendo a programar.
(Nuestro primer programa: un videojuego)
- **Semana 2:**
 - Python - Parte I.
(Desarrollo de un Sistema Cálculo de Presentismo
Creación de un Asistente por Voz)
- **Semana 3:**
 - Python – Parte II.
(Creación de un Pliego Particular Jurídico usando I.A. - Sistemas Expertos - Probab. Bayesianas – Proc. Leng. Natural)
- **Semana 4:**
 - Base de Datos.
(Desarrollo de un Sistema Experto de Diagnóstico Médico)
- **Semana 5:**
 - Web Scraping y RPA.
(Desarrollo de un Sistema de Extracción de Datos de una Web)
- **Semana 6:**
 - Introducción al Big Data y al Deep Learning.
(Desarrollo de un Sistema de Reconocimiento de Personas)
- **Semana 7:**
 - Dialogflow de Google. Watson de IBM.
(Desarrollo de un Chatbot de Consultas)
- **Semana 8:**
 - Introducción a las Redes Neuronales y Algoritmos de Minería de Datos.

Objetivos Generales del Curso:

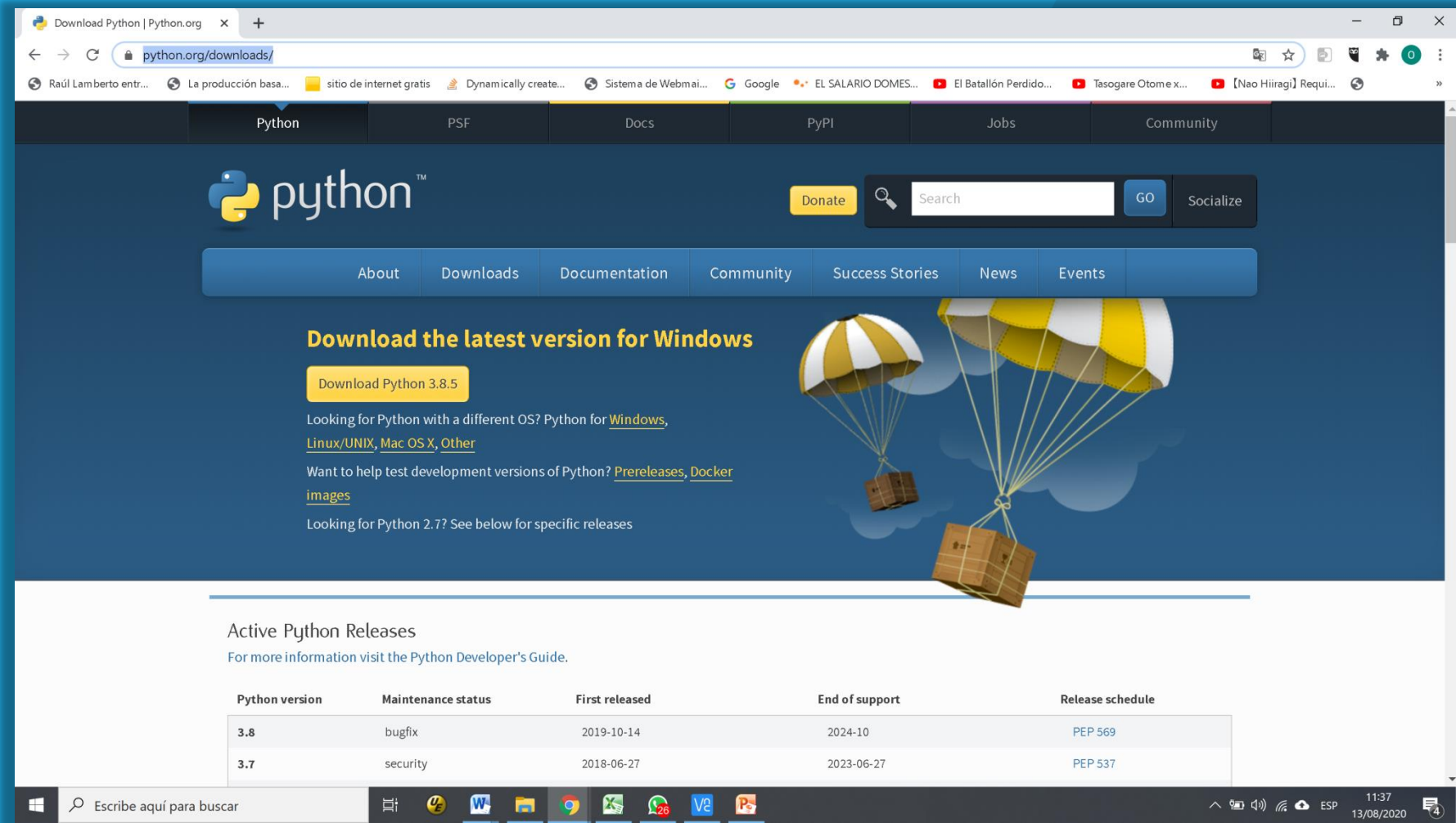
- **Semana 9:**
 - Herramientas de Big Data: Hadoop, Spark, Hive, Pig.
- **Semana 10:**
 - BigData para chicos malos. Hacking y Seguridad de los Datos.
- **Semana 11:**
 - Datamining y Machine Learning con Python Parte I.
(Sistema de recomendación de películas)
- **Semana 12:**
 - Datamining y Machine Learning con Python Parte II.
(Sistema de recomendación de películas)
- **Semana 13:**
 - Introducción a Blockchain y Criptomonedas.
(Nuestro primer programa empleando blockchain)
- **Semana 14:**
 - Smart Contracts con Python.
(Nuestro primer Smart Contract)
- **Semana 15:**
 - Presentación de Desafíos para Proyectos Finales.
- **Semana 16:**
 - Presentación de Proyectos Finales.

Objetivos de esta clase:

- Instalar Python. Editor de Python. IDEs de desarrollo.
- Estructura de un programa.
- Diagramas de flujo.
- Programación Orientada a Objetos.
- Entrada / Salida.
- Variables y tipos de datos. Nomenclatura para el uso de variables.
- Comparación y operadores relacionales.
- Ciclos (bucles).
- Desarrollo 1: Sistema de Cálculo de Presentismo (versión por consola, versión Desktop)
- Desarrollo 2: Sist. de Reconocimiento de Voz y Rptas. Habladas (Asistente por Voz)
- Tareas para la Casa



Instalar Python



The screenshot shows the Python.org website's download page. The browser's address bar displays 'python.org/downloads/'. The page features the Python logo, a search bar, and a navigation menu with links to 'About', 'Downloads', 'Documentation', 'Community', 'Success Stories', 'News', and 'Events'. A prominent section titled 'Download the latest version for Windows' includes a 'Download Python 3.8.5' button and links for other operating systems. Below this, a table titled 'Active Python Releases' provides details on the current and previous versions.

Download the latest version for Windows

[Download Python 3.8.5](#)

Looking for Python with a different OS? Python for [Windows](#), [Linux/UNIX](#), [Mac OS X](#), [Other](#)

Want to help test development versions of Python? [Prereleases](#), [Docker images](#)

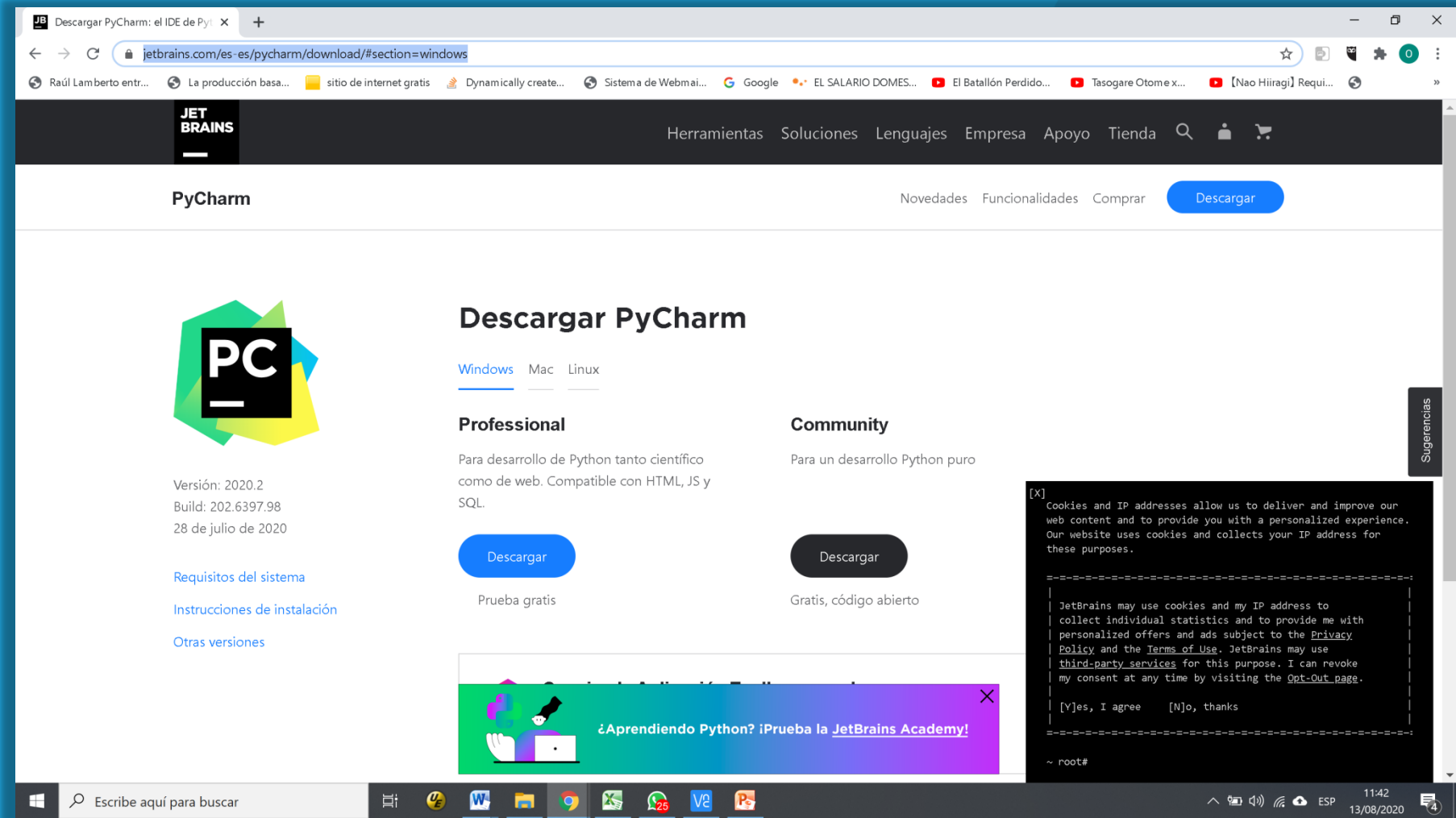
Looking for Python 2.7? See below for specific releases

Active Python Releases
For more information visit the [Python Developer's Guide](#).

Python version	Maintenance status	First released	End of support	Release schedule
3.8	bugfix	2019-10-14	2024-10	PEP 569
3.7	security	2018-06-27	2023-06-27	PEP 537

<https://www.python.org/downloads>

Instalar PyCharm



The screenshot shows the JetBrains website's PyCharm download page for Windows. The page features the PyCharm logo, version information (2020.2), and download links for Professional and Community editions. A cookie consent banner is visible on the right, and a banner for JetBrains Academy is at the bottom.

Descargar PyCharm

Windows Mac Linux

Professional
Para desarrollo de Python tanto científico como de web. Compatible con HTML, JS y SQL.
[Descargar](#)
Prueba gratis

Community
Para un desarrollo Python puro
[Descargar](#)
Gratis, código abierto

Versión: 2020.2
Build: 202.6397.98
28 de julio de 2020

[Requisitos del sistema](#)
[Instrucciones de instalación](#)
[Otras versiones](#)

¿Aprendiendo Python? ¡Prueba la [JetBrains Academy](#)!

Cookies and IP addresses allow us to deliver and improve our web content and to provide you with a personalized experience. Our website uses cookies and collects your IP address for these purposes.

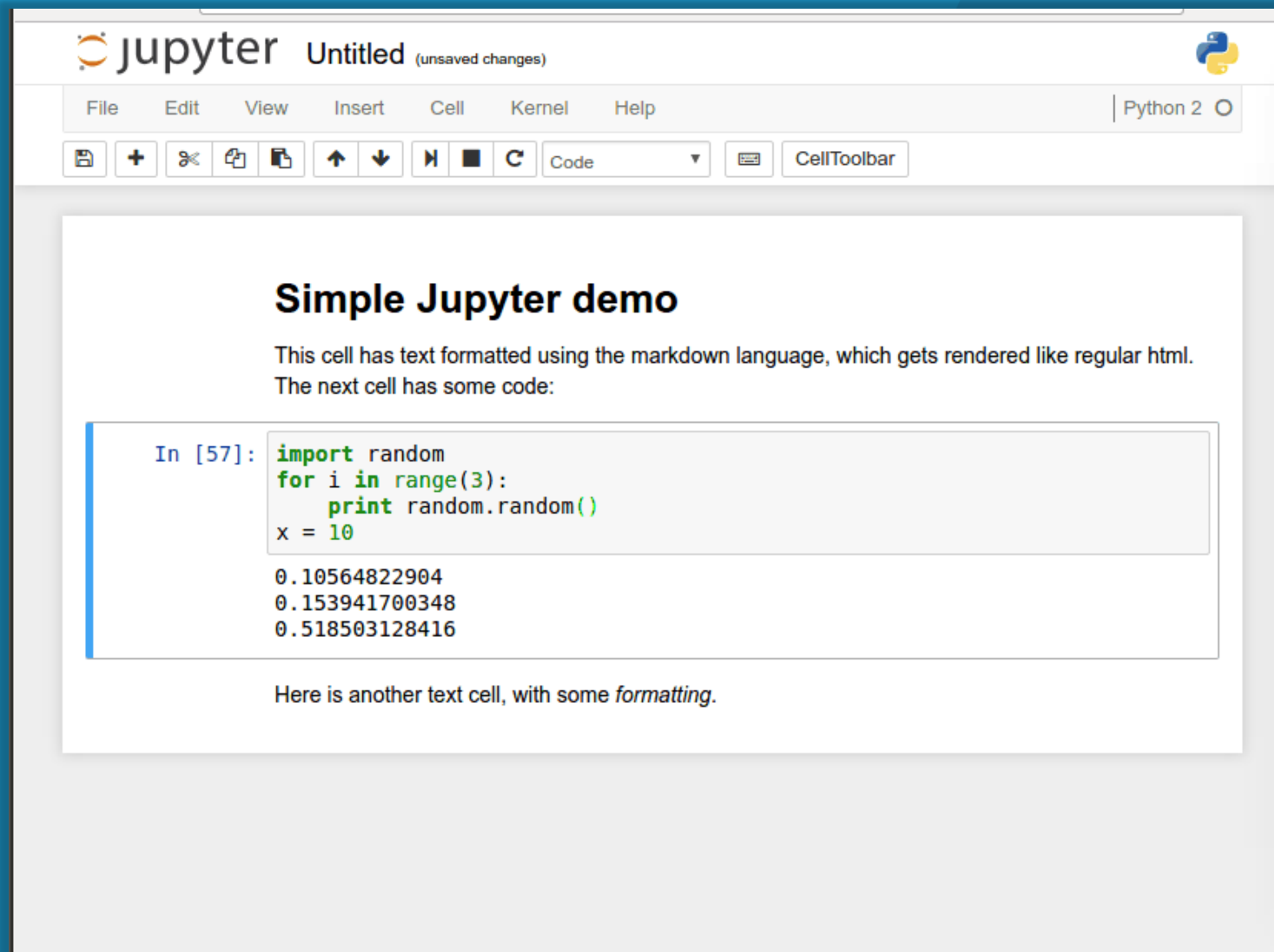
JetBrains may use cookies and my IP address to collect individual statistics and to provide me with personalized offers and ads subject to the [Privacy Policy](#) and the [Terms of Use](#). JetBrains may use [third-party services](#) for this purpose. I can revoke my consent at any time by visiting the [Opt-Out page](#).

[Y]es, I agree [N]o, thanks

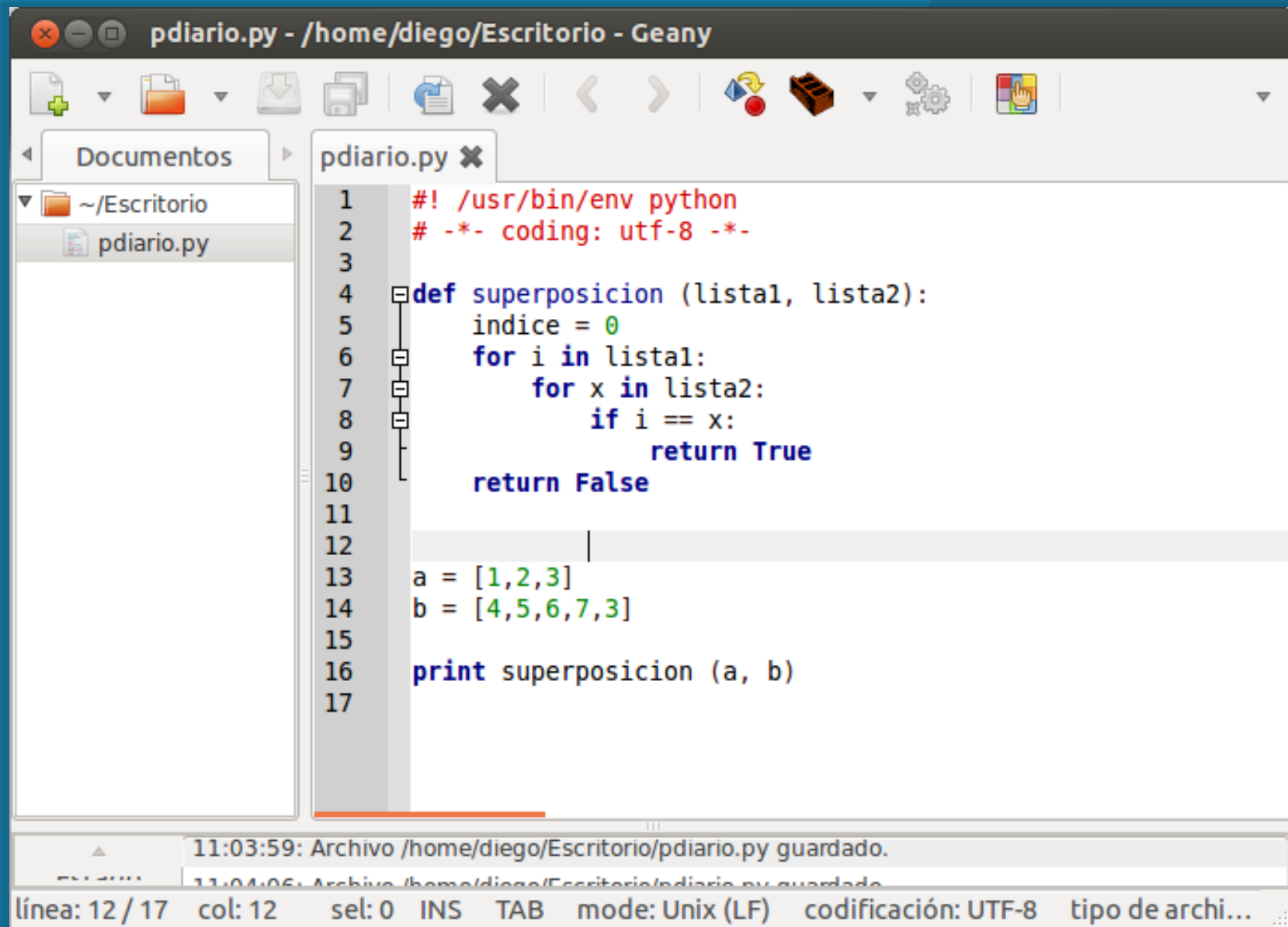
~ root#

<https://www.jetbrains.com/es-es/pycharm/download/#section=windows>

Jupyter Notebook



Estructura de un programa Python



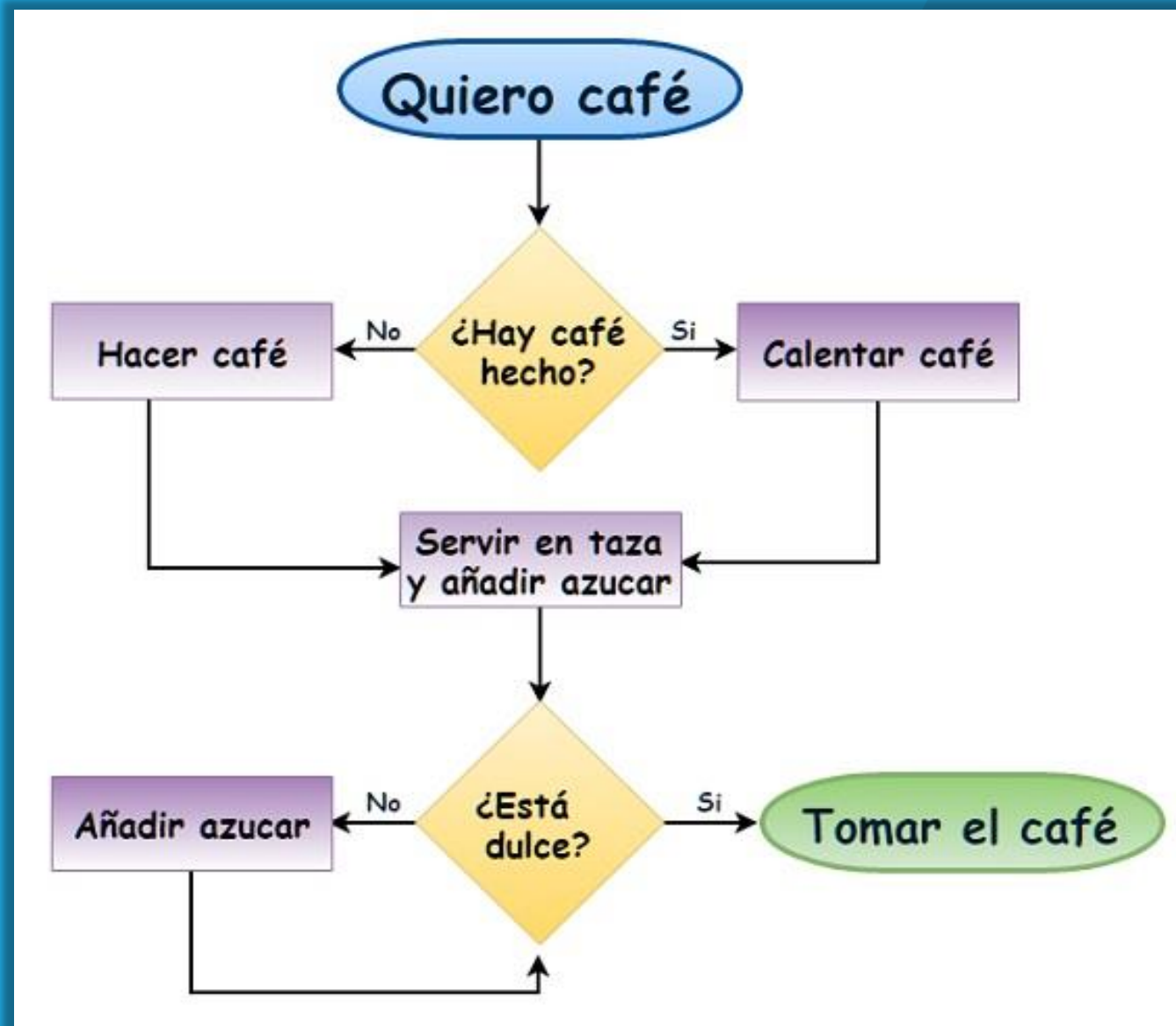
```
pdiario.py - /home/diego/Escritorio - Geany

1  #!/usr/bin/env python
2  # -*- coding: utf-8 -*-
3
4  def superposicion (lista1, lista2):
5      indice = 0
6      for i in lista1:
7          for x in lista2:
8              if i == x:
9                  return True
10     return False
11
12
13     a = [1,2,3]
14     b = [4,5,6,7,3]
15
16     print superposicion (a, b)
17
```

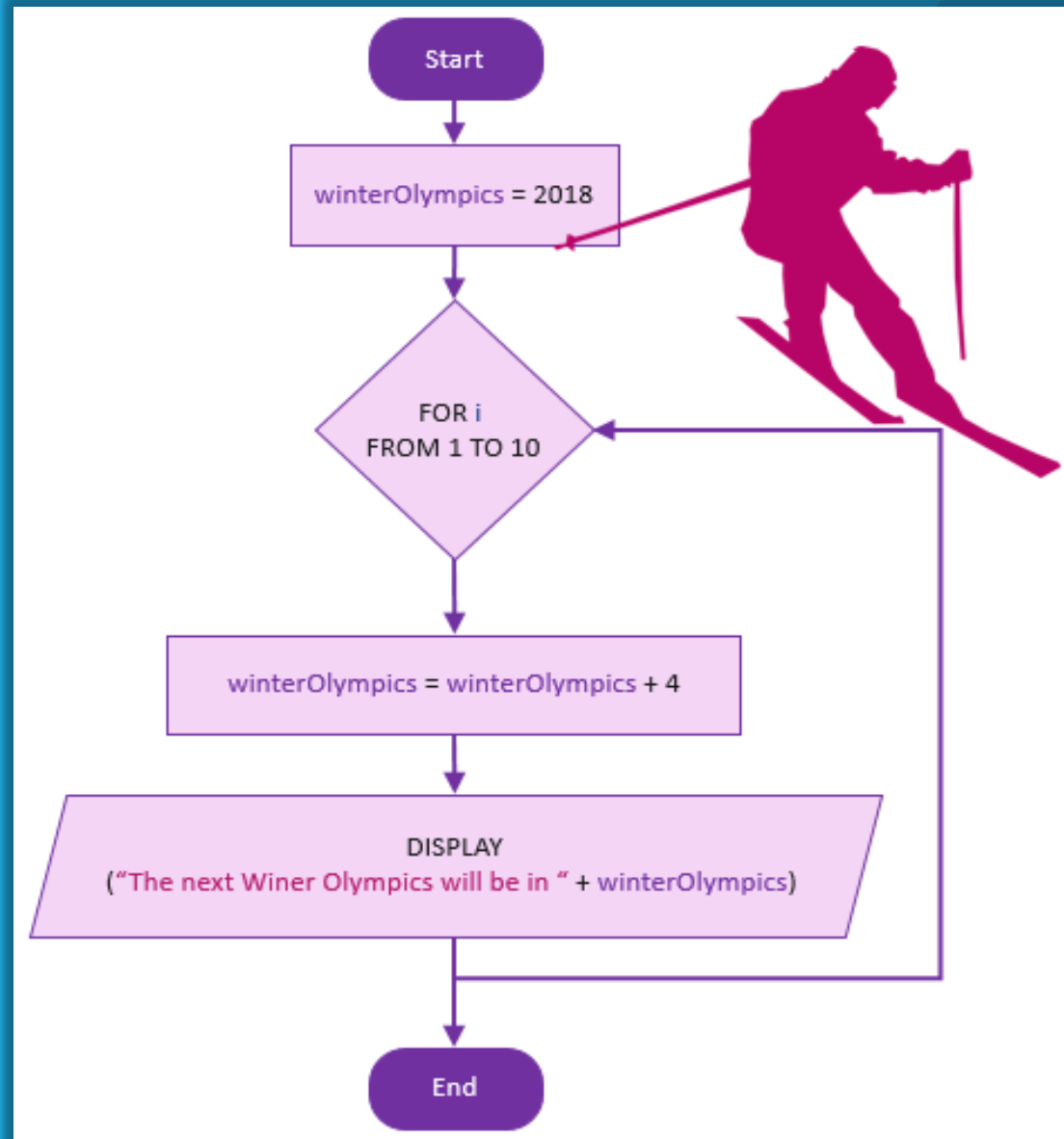
11:03:59: Archivo /home/diego/Escritorio/pdiario.py guardado.
11:04:06: Archivo /home/diego/Escritorio/pdiario.py guardado.

línea: 12 / 17 col: 12 sel: 0 INS TAB mode: Unix (LF) codificación: UTF-8 tipo de archi...



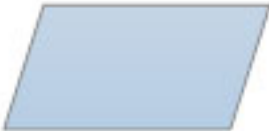


Diagramas de Flujo



Diagramas de Flujo



Diagramas de Flujo

Símbolo	Nombre	Función
	Inicio / Final	Representa el inicio y el final de un proceso
	Linea de Flujo	Indica el orden de la ejecución de las operaciones. La flecha indica la siguiente instrucción.
	Entrada / Salida	Representa la lectura de datos en la entrada y la impresión de datos en la salida
	Proceso	Representa cualquier tipo de operación
	Decisión	Nos permite analizar una situación, con base en los valores verdadero y falso

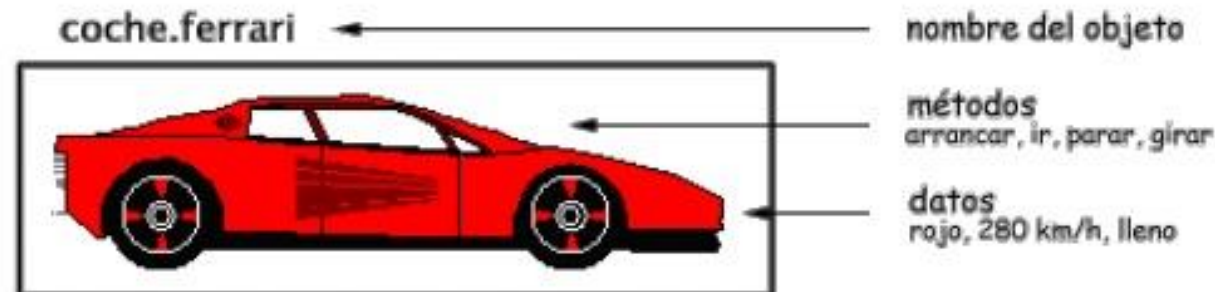
POO (Programación Orientada a Objetos)

Ejemplo de clases y objetos

- Clase: *Coche*



- ♦ Objeto: *Ferrari*



POO (Programación Orientada a Objetos)

Atributos: Son los datos o variables que caracterizan al objeto y cuyos valores en un momento dado indican su estado.

Métodos: Son las operaciones (acciones o funciones) que se aplican sobre los objetos y que permiten crearlos, cambiar su estado o consultar el valor de sus atributos



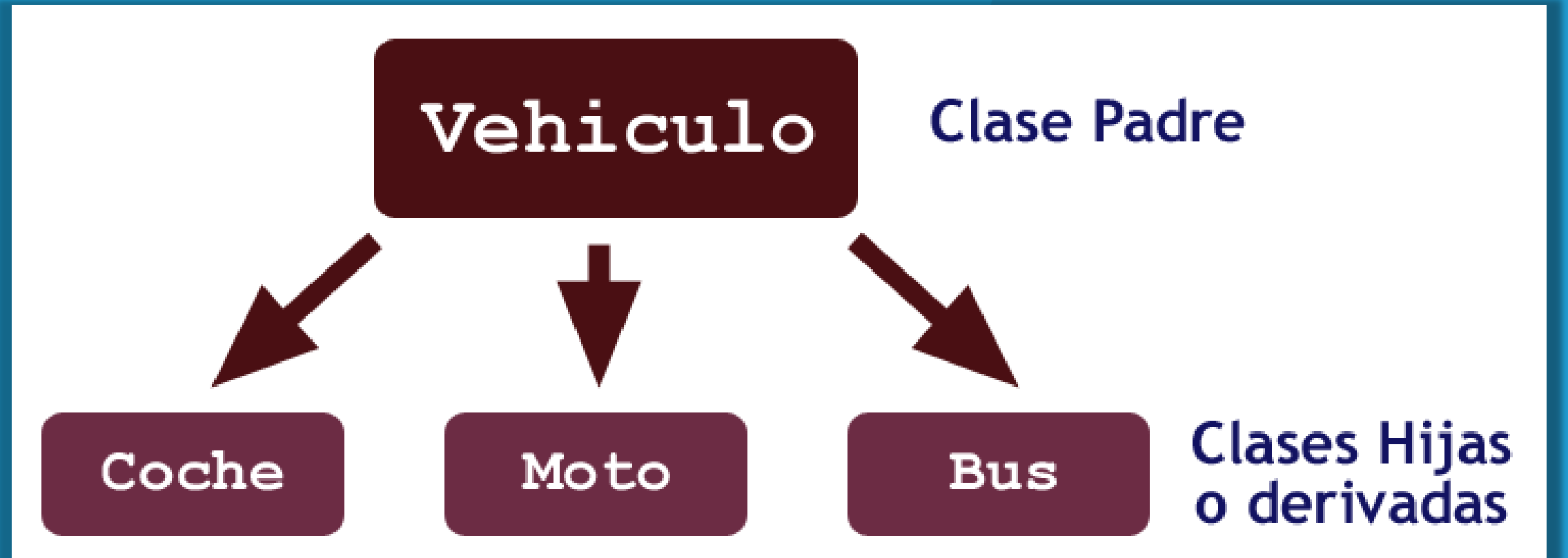
▪ Atributos:

- color
- velocidad
- ruedas
- motor

▪ Métodos:

- arranca()
- frena()
- dobla()

POO (Programación Orientada a Objetos)



POO (Programación Orientada a Objetos)

Declaro la función:

```
function estacionar ( Vehiculo ) { }
```

Invoco la función: (soporto polimorfismo)

```
estacionar ( Coche ) ;
```

```
estacionar ( Moto ) ;
```

```
estacionar ( Bus ) ;
```

No puedo invocar la función: (no lo permitiría, porque no ser clasificación de herencia de vehículos)

```
estacionar ( Mono ) ;
```

```
estacionar ( INT ) ;
```

En el futuro si podría: (Si creo las clases "Van" o "Nave espacial" y heredan de Vehiculo)

```
estacionar ( Van ) ;
```

```
estacionar ( Nave espacial ) ;
```

POO (Programación Orientada a Objetos)

```
1  CREAR UN PROGRAMA QUE PERMITA CREAR LA CLASE ESTUDIANTE CAPAZ DE ALMACENAR INFORMACION
2  DE UN ESTUDIANTE: NOMBRE_COMPLETO, CARNET, CARRERA Y EDAD. MOSTRAR DICHA INFORMACION
3
4  class Estudiante:
5
6      def __init__(self,nombres,apellidos,carnet,carrera,edad):
7          self.nombres = nombres
8          self.apellidos = apellidos
9          self.carnet = carnet
10         self.carrera = carrera
11         self.edad = edad
12
13         # Metodos Accesores
14         def getNombres(self):
15             return self.nombres
16         def getApellidos(self):
17             return self.apellidos
18         def getCarnet(self):
19             return self.carnet
20         def getCarrera(self):
21             return self.carrera
22         def getEdad(self):
23             return self.edad
24
25         def imprimirEstudiante(self):
26             print("\nNombres: " +self.getNombres()+"\nApellidos: "+self.getApellidos()+"\nCarnet: "+self.getCarnet()+
27                 "\nCarrera: " +self.getCarrera()+"\nEdad: "+str(self.getEdad()) )
28
29         nombres = raw_input("Favor ingresar los nombres: ")
30         apellidos = raw_input("Favor ingresar los apellidos: ")
31         carnet = raw_input("Favor ingresar el numero de carnet: ")
32         carrera = raw_input("Favor ingresar nombre de carrera: ")
33         edad = raw_input("Favor ingresar edad: ")
34         e = Estudiante(nombres,apellidos,carnet,carrera,edad)
35         e.imprimirEstudiante()
```

Entrada y Salida en Python

input.py ✕

```
1  #!/usr/bin/python
2  # -*- coding: utf-8 -*-
3
4  suma = input("Ingrese la suma deseada: ")
5
6  print ""
7  print "La suma es:", suma
8
```


Variable y Tipos de Datos

Los tipos (clases) básicos de datos (objetos) en Python son:

- **Números.** Ejemplos de valores (atributos):
 - 3 (tipo/clase: entero, integer, **int**)
 - 21 (tipo/clase: entero, integer, **int**)
 - 15.57 (tipo/clase: real, coma flotante, floating point, **float**)
 - 7+5j (tipo/clase: complejo, **complex**)
- **Cadenas de texto** (String, **str**).
Ejemplos: "hola mundo", "3", '37+5', "Nacho's", '123@s'
- **Booleanos.** (Boolean, "bool"). Solo dos valores (atributos):
True (verdadero) y **False** (falso)

Variable y Tipos de Datos

- Python es case sensitive (distingue mayúsculas y minúsculas), por lo que no será lo mismo una variable llamada "nombre" que "Nombre" o "nomBRE" por ejemplo. Una buena práctica, es escribir el nombre de las variables siempre en minúsculas, de ese modo es más difícil equivocarse y no tienes que recordar si escribiste alguna letra de la variable en Mayúscula.
- Usar nombres descriptivos. Imagina que tienes dos variables la primera almacena un nombre y la segunda los apellidos. Una mala práctica sería llamar a estas variables por ejemplo "n" y "a" en lugar de por ejemplo "nombre" y "apellidos".
- Cuando vaya a escribir un nombre de variable formado por varias palabras, nunca dejes espacios. Debes escribirlo con barra baja o con mayúsculas, como te vaya mejor. Ejemplo: nombre_empleado o nombreEmpleado
- Es recomendable hacer una distinción entre variables globales y locales.
- Los nombres de variables no deben coincidir con una palabra reservada de Python.

Comparación y Operadores Relacionales

```
1  # La función input() nos permite asignar a una variable
→ 2  a = input()
→ 3  b = input()
4  if a == b:
5      print('Los números', a, 'y', b, 'son iguales')
6  elif a < b:
7      print('El número', a, 'es menor que', b)
8  else:
9      print('El número', b, 'es menor que', a)
```

Comparación y Operadores Relacionales

SÍMBOLO	DESCRIPCIÓN	EJEMPLO	BOOLEANO
==	IGUAL QUE	5 == 7	FALSE
!=	DISTINTO QUE	ROJO != VERDE	TRUE
<	MENOR QUE	8 < 12	TRUE
>	MAYOR QUE	12 > 7	TRUE
<=	MENOR O IGUAL QUE	16 <= 17	TRUE
>=	MAYOR O IGUAL QUE	67 >= 72	FALSE

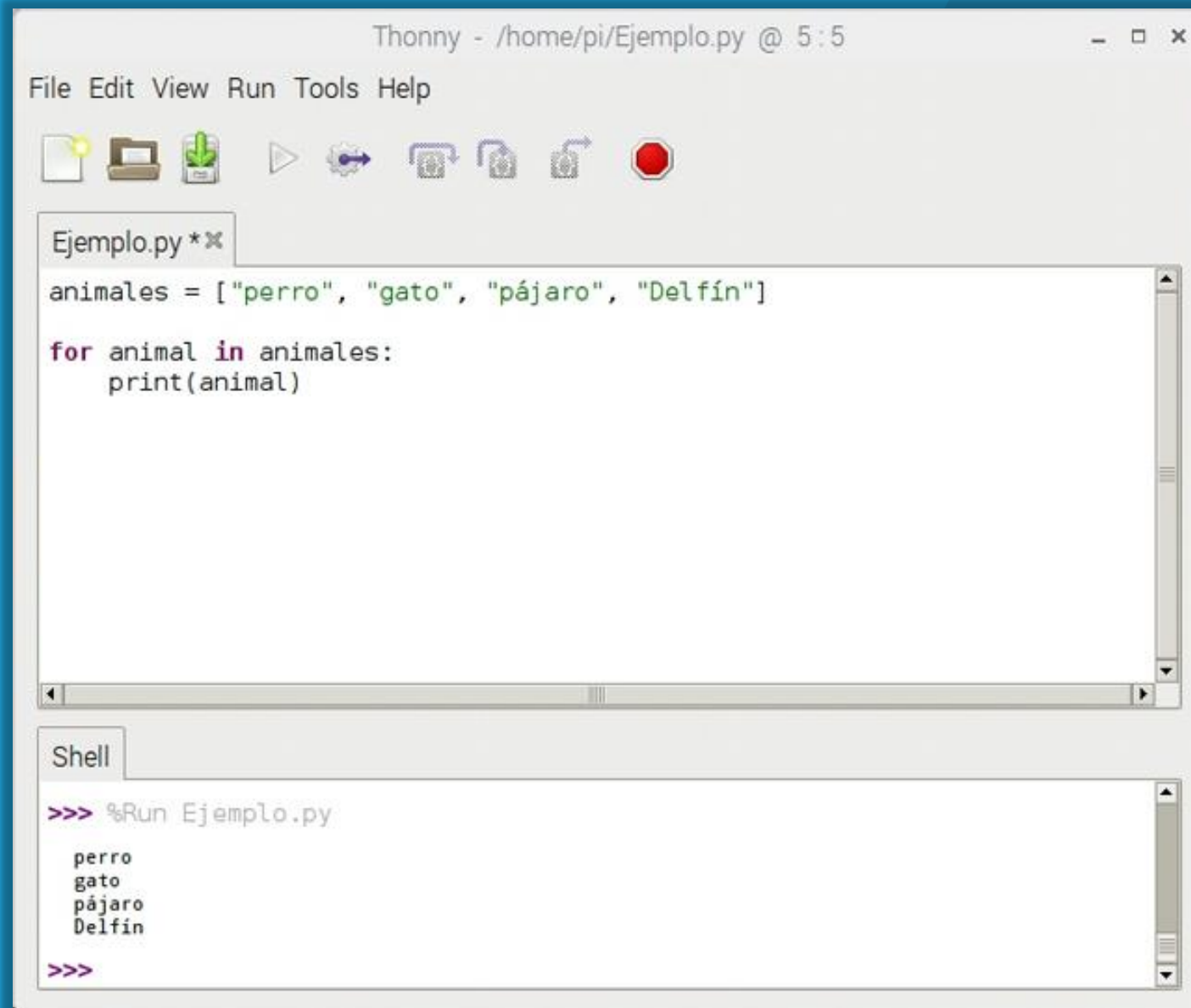
Ciclos: for

```
*funciones.py - E:/Practicas Python 2013/funciones.py*
File Edit Format Run Options Windows Help

"""
Una funcion es conjunto de lineas de codigo
que realizan una tarea específica
***Reutilizacion de codigo
***Divide y Venceras
Sintaxis para crear una función
def nombre_funcion(parametros):
    Accion1
    Accion2
    AccionN
Sintaxis para mandar llamar a una función
nombre_funcion(parametros)
"""
def imprime_diez():
    for i in range(1,11):
        print (i)
```

Ln: 15 Col: 4

Ciclos: for



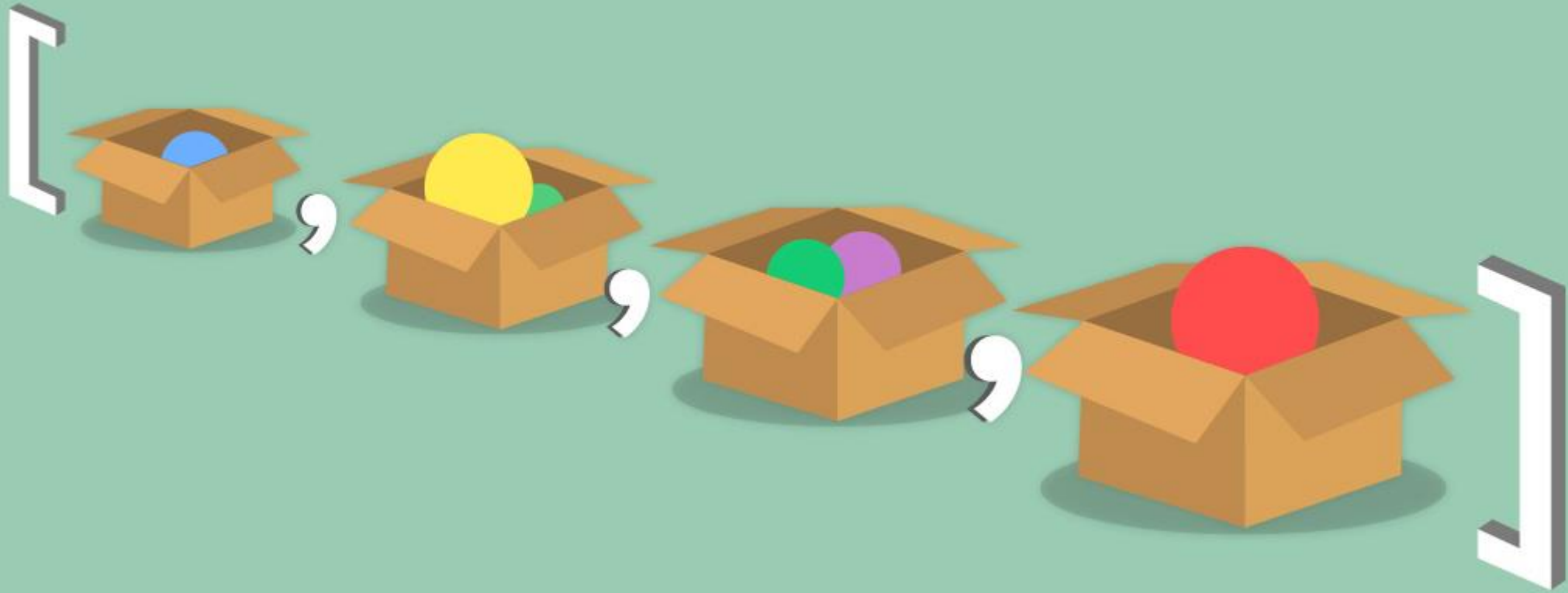
The image shows a screenshot of the Thonny Python IDE. The window title is "Thonny - /home/pi/Ejemplo.py @ 5:5". The menu bar includes "File", "Edit", "View", "Run", "Tools", and "Help". The toolbar contains icons for opening a file, saving, running, stepping through code, and other development tools. The main editor window shows a Python script named "Ejemplo.py" with the following code:

```
Ejemplo.py *  
animales = ["perro", "gato", "pájaro", "Delfín"]  
  
for animal in animales:  
    print(animal)
```

Below the editor is a "Shell" window showing the output of running the script:

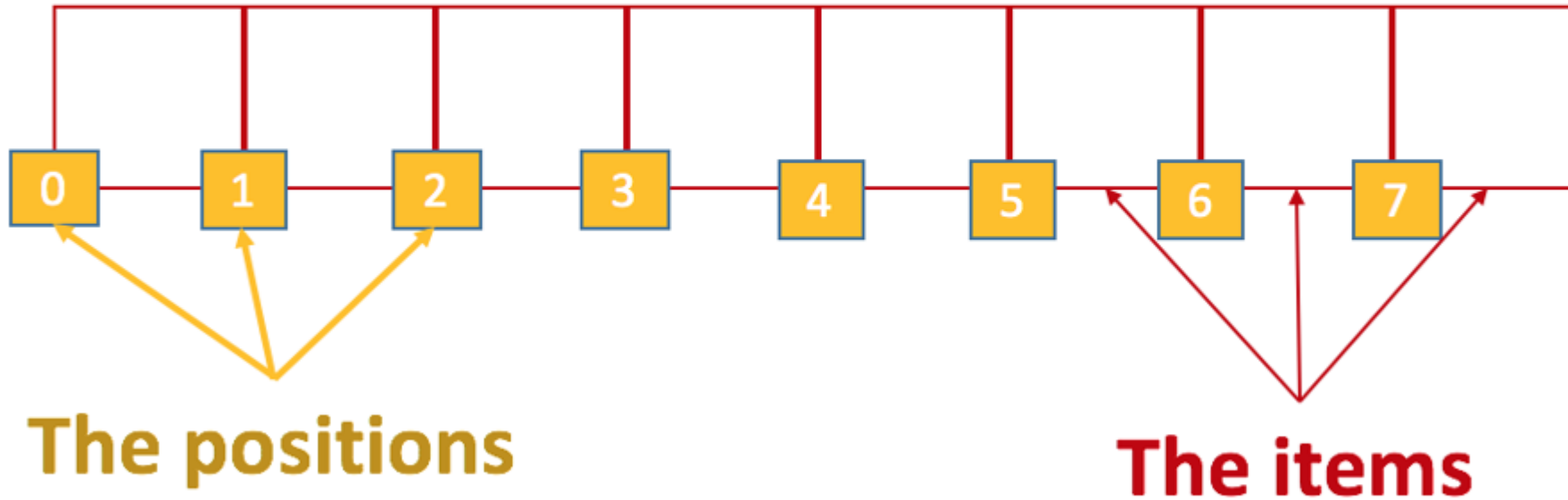
```
>>> %Run Ejemplo.py  
  
perro  
gato  
pájaro  
Delfin  
  
>>>
```

Arreglos, Listas, Tuplas, Diccionarios

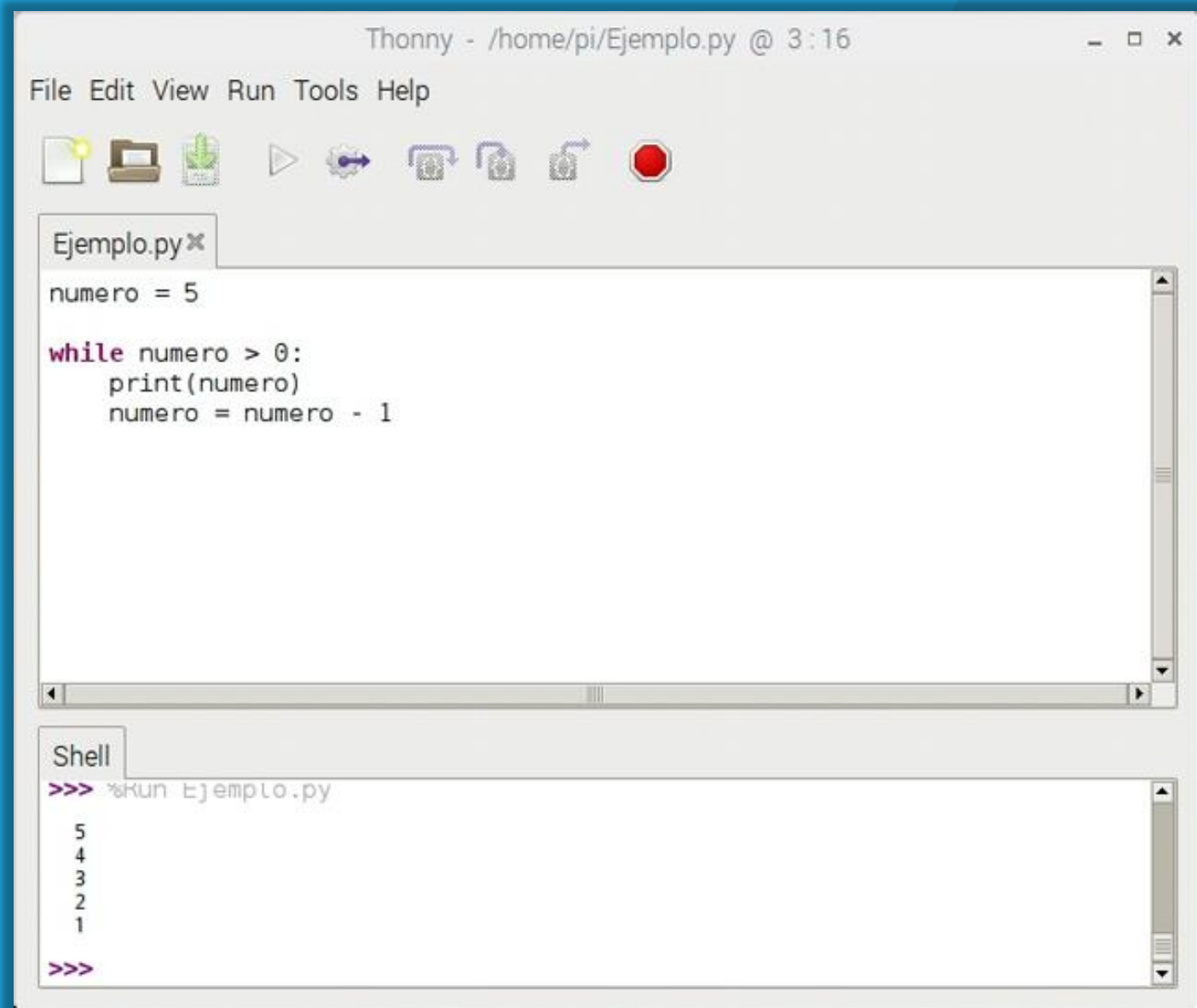


Arreglos, Listas, Tuplas, Diccionarios

This is an Array with Length=8



Ciclos: while



The image shows a screenshot of the Thonny Python IDE. The window title is "Thonny - /home/pi/Ejemplo.py @ 3:16". The menu bar includes "File", "Edit", "View", "Run", "Tools", and "Help". The toolbar contains icons for opening a file, saving, running, and other development tools. The main editor window shows a Python script named "Ejemplo.py" with the following code:

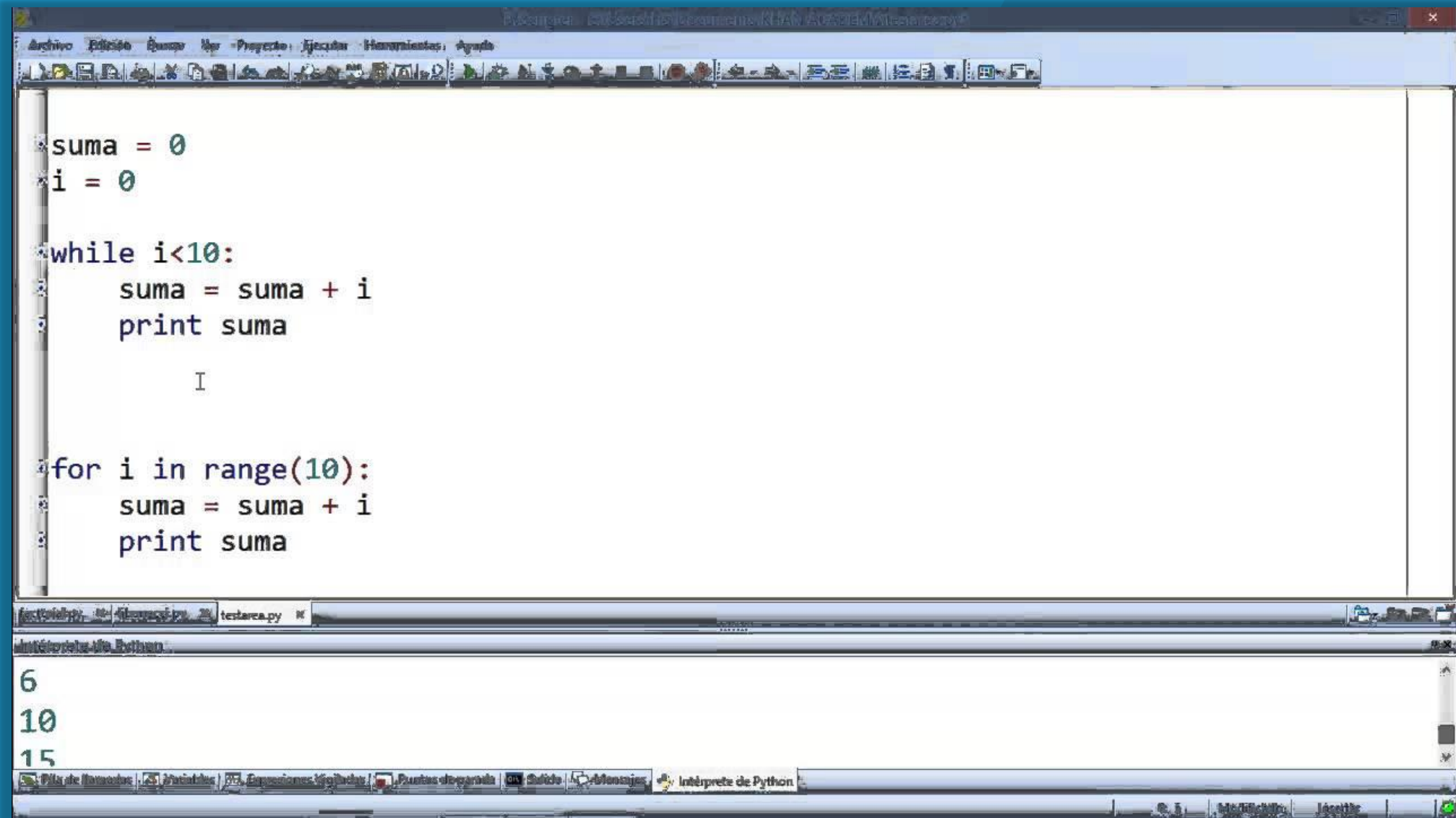
```
numero = 5

while numero > 0:
    print(numero)
    numero = numero - 1
```

Below the editor is a "Shell" window showing the execution output. The prompt is ">>> %run Ejemplo.py", and the output is:

```
5
4
3
2
1
>>>
```

Ciclos: while



The image shows a screenshot of a Python IDE window. The main editor area contains the following Python code:

```
suma = 0
i = 0

while i<10:
    suma = suma + i
    print suma
    I

for i in range(10):
    suma = suma + i
    print suma
```

Below the editor, there is a console window titled "Interprete de Python" which displays the output of the code execution:

```
6
10
15
```

The IDE interface includes a menu bar at the top with options like Archivo, Edición, Ejecución, and a toolbar with various icons. The status bar at the bottom shows the Python version as 2.5 and the file name as testarea.py.

Desarrollo de un Sistema de Cálculo de Presentismo

- Sueldo básico según el puesto de trabajo.
- Antigüedad.
- Asistencia y puntualidad.
- Liquidación feriados y no trabajados.
- Descuentos (jubilación, obra social, aportes, etc.).

Sueldo Básico

Escala Salarial 2019 Empleados de Comercio CCT 130/75					
CATEGORÍAS	BÁSICO ENERO 2019		BÁSICO FEBRERO 2019		BÁSICO MARZO 2019
Maestranza A	\$	24.862,98	\$	26.181,47	\$ 27.311,61
Maestranza B	\$	24.957,05	\$	26.280,53	\$ 27.414,94
Maestranza C	\$	25.286,73	\$	26.627,69	\$ 27.777,09
Administración A	\$	25.216,13	\$	26.553,35	\$ 27.699,54
Administración B	\$	25.357,51	\$	26.702,23	\$ 27.854,84
Administración C	\$	25.498,80	\$	26.850,01	\$ 28.010,05
Administración D	\$	25.922,72	\$	27.297,41	\$ 28.475,72
Administración E	\$	26.275,91	\$	27.669,33	\$ 28.863,69
Administración F	\$	26.794,03	\$	28.214,93	\$ 29.432,83
Cajeros A	\$	25.333,85	\$	26.677,31	\$ 27.828,85
Cajeros B	\$	25.498,79	\$	26.851,00	\$ 28.010,04
Cajeros C	\$	25.710,74	\$	27.074,19	\$ 28.242,86
Personal Auxiliar A	\$	25.333,85	\$	26.677,31	\$ 27.828,85
Personal Auxiliar B	\$	25.569,36	\$	26.925,31	\$ 28.087,56
Personal Auxiliar C	\$	26.346,57	\$	27.743,74	\$ 28.941,31
Auxiliar Especializado A	\$	25.616,59	\$	26.975,05	\$ 28.139,44
Auxiliar Especializado B	\$	26.040,43	\$	27.421,36	\$ 28.605,02

Cálculo de Presentismo

- **Antigüedad:** el adicional por antigüedad es el 1% del básico de convenio por cada año de antigüedad.
- **Asistencia y Puntualidad:** según lo establecido en artículo 40 del CCT 130/75 el adicional por Asistencia y Puntualidad es la DOCEAVA parte de las remuneraciones del mes (Básico + Antigüedad).
- **Liquidación Feriados trabajados y no trabajados:** a modo de ejemplo, en junio hay dos feriados. Si suponemos que un trabajador trabajó un feriado y el otro no, veamos como hacer la liquidación de un empleado con básico de \$27.699:
 - **Feriados : - \$2.200,57**
Primero, restamos los dos feriados en base 30:
 $(27.699,54 + 2.769,95 + 2.539,12) / 30 = 1.100,287$
Donde 1.100,287 es el valor de un día normal. En este caso como son dos días lo multiplicamos por 2 y queda en 2.220,574
 - **Pago feriado no Trabajado: \$ 1.320,34.**
Los feriados son días pagos, aún cuando no se haya trabajado ese día y, además, ese pago es con un plus, lo que se conoce como “plus por feriado”. Veamos como hacer el cálculo:
Calculamos el valor del día feriado, calculado en base 25:
 $(27.699,54 + 2.769,95 + 2.539,12) / 25 = 1.320,3444$
Donde 1.320,34 es el valor de un día feriado no trabajado.
 - **Pago feriado Trabajado: \$ 2.420,64**
Si el trabajador prestó servicios en día feriado, se deberá liquidar ese día como Feriado trabajado. Para ese caso, el artículo 166 de la LCT, en el segundo párrafo, establece la forma de cálculo para los trabajadores que se desempeñan los días feriados.
Cálculo del día feriado
 $(27.699,54 + 2.769,95 + 2.539,12) / 25 = 1.320,3444$
Cálculo del día normal
 $(27.699,54 + 2.769,95 + 2.539,12) / 30 = 1.100,287$
Ahora sumamos ambos valores y tenemos el total de un día feriado trabajado: $1320,34 + 1.100,28 = 2.420,62$

Cálculo de Presentismo

- **Descuentos**

- 11% de Jubilación
 - 3% de Obra Social
 - 3% según Ley 19.032
 - 2% de Aporte Solidario y OBLIGATORIO con destino al Sindicato de Empleados de Comercio. Esté el empleado afiliado o no al sindicato. (según el Art. 100 del CCT 130/75)
 - 0,5% de Aporte Solidario OBLIGATORIO con destino a FAECyS. Esté el empleado afiliado o no al sindicato. (según el Art. 100 del CCT 130/75)
 - 2% con destino al sindicato para empleados afiliados al gremio. Para este ejemplo consideré que no está afiliado por eso no se hizo el cálculo. Hay que tener en cuenta también que según la zona, este porcentaje puede cambiar.
-
- **Aporte OSECAC:** aporte de \$100 con destino a la O.S.E.C.A.C. Sólo para afiliado a esta obra social. El descuento no corresponde para los que tienen otra obra social o una pre-paga.

```
GA Command Prompt - python
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\yesi>python
Python 2.7.3 (default, Apr 10 2012, 23:31:26) [MSC v.1500 32 bit (Intel)] on win
32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Conexion pyqt4 y sqlite3 - conBase.ui

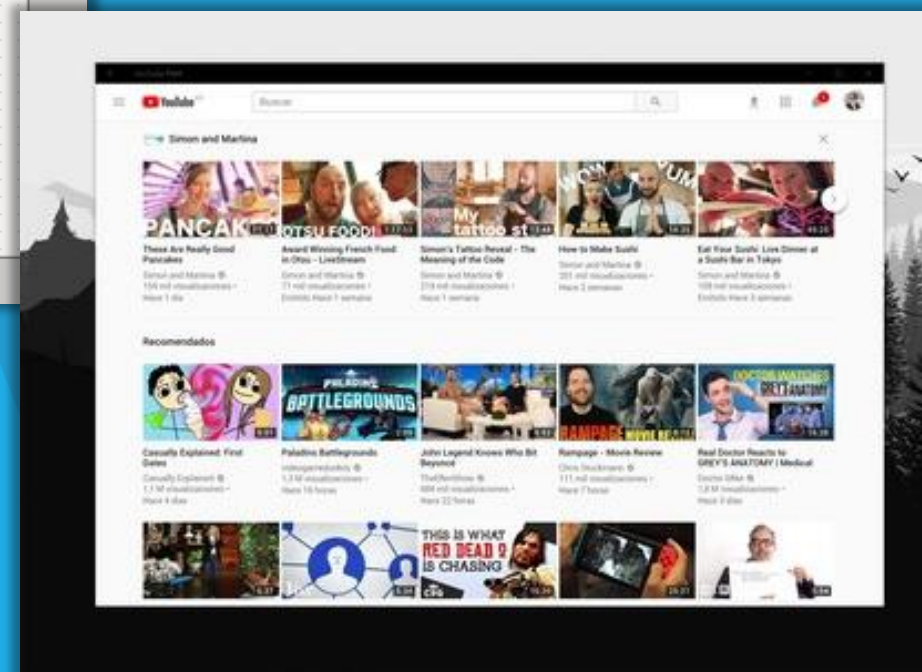
www.pythondiario.com

Cientes

Nombre

Apellido

Localidad



Python GUI Frameworks (librerías)



1 PYQT5

PyQt5 is a set of Python bindings for Qt5 application framework from Digia. It is available for the Python 2.x and 3.x. Qt library is one of the most powerful GUI libraries. The official home site for PyQt5 is www.riverbankcomputing.co.uk/. PyQt5 is developed by Riverbank Computing. PyQt5 is implemented as a set of Python modules. It has over 620 classes and functions and methods. It is a multiplatform toolkit which runs on all major operating systems, including Unix, Windows, and Mac OS. PyQt5 is dual licensed. Developers can choose between a GPL and a commercial license.



2 Tkinter

Tkinter is a Python binding to the Tk GUI toolkit. Tk is the original GUI library for the Tcl language. Tkinter is implemented as a Python wrapper around a complete Tcl interpreter embedded in the Python interpreter. It is a built-in module in Python, when you have installed Python, Tkinter will be installed by default.



4 Kivy

Kivy is an open source Python framework for creating cross-platform mobile applications with Natural User Interface. Its speed is comparable to mobile alternatives like Java for Android and Objective C for iOS. Kivy is a solution for coding in Python on mobile devices. It is also capable of running on multiple platforms like HTML5 because it does not depend on heavy browser and is implemented in C using Cython due to which it runs directly on the hardware. The main aim that was kept in mind while developing Kivy was to run the same code on multiple platforms.



5 PyForms

At just two years old (making it one of the more recent frameworks), **Pyforms** is a Python 2.7/3.x cross-environment framework for developing GUI applications. It is modular and encourages code reusability with minimal effort. Based on PyQt, OpenGL and other libraries, it provides a quite comprehensive set of 22 controls, all based on a ControlBase object; it also includes a video player, web browser and OpenGL.

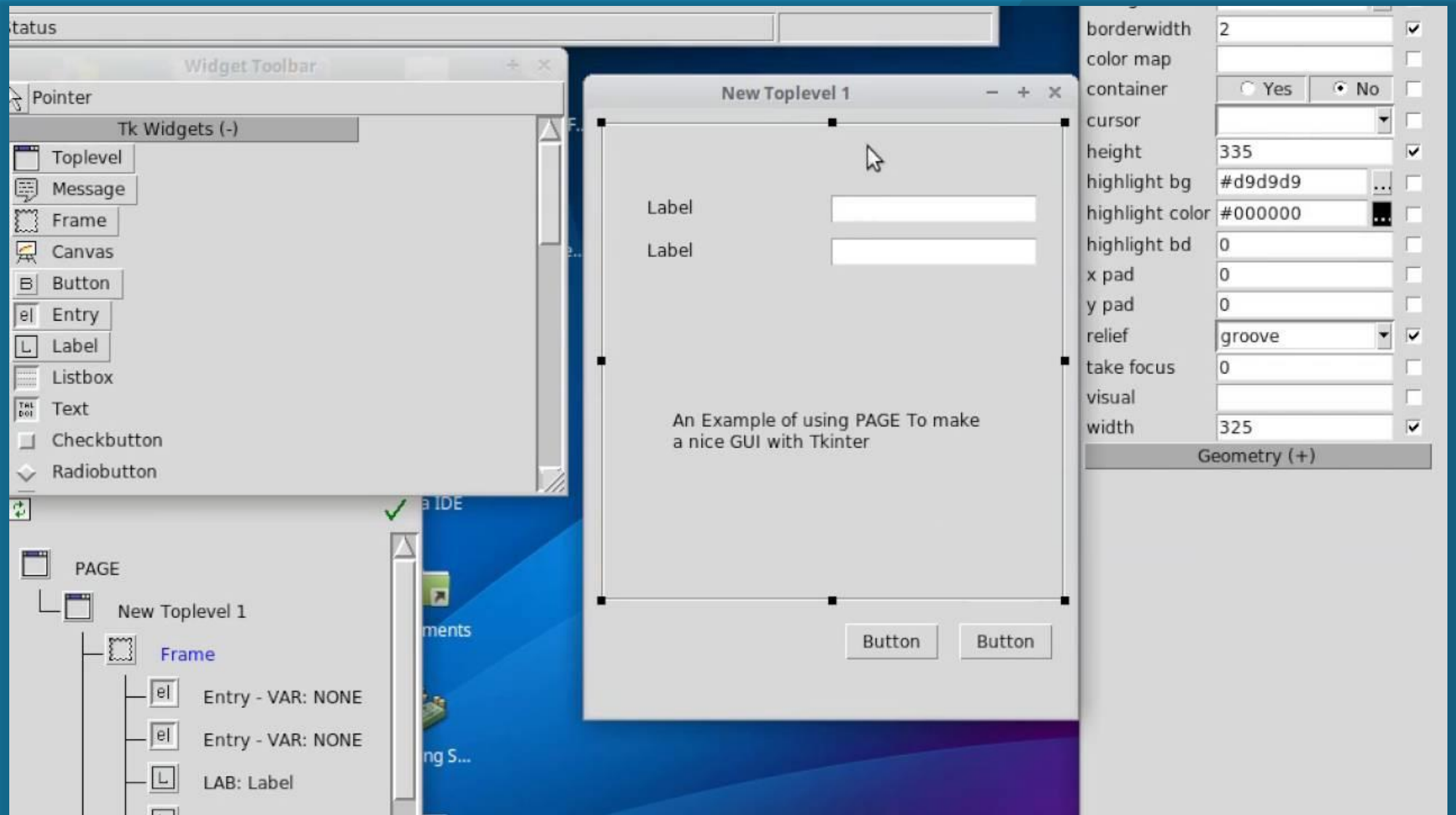


3 Wxpython

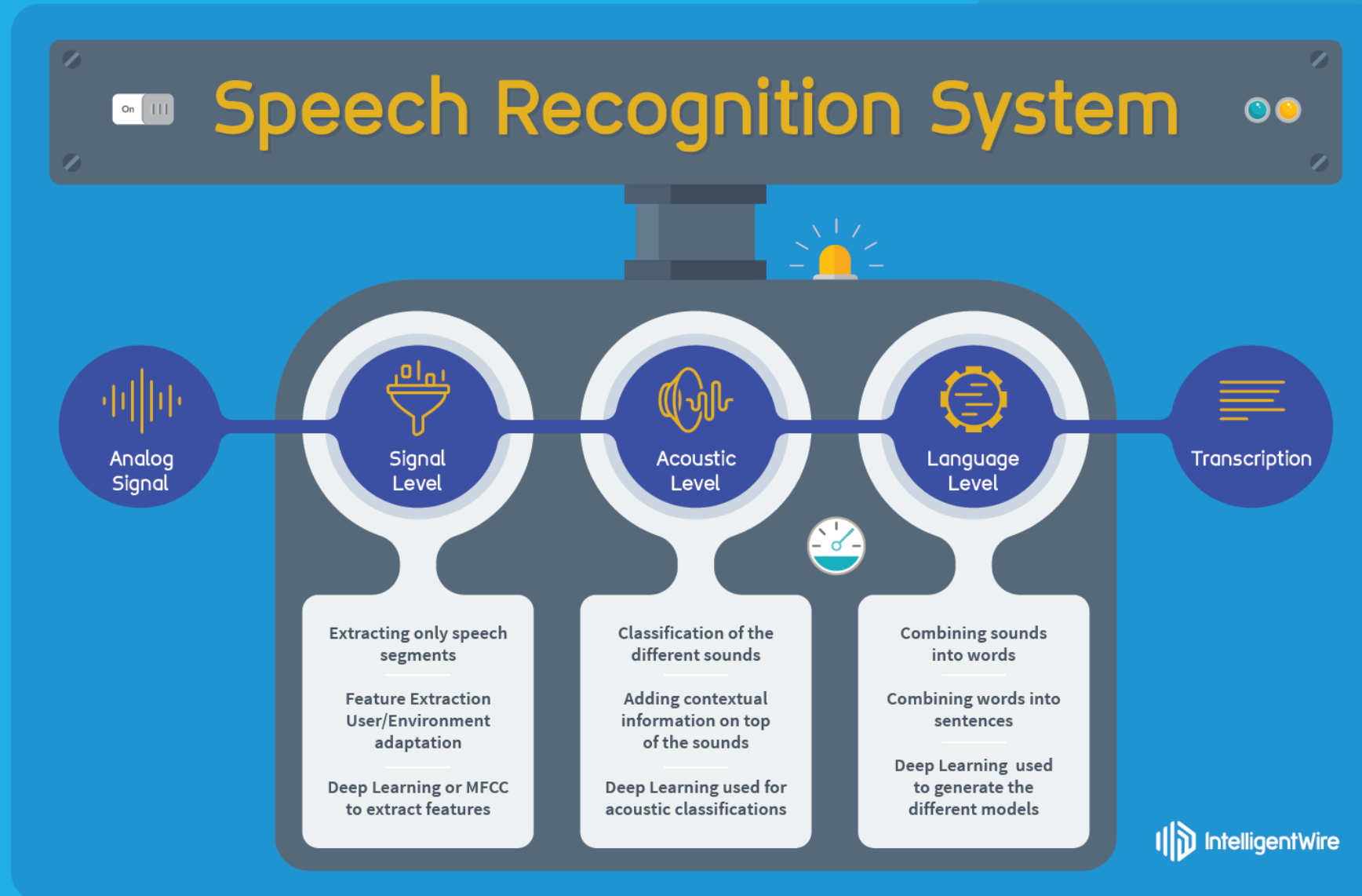
WxPython is a Python wrapper for **wxWidgets** (which is written in C++), a popular cross-platform GUI toolkit. Developed by Robin Dunn along with Harri Pasanen, WxPython is implemented as a Python extension module. WxWidgets, WxPython is also a free software. It can be downloaded from the official website <http://wxpython.org>. Binaries and source code for many operating system platforms are available for download on this site. The modules in WxPython API include a core module. It consists of **wxObject** class, which is the base class for all classes in the API. Control module contains all the widgets used in GUI application development. For example, **wx.Button**, **wx.StaticText** (analogous to a label), **wx.TextCtrl** (editable text control), etc. WxPython API has GDI (Graphics Device Interface) module. It is a set of classes used for drawing on the screen. Classes like font, color, brush, etc. are a part of it. All the container window classes are part of the Windows module.

Una Ventana con un Botón en Python

```
# Las dos líneas siguientes son necesarias para hacer  
# compatible el interfaz Tkinter con los programas basados  
# en versiones anteriores a la 8.5, con las más recientes.  
from tkinter import * # Carga módulo tk (widgets estándar)  
from tkinter import ttk # Carga ttk (para widgets nuevos 8.5+)  
  
# Define la ventana principal de la aplicación  
raiz = Tk()  
  
# Define las dimensiones de la ventana, que se ubicará en  
# el centro de la pantalla. Si se omite esta línea la  
# ventana se adaptará a los widgets que se coloquen en ella.  
raiz.geometry('300x200')  
  
# Asigna un color de fondo a la ventana. Si se omite  
# esta línea el fondo será gris  
raiz.configure(bg = 'beige')  
  
#Asigna un título a la ventana  
raiz.title('Aplicación')  
  
# Define un botón en la parte inferior de la ventana  
# que cuando sea presionado hará que termine el programa.  
# El primer parámetro indica el nombre de la ventana 'raiz'  
# donde se ubicará el botón  
ttk.Button(raiz, text='Salir', command=quit).pack(side=BOTTOM)  
  
# Después de definir la ventana principal y un widget botón  
# la siguiente línea hará que cuando se ejecute el programa  
# construya y muestre la ventana, quedando a la espera de  
# que alguna persona interactúe con ella.  
  
# Si la persona presiona sobre el botón Cerrar 'X', o bien,  
# sobre el botón 'Salir' el programa llegará a su fin.  
  
raiz.mainloop()
```

Asistente por Voz, proceso de reconocimiento de voz:

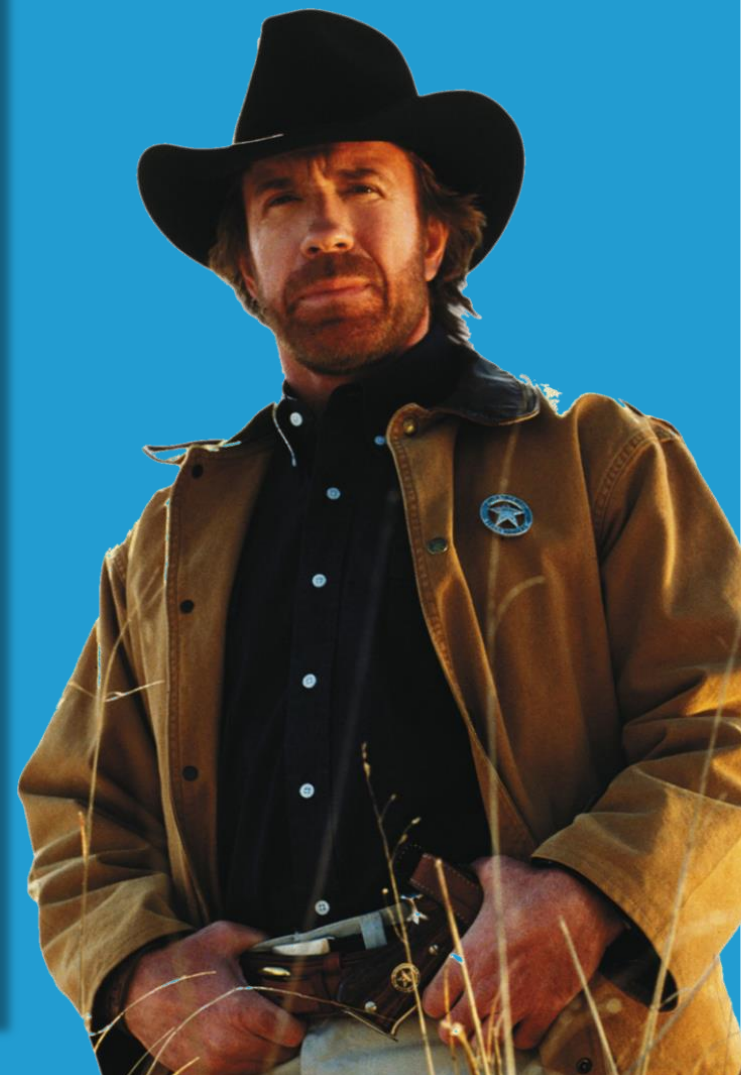


Asistente por Voz, proceso de reconocimiento de voz:

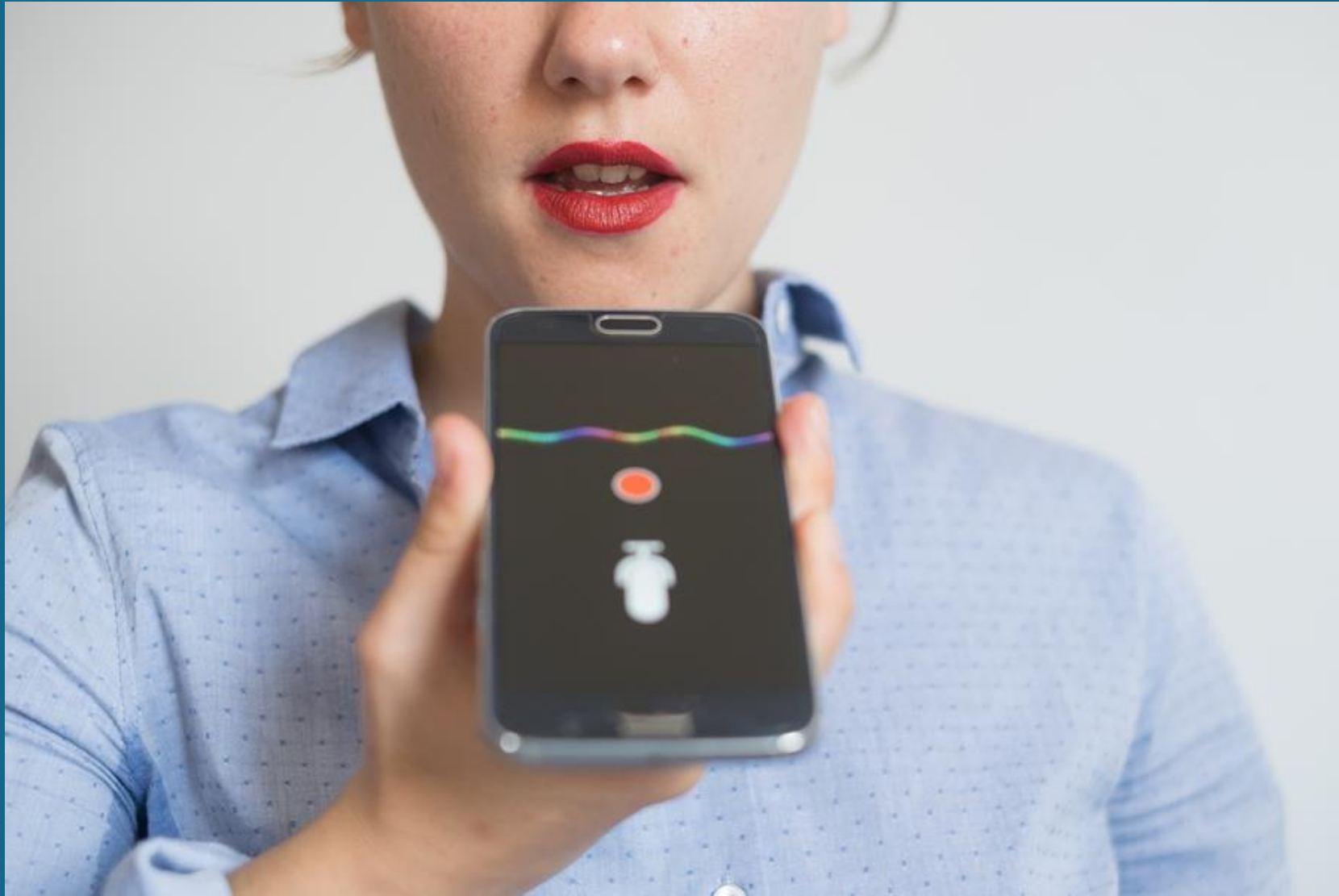
- [apiai](#)
- [assemblyai](#)
- [google-cloud-speech](#)
- [pocketsphinx](#)
- [SpeechRecognition](#)
- [watson-developer-cloud](#)
- [wit](#)

Let's code!

Concepto	Unidades	Remunerativo		Descuentos
Básico	30	27.699,54		
Antigüedad	10	2.769,95		
Asistencia y puntualidad		2.539,12		
Día Feriado	2	-2.200,57		
Pago día Feriado no trabajado	1	1.320,34		
Pago día Feriado trabajado	1	2.420,63		
Básico			1.246,48	
Antigüedad			124,65	
Asistencia y puntualidad			114,26	
Día Feriado	2		-99,03	
Pago día Feriado no trabajado	1		59,42	
Pago día Feriado trabajado	1		108,93	
Jubilación	11,0%			3.800,39
Ley 19.032	3,0%			1.036,47
Obra Social	3,0%			1.083,11
S.E.C. Art 100 CCT 130/75	2,0%			722,07
F.A.E.C.y.S. Art 100 CCT 130/75	0,5%			180,52
Aporte Fijo OSECAC s/Acuerdo 2019				100,00
www.ignacioonline.com.ar			1.554,71	6.922,57
			A pagar	\$ 29.181,16



Let's code!



TAREA PARA LA CASA

- Instalar Tkinter / PAGE y escribir y probar códigos de clase.
- Desarrollar un asistente de voz que permita ingresar un texto en español y traducir el mismo a inglés.
- Desarrollar un asistente de voz que permita ingresar un texto en inglés y traducir el mismo a español.



Más información

<http://www.aetti.org/hub>

<https://www.facebook.com/aettitucuman>

AETTI HUB
Córdoba 191 - SM Tucumán

DATOS DE CONTACTO
Omar Rivas
omar@planetabots.com



AETTI
HUB