

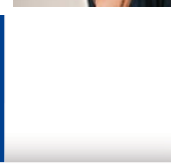
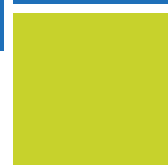
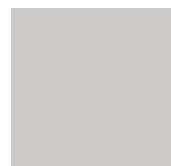
MÓDULO

1

Área: **NEGOCIOS**

Curso: **PROGRAMACIÓN BÁSICA (PYTHON)**

Módulo: **Comprensión de problemas e inmersión al lenguaje de programación**



IPP

SUEÑA • APRENDE • CRECE

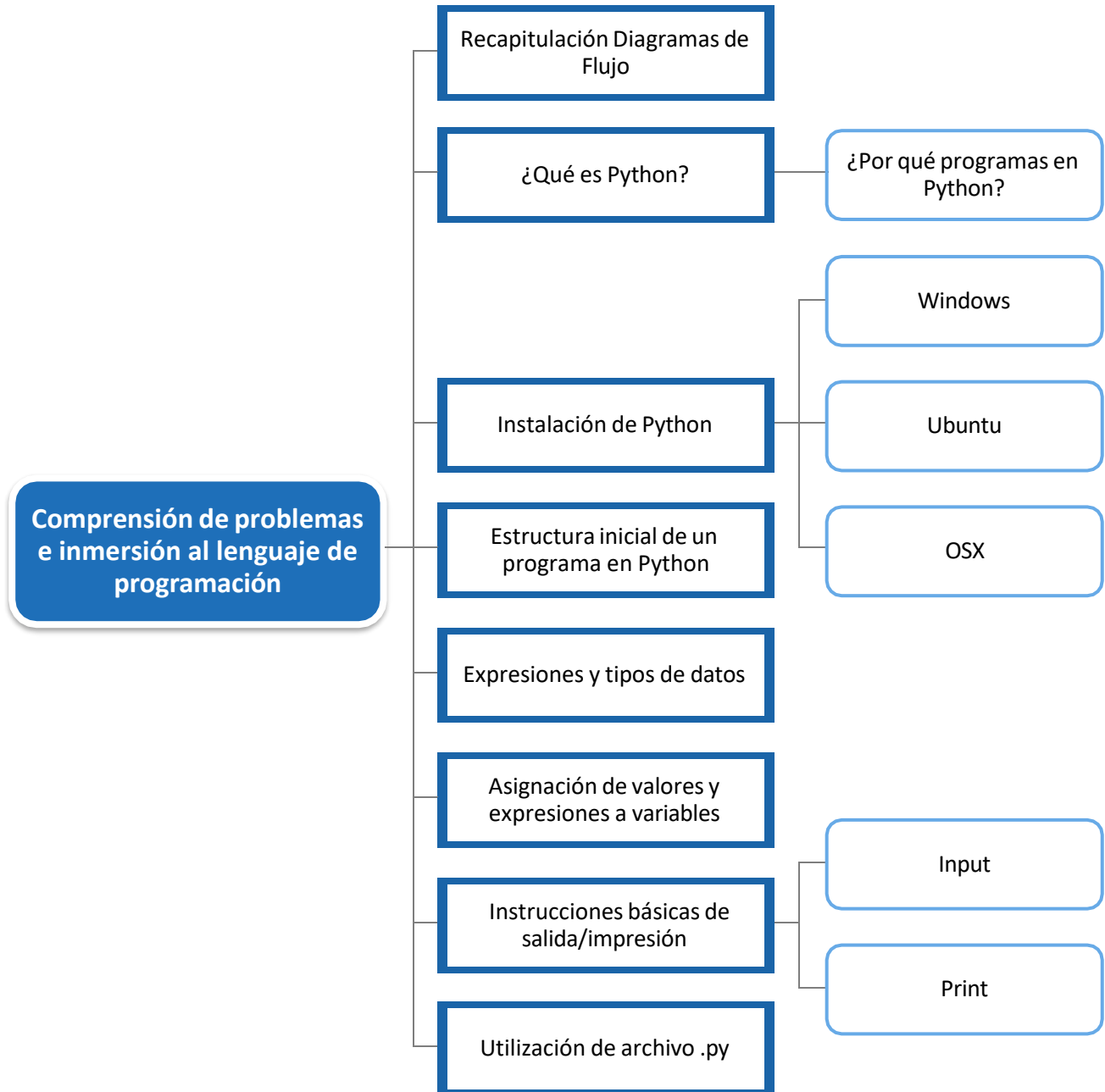


Índice

Introducción	1
1. Recapitulación DDF de Arquitectura de Computadores.....	1
2. ¿Qué es Python?	4
2.2.¿Por qué programar en Python?	4
3. Instalación de Python3	6
3.1.En Windows	6
3.2.En Ubuntu (Linux).....	7
3.3.En OSX (MAC)	8
4. Estructura inicial de un programa en Python.....	9
5. Expresiones y tipos de datos básicos.....	11
6. Asignación de valores y expresiones a variables.....	12
7. Instrucciones básicas de salida/impresión	14
7.1.Comando input.....	14
7.2.Comando print	16
8. Utilización de archivo .py	17
9. Cierre.....	19
10. Glosario modular	20



Mapa de Contenido



RESULTADO DE APRENDIZAJE DEL MÓDULO

Crea un programa computacional simple que responde a un enunciado previamente conocido, a partir de la identificación de las partes fundamentales del problema, separando datos de entrada y transformaciones necesarias para producir los resultados de salida esperados.

Introducción

En la actualidad, todo lo que nos rodea está automatizado. Abundan los conceptos de inteligencia artificial, programación, sistemas y tecnología, entre otros. Pero ¿cómo funciona esta nueva tecnología, que cada vez está más cercana y de bajo costo?

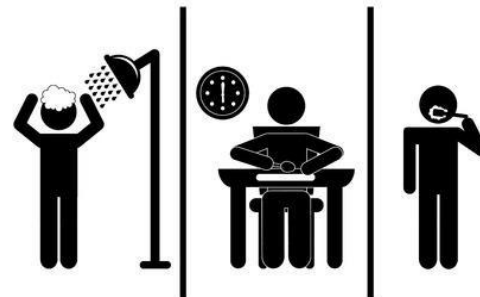
Todos los sistemas, programas e incluso la inteligencia artificial, están diseñados con secuencias lógicas, que normalmente pensamos o realizamos. Estas secuencias programadas se unen para formar un gran sistema, como el sistema de pago Redcompra, que es capaz de soportar millones de compras diarias. En el área de vigilancia, una imagen de un rostro puede proporcionar información completa sobre una persona mediante una computadora.

Esto es solo el comienzo. La mayoría de las personas que ahora manejan estos súper sistemas empezaron con algoritmos simples, como un básico "Hola mundo" en un programa. Este es básicamente lo que aprenderán en este curso, comenzando por un "Hola mundo IPP" y terminando con la creación de programas simples en Python

1. Recapitulación DDF de Arquitectura de Computadores

Antes de realizar un programa, independientemente del lenguaje que se utilice, es importante tener descompuestas las distintas actividades, las cuales pueden ser fácilmente ejemplificadas en la vida cotidiana, como, por ejemplo: "Levantarse para ir al trabajo". Para ello, debemos hacer lo siguiente:

1. Me despierto
2. Me levanto
3. Me ducho
4. Me Seco
5. Me Visto
6. Tomo Desayuno
7. Camino hacia el paradero de locomoción Colectiva
8. Tomo la locomoción
9. Me bajo en mi destino
10. Camino hacia el trabajo
11. Saludo al recepcionista



Al ver este simple ejemplo, uno se da cuenta que todo tipo de acción puede descomponerse fácilmente en diferentes acciones, en donde si no podemos realizar cualquier de ellas, no podemos realizar las siguientes.

Las acciones enumeradas anteriormente son conocidas como "Tareas Unitarias". Una tarea unitaria se define porque tiene un objetivo simple y solo requiere una acción, como por ejemplo "Despertarse". La única tarea unitaria que importa en ese momento es despertarse, que consiste en estimular el cerebro para que la persona pueda tomar conciencia de sus actos, abrir los ojos y mover el cuerpo. A pesar de la explicación detallada de cómo despertamos, nos damos cuenta de que lo único importante en ese momento y el único objetivo es despertarse, y esto se conoce como tarea unitaria.



Otro sencillo ejemplo podría ser el "Cambiar la rueda de un automóvil". Primero tenemos que pensar en qué necesitamos para poder cambiar el neumático.:

- Un neumático nuevo
- Una Gata
- Llave Cruz

En la programación ocurre lo mismo, en este caso el neumático nuevo y la gata, serían mis variables o recursos para utilizar.

Paso 1	<ul style="list-style-type: none"> El siguiente paso es colocar la gata debajo del automóvil, en el lado en el que se encuentra la rueda defectuosa y cerca de ella. Una vez posicionada la gata, comenzamos a levantar el auto girando la palanca de la gata hasta que la rueda defectuosa ya no toque más el suelo. En programación eso se podría definir de la siguiente manera: tenemos que hacer girar la gata "mientras" la rueda defectuosa toque el suelo. Al poner esta instrucción, dejaremos de girar la gata cuando nos percatemos de que ésta no toca el piso.
Paso 2	<ul style="list-style-type: none"> Como tenemos el conocimiento de que un neumático tiene 4 pernos que lo sostienen, lo que tenemos que hacer es sacar cada perno y esto lo haremos con la llave cruz. En este caso, como sabemos la cantidad de pernos que existen, podremos realizar un ciclo "Para". El cual dirá "Para cada perno en la rueda (1 a 4), lo iremos sacando con la llave cruz"
Paso 3	<ul style="list-style-type: none"> Posterior a sacar todos los pernos, procedemos a retirar el neumático defectuoso y a colocar el neumático nuevo. Ahora para colocar el neumático nuevo, debemos colocar nuevamente los pernos. Entonces, realizaremos de nuevo el paso de "Para cada perno de 1 a 4, lo iremos colocando" Posterior a todo esto, tenemos que bajar la gata. En este caso sería "Mientras la gata no esté completamente junta, seguir girando". Una vez que la gata esté totalmente junta, la sacamos y tenemos nuestro automóvil listo con un nuevo neumático listo para su uso."

Toda la actividad anterior, puede ser graficada en un Diagrama De Flujo, la cual nos modela cada una de las acciones descritas, estableciendo las relaciones entre cada una de ellas.

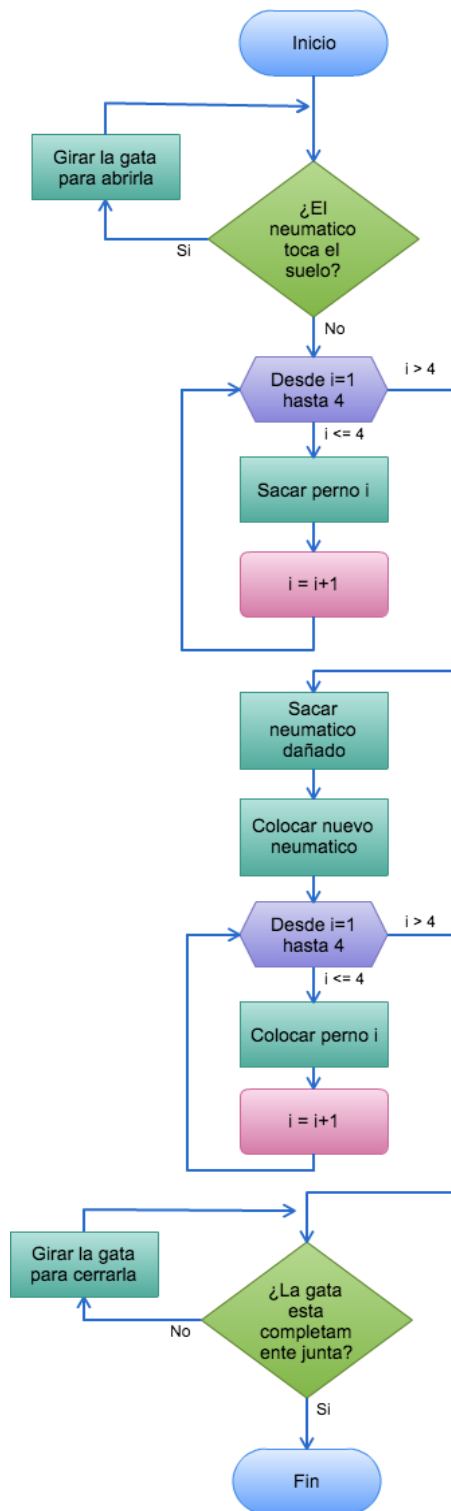


Figura 1 - Diagrama de "Cambiar la rueda de un automóvil"

SABER MÁS

Más ejemplos en los siguientes textos:

- Noguera O., Riera, F. (2013). Programación. Capítulo 4 páginas 23 a 27.
- Bores, M. (1993). Computación: Metodología, lógica computacional y programación. Páginas 11 a 17.

2. ¿Qué es Python?

Python es un lenguaje de programación creado a finales de los años 80 por el científico de la computación Guido van Rossum de Holanda. Su característica principal es tener una sintaxis más clara que otros lenguajes de programación y una gran similitud con el pseudocódigo, lo que lo hace accesible para principiantes. Algunas características únicas de Python son:

1. **Lenguaje interpretado:** se ejecuta a través de un programa llamado intérprete, en lugar de un lenguaje de máquina como ocurre con los lenguajes compilados.
2. **Tipado dinámico:** no es necesario declarar el tipo de dato de una variable, ya que se ajusta según el tipo asignado en tiempo de ejecución.
3. **Fuertemente tipado:** no se pueden combinar variables de distintos tipos.
4. **Multiplataforma:** el intérprete está disponible en varias plataformas.

2.2. ¿Por qué programar en Python?

De acuerdo con el ranking de TIOBE de agosto de 2022, Python se encuentra en el primer lugar como el lenguaje de programación más utilizado en el mundo. Este éxito se debe a su facilidad de uso y a sus características descritas anteriormente. Python es un lenguaje que ha mantenido su popularidad con el tiempo, a diferencia de Java y C, que han perdido importancia con el paso de los años, aunque aún mantienen su posición

Esta lista se compone de:

 1er lugar: Python	<p>Python es una gran herramienta de programación que utilizan las grandes empresas, incluso algunas que no están vinculadas al área de programación. Por ejemplo, en Walt Disney Animation Studios (WDA), utilizan este lenguaje tanto en herramientas administrativas, gestión y de desarrollo gráfico de sus productos.</p>	
 2do lugar: C	<p>Es un lenguaje que puede trabajar en bajo y alto nivel, ocupado bastante en la robótica y la automatización.</p>	
 3er lugar: JAVA	<p>Es uno de los primeros lenguajes multiplataforma que se crearon. Fue diseñado específicamente para tener tan pocas dependencias de implementación como fuera posible, su intención es permitir que los desarrolladores de aplicaciones escriban el programa una vez y lo ejecuten en cualquier dispositivo. A partir de 2012 es uno de los lenguajes de programación más populares en uso, particularmente para aplicaciones de cliente-servidor de web, con unos 10 millones de usuarios reportados.</p>	
 4to lugar: C ++	<p>Es la mejora realizada a C, en este se puede trabajar con objetos y trabaja en mayor alto nivel que C.</p>	
 5to lugar: C#	<p>Lenguaje de programación orientado a objetos desarrollado y estandarizado por Microsoft como parte de su plataforma .NET. Su sintaxis básica deriva de C/C++ y utiliza el modelo de objetos similar al de Java, aunque incluye mejoras derivadas de otros lenguajes.</p>	

SABER MÁS

Más información sobre este tema en: <https://www.tiobe.com/tiobe-index/>

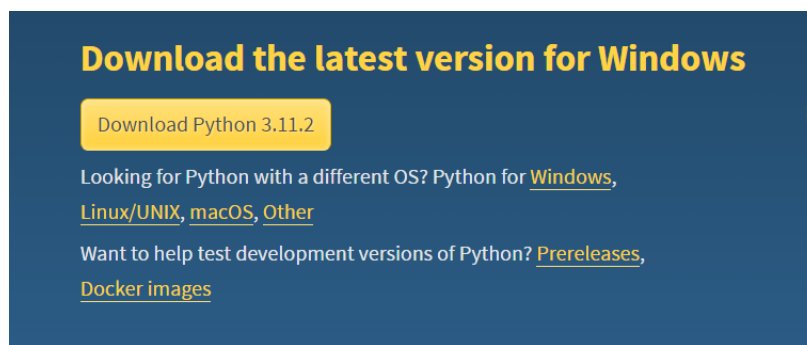
3. Instalación de Python3

A continuación, detallaremos los pasos para instalar **Python versión 3.11.2** en distintos sistemas operativos, lo cual permita que utilicen Python desde un nivel inicial básico.

3.1. En Windows

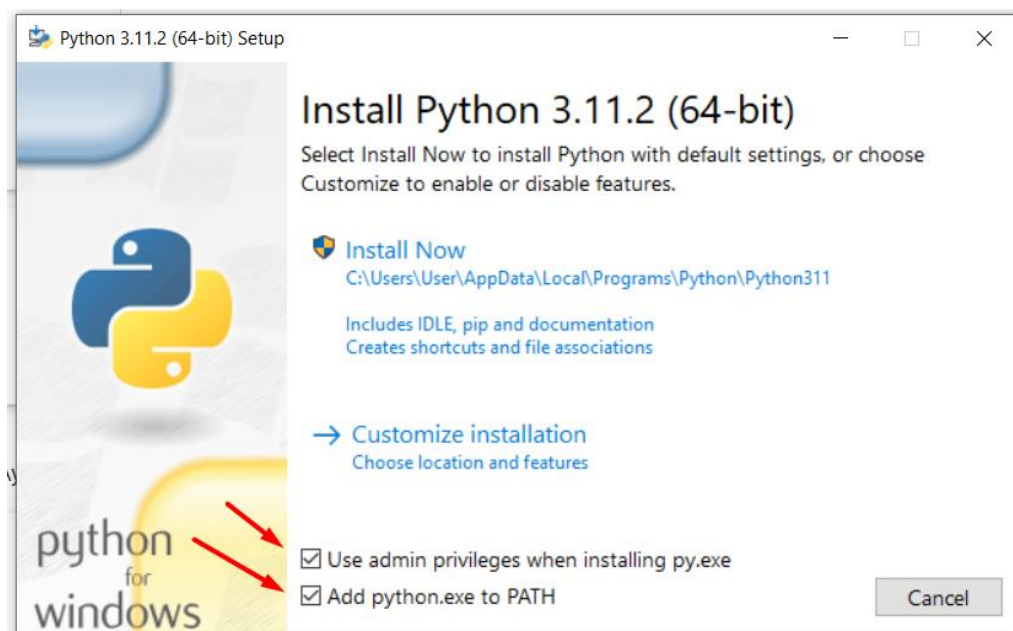
Los pasos que se presentarán están pensados para **Windows 10**.

1. Ingresa en el siguiente sitio para descargar la última versión: <https://www.python.org/downloads/>

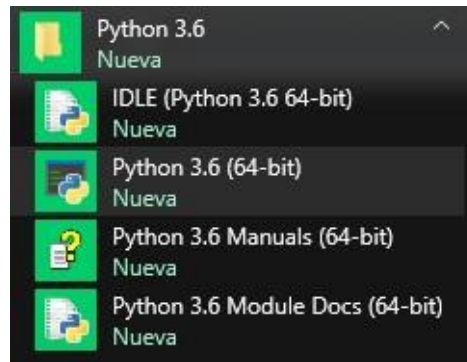


En el desarrollo de este curso la última versión en febrero de 2023 es la 3.11.2

2. Se selecciona el archivo ejecutable, para esta versión sería *Windows x86-64 executable installer* (también se puede descargar para una sola arquitectura, x86 o x64, pero si no conoces bien la arquitectura de tu sistema operativo, ocupa x86-64)
3. Es importante marcar la alternativa que dice **ADD Python to PATH**



4. Para seguir, hacer clic en Install Now. Posterior se iniciará la instalación, cuando esta concluya solo cierre el instalador. Posterior a la instalación tendremos los siguientes programas en nuestro equipo.



Para comprobar que tenemos una instalación exitosa, realizaremos un **Hola Mundo**.

```
Python 3.11 (64-bit)
Python 3.11.2 (tags/v3.11.2:878ead1, Feb  7 2023, 16:38:35) [MSC v.1934 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> print("hola mundo")
hola mundo
>>> _
```

Una vez que se ejecute esta operación sin problema, significa que tenemos nuestro Python 3 listo para trabajar.

3.2. En Ubuntu (Linux)

En Ubuntu es mucho más sencillo dependiendo de versión que se tenga, en unos casos basta solo con tener que actualizar el repositorio y el sistema:

- Primero abrimos la consola, aplicaciones -> Consola o Control + t:
 - `sudo apt-get update`
 - `sudo apt-get -y upgrade`

- Se le recomienda también ejecutar el siguiente comando para no tener problemas a futuro:
 - `sudo apt-get install build-essential libssl-dev libffi-dev python-dev`
 - `sudo apt-get install -y python3-pip`
- Otra manera es directamente colocando el nombre del paquete Python:
 - `sudo apt-get install python3`
 - `sudo apt-get install build-essential libssl-dev libffi-dev python-dev`
 - `sudo apt-get install -y python3-pip`

Para comprobar que todo está bien, en consola se escribe `python3` (apretamos Enter) y escribimos el siguiente comando `print('Hola Mundo IPP')`, deberían ver los siguiente:

```
Python 3.4.3 (default, Nov 17 2016, 01:08:31)
[GCC 4.8.4] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> print('Hola Mundo IPP')
Hola Mundo IPP
>>> █
```

Una vez que se ejecute esta operación sin problema, significa que tenemos nuestro Python 3 listo para trabajar.

3.3. En OSX (MAC)



Primero se les recomienda instalar el Xcode que se encuentra en app store, una vez realizado esto tendrán Python listo en sus computadores pero con la versión 2.7. En este curso se trabajará con Python3 así que tenemos que actualizar esta versión. Para realizarlo debemos entrar a la página web de Python (<https://www.python.org>), hacer clic en descarga y selecciona la versión 3.11.2 (la más actual hasta el 10-02-2023).

IMPORTANTE

Encuentra en el Plan de Estudio el instructivo para instalar **Xcode**.

Cuando realicen clic se les descargará un archivo pkg., luego realizan doble clic y aceptan el contrato. Posterior a esto se les instalará Python3.

Para comprobar que tenemos todo bien instalado abrimos la consola de **OSX (Aplicaciones -> otras -> terminal)** y escribimos `python3` (presionamos Enter). Una vez que aparezcan `>>>` significa que estamos dentro de Python3. Para probar si está todo bien hay que realizar `print('Hola Mundo IPP')`, mostrando el siguiente mensaje.

```
[GCC 4.2.1 Compatible Apple LLVM 8.0.0 (clang-800.0.34)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> print('Hola Mundo IPP')
Hola Mundo IPP
>>> |
```

Una vez que se ejecute esta operación sin problema, significa que tenemos nuestro Python 3 listo para trabajar.

IMPORTANTE

Encuentra en el Plan de Estudio el instructivo para instalar **Thonny**. El cual te orientará en la primera experiencia de programación en Python.

4. Estructura inicial de un programa en PYTHON

Python es un lenguaje de programación interpretado, es decir que la máquina comienza a leer línea por línea. Si esta tiene algún problema o error en alguna línea, el programa dejará inmediatamente de ejecutarse mostrando el error. Esta diferencia se genera con los lenguajes de programación compilador, como C, C++, entre otros, en los cuales el compilador verifica todo el código antes de ejecutarse.

La estructura que se puede generar en Python es la siguiente:

1. **Línea de inicialización:** Esta se genera principalmente en los sistemas operativos con base Unix, como Linux y OS X. La cual, por defecto, es: `#!/usr/bin/env Python`
2. **Módulos que se importan:** También se pueden importar librerías, las cuales aprenderemos a instalar durante el curso. Aquí ocuparemos una librería que por defecto se instala con Python, esta es **sys**, la cual permite ejecutar programas **bash** de nuestro computador:

```
import sys
```

3. **Declaración de Variables globales:** Las variables que no se encuentran dentro de ninguna clase o función son llamadas **Variables Globales**, las cuales pueden ser usadas y modificadas en cualquier clase o función del programa. Como ejemplo, declararemos una variable "a" con el valor de **15**:

```
a = 15
```

4. **Declaración de Clases:** Las clases son un conjunto de operaciones y funciones con un objetivo determinado (las cuales aprenderemos a utilizar más adelante en el curso). La estructura de una clase es la siguiente:

```
class Clase1(Object):  
  
"Objetos de la clase"
```

5. **Declaración de Funciones:** Es una agrupación de acciones específicas, como, por ejemplo, tomar 2 números y sumarlos. La estructura de una función es la siguiente:

```
def funcion1(variables):  
"operaciones de la función"
```

6. **Cuerpo Principal:** El cuerpo principal del programa es el main, es aquí donde el programa comienza a realizar acciones usando variables, funciones o clases:

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

Todo lo anterior mencionado se vería de la siguiente manera:

```
1  #/usr/bin/env python          #Linea de inicialización  
2  
3  import sys                    #Modulos que se importan  
4  
5  a = 15                        #Declaracion de variables Globales  
6  
7  class Clase1(Object):  
8      "Objetos de la clase"     #Declaracion de Clase  
9  
10 def funcion1():  
11     "operaciones de la funcion" #Declaracion de Funciones  
12  
13 if __name__ == '__main__':    #Cuerpo Principal  
14     funcion1()
```

Al iniciar este curso no ocuparemos ninguna estructura, para que puedan obtener un conocimiento de manera más sencilla y posteriormente aprendan con mayor facilidad el manejo de esta estructura y algunos problemas que podrían surgir al mal uso de estas.

5. Expresiones y tipos de datos básicos

En Python puedes tener diversos tipos de datos, los cuales pueden ser número naturales, números enteros de gran tamaño o de poco tamaño, letras y texto (conocido como cadena de caracteres).

Para declarar estas variables en Python es muy sencillo, solo tienes que igualar el nombre de la variable que le quieres dar, por ejemplo:

a = 15

En este caso Python interpreta directamente que la **variable a** vale **15** y es un numero natural.

a= 15.3

En este caso Python interpreta que **a es una variable de tipo entero**, con un numero con decimales.

a= 'p'

En este caso, Python interpreta que es un string (cadena de caracteres), al igual que si escribiéramos `a='pepe'`. Todas las letras o cadena de texto tienen que ir entre comilla simple o dobles, pero recuerden siempre ser concisos, **si se empieza con comilla simple terminar con comilla simple, es lo mismo para la comilla doble.**

a=True

Este tipo de dato se llama boolean, el cual solo puede ser verdadero o falso (True o False), para declarar esta variable siempre deber poner la primera letra en mayúscula.

Python tiene un comando para poder saber qué tipo de dato fue el que interpreto, este comando se llama **type()**.

Un ejemplo:

a=15
type(a)

Python retornará que es un entero, el cual se define como **int** (la cual es abreviación de integer, entero en inglés).

En esta imagen se ven escritos los ejemplos anteriores:

```
~ » python3
Python 3.4.3 (default, Nov 17 2016, 01:08:31)
[GCC 4.8.4] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> a = 15
>>> type(a)
<class 'int'>
>>> a = 15.4
>>> type(a)
<class 'float'>
>>> a = 'p'
>>> type(a)
<class 'str'>
>>> a = 'pepe'
>>> type(a)
<class 'str'>
>>> a = True
>>> type(a)
<class 'bool'>
>>> 
```

6. Asignación de valores y expresiones a variables

Con Python se puede fácilmente realizar operaciones matemáticas (sumar, restar, multiplicar y dividir), como, por ejemplo, declaramos dos variables: **a** y **b**:

```
a = 10
b = 15
c = a + b (resultado c=25)
c = a - b (resultado c=-5)
c = a * b (resultado c=150)
c = a / b (resultado c=0,66666667)
```

También podemos colocar ecuaciones con variables. Como saber la velocidad final de un automóvil con una aceleración constante en un tiempo determinado.

Siendo la fórmula matemática la siguiente:

$$v_f = v_0 + a * t$$

v_f = velocidad final
 v_0 = velocidad inicial
 a = aceleración
 t = tiempo

Entonces si tenemos que la velocidad inicial es 16 m/ s, la aceleración 2 m/s² y el tiempo es 20 s.

Entonces en Python pondríamos lo siguiente:

```
vi = 16  
a = 2  
t = 20
```

Entonces
 $v_f = v_i + a * t$

Siendo $v_f = 56$

En el computador, mostrando todo lo explicado, quedaría de la siguiente manera:

```
>>> a = 10  
>>> b = 15  
>>> a+b  
25  
>>> a-b  
-5  
>>> a*b  
150  
>>> a/b  
0.6666666666666666  
>>> vi = 16  
>>> a = 2  
>>> t = 20  
>>> vf = vi + a*t  
>>> vf  
56  
>>> █
```


7. Instrucciones básicas de salida/impresión

Python tiene la facilidad de obtener datos desde teclado y mostrar por consola lo que nosotros queramos, esto se puede hacer con los siguientes comandos.

input() = Para obtener datos desde el teclado

print() = mostrar datos en consola

7.1. Comando Input

Entonces para poder obtener datos, se puede hacer de dos maneras:

1.

```
a = input()
```

#al hacer esto el computador se quedará esperando a que el usuario escriba algo y aprete *enter*.

2.

La otra manera es ocupar input, pero mostrándola un mensaje al usuario

```
a = input('escriba su número favorito')
```

Esto mostrara por pantalla el mensaje y se quedara esperando a que el usuario escriba algo y presione *enter*.

Lo que el usuario digite quedara guardado en la **variable a**. Queda de la siguiente forma:

```
>>> a = input()
15
>>> a
'15'
>>> a = input('escriba su numero favorito')
escriba su numero favorito?
>>> a
'7'
```

Como se darán cuenta en el **input con mensaje**, nos dejó escribir el número apegado al mensaje, pero esto no es muy agradable para el usuario. Por ello al final del mensaje debemos agregar un `\n`, lo que significa un salto de línea.

```
>>> a = input('escriba su numero favorito \n')
escriba su numero favorito
7
>>> a
'7'
```

Siempre que hacemos un input, lo que guardemos queda como texto, si ejecutamos un **type()** de **a** nos retornará un **str**, siendo que es un número. Al ser así, no podremos sumar nuestros valores obtenidos por teclado. Esto se arregla con algo que se define **Castear**: que es poder cambiar los tipos de datos, pasar de número a string o de string a número. Esto se realiza con el siguiente comando:

Int() = para pasar a entero sin decimales

Float() = para pasar a numero entero con decimales

Str() = para pasar a una cadena de texto

Ejemplo

En nuestro caso queremos que **a** deje de ser entero y pase a ser un número, entonces realizamos lo siguiente:

```
a = int(a)
```

Dentro del paréntesis va la variable que se desea convertir, esta se puede igualar a cualquier variable como por ejemplo: **a_siendo_numero = int(a)**

Entonces ahora podremos realizar suma de 2 números cualquiera.

Ejemplo

```
a = input('ingrese el valor de la primera variable \n')
b = input('ingrese el valor de la segunda variable \n')
a= int(a)
b= int(b)
c = a+b
```

En el computador, quedaría de la siguiente manera:

```
>>> a = input('ingrese el valor de la primera variable \n')
ingrese el valor de la primera variable
34
>>> b = input('ingrese el valor de la segunda variable \n')
ingrese el valor de la segunda variable
453
>>> a= int(a)
>>> b= int(b)
>>> c = a+b
>>> c
487
```

7.2. Comando print

Ahora que ya manejamos de mejor manera el poder recibir datos, empecemos a mostrarlos de mejor forma: con el comando **print()**.

Como se les mencionó al momento de instalar Python y realizar su primer "Hola Mundo", el comando **print()** es para mostrar datos por consola. Pero no solo nos sirve para mostrar texto como el "Hola Mundo" que hicimos, sino también para mostrar nuestras variables de una forma más legible para un usuario.

Siguiendo con el ejemplo anterior, queremos mostrar **c** de manera ordenada:

```
print("el numero obtenido al sumar a con b es : " + c )
```

Si escribimos todo tal cual, nos arrojará un error, diciendo que 'c' no es un texto y, por lo tanto, no puede ser mostrado, ya que print solo puede mostrar caracteres de texto.

```
>>> print('el numero obtenido al sumar a con b es : ' + c )
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: Can't convert 'int' object to str implicitly
```

Entonces, lo que tenemos que hacer es convertir la variable 'c' en una cadena de texto con el comando **str()**. Esto se puede hacer dentro del mismo print, quedando de la siguiente manera:

```
print('el numero obtenido al sumar a con b es : ' + str(c) )
```

Ahora tenemos nuestro mensaje mostrado de la mejor manera posible:

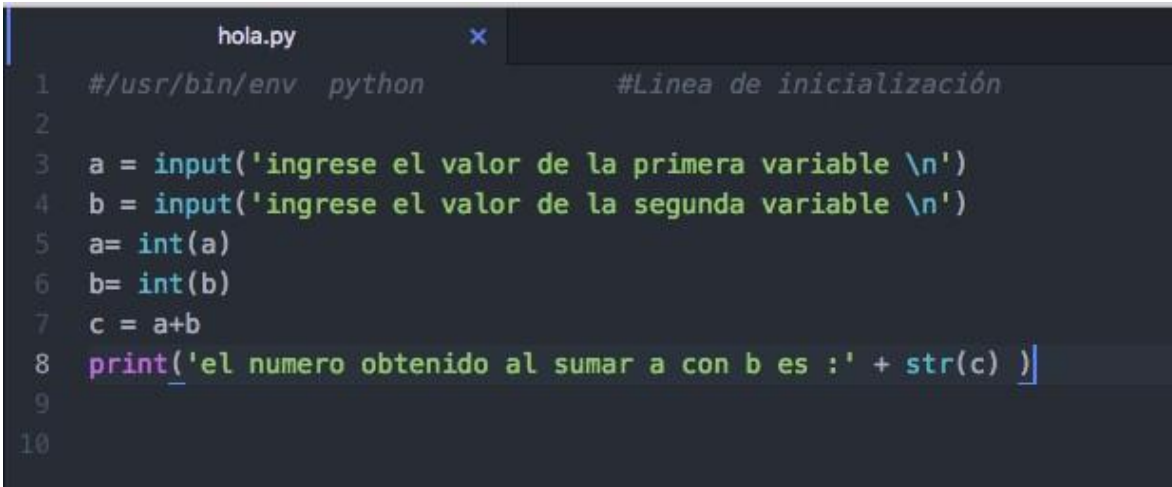
```
>>> print('el numero obtenido al sumar a con b es : ' + str(c) )
el numero obtenido al sumar a con b es :487
```

8. Utilización de archivo .py

Durante todas las pruebas que hemos realizado, escribir una y otra vez cada comando se ha vuelto tedioso y, si nos equivocamos, tenemos que empezar todo de nuevo. Pero esto tiene solución, ya que Python es un lenguaje interpretado y se pueden dejar una serie de comandos en un archivo con extensión '.py', en el que guardaremos todos los comandos y los podremos ejecutar una y otra vez con mayor facilidad.

Escribimos nuestros comandos en un editor de texto, que puede ser un bloc de notas en Windows, Linux o OSX. Pero también existen editores de texto creados específicamente para programar, que nos pueden facilitar en algunos momentos la codificación, como Sublime Text y Atom (los más conocidos).

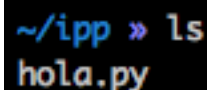
En este caso ocuparemos Atom, resultando de la siguiente manera:



```
hola.py x
1  #!/usr/bin/env python          #Linea de inicialización
2
3  a = input('ingrese el valor de la primera variable \n')
4  b = input('ingrese el valor de la segunda variable \n')
5  a= int(a)
6  b= int(b)
7  c = a+b
8  print('el numero obtenido al sumar a con b es : ' + str(c) )
9
10
```

El archivo se llama: **hola.py**

Para poder ejecutarlo, solo hay que llamarlo. Para este ejemplo ocuparemos el sistema operativo Ubuntu (Linux). Si vemos en nuestra carpeta con la terminal de Linux, el archivo se verá con el comando **ls**.



```
~/ipp » ls
hola.py
```

Para ejecutar nuestro programa se realizará con el siguiente comando:

python3 nombre_archivo.py

En este caso sería: **python3 hola.py** y se vería lo siguiente:

```
~/ipp » python3 hola.py
ingrese el valor de la primera variable
564
ingrese el valor de la segunda variable
28
el numero obtenido al sumar a con b es :592
-----
~/ipp » python3 hola.py
ingrese el valor de la primera variable
32432
ingrese el valor de la segunda variable
563453
el numero obtenido al sumar a con b es :595885
-----
```

También se simplificó en gran manera poder ejecutar dos veces el mismo programa, como muestra la imagen anterior.

9. Cierre

El conocimiento lógico de cualquier problema es de suma importancia para cualquier persona que quiera crear, analizar o supervisar alguna actividad. Este tipo de conocimiento lógico se integrará a lo largo del curso, y con la habilidad de obtener y mostrar variables a voluntad, les permitirá a los estudiantes aplicar constantemente dichas funciones durante toda la asignatura. Además, es importante saber cómo instalar el ambiente de trabajo en cualquier sistema operativo y poner en práctica continuamente lo aprendido.

APORTE A TU FORMACIÓN

En la vida laboral es posible trabajar tanto con la programación como con el pensamiento lógico. Al lograr separar problemas grandes en pequeñas secciones y analizar cada parte, se puede encontrar la mejor solución racional.

También es importante identificar qué tareas que se realizan mecánicamente en un computador se pueden automatizar, con el fin de disminuir la carga de trabajo metódico de las personas y permitirles dedicarse a análisis u otras labores.

Otra gran ventaja de la programación es la capacidad de crear programas o soluciones informáticas desde cero, simplemente identificando una necesidad y construyendo el programa módulo por módulo. Esto puede generar ganancias tanto en la empresa como en el negocio propio a corto o largo plazo.

10. Glosario modular

Main: Es punto de entrada de una aplicación o programa. Normalmente esta función es la que controla todo el programa, generando variables, llamando clases y dirigiéndolas a otras funciones. Es como el director técnico de un equipo de fútbol, quien elige las jugadas y sabe que alineación debe ocupar el equipo.

Bash: Es un programa basado en la estructura Unix (que es la ocupada en los computadores con Linux y OSX de Apple). Son comandos pre- programados que nos ayudan desde consola o un script (conjunto de comandos guardados en un archivo .sh), para poder comunicarnos con nuestra computadora. Bash tiene una gran variedad de comandos que pueden controlar cómo funciona el hardware de nuestro computador, mostrar la información de procesamiento, movernos entre carpetas dentro del computador, cambiar los permisos de acceso, apagar el computador y etc...

C y C++: Son lenguajes de programación. C es un lenguaje muy popular ocupado desde 1972 en los laboratorios de Bell y C++ es su versión mejorada, creada en 1980, con la ventaja que en C++ puede trabajar con objetos.

IDE: Integrated Development Environment, en español: Entorno de Desarrollo Integrado. Son herramientas que facilitan la programación, las cuales tienen funciones detectar errores, destacar funciones o variables, hasta incluso pueden auto corregir o sugerir cambios a un posible error que tenga el programador. Hoy en día existen millones de IDE, cada lenguaje de programación tiene su propios IDE, existen algunos que son universales para distintos tipos de lenguaje de programación,

o algunos que son complementos que pueden adaptar de mejor manera al programador o el lenguaje de programación.

Para Python los IDE más populares son:

Pycharm: Este IDE es creado por JetBrains, es pagado, pero está dentro de los mejores para la programación en Python.
<https://www.jetbrains.com/pycharm/>

Eclipse: Eclipse es un IDE conocido por la programación en Java, pero con un complemento llamado PyDev no crea un potente editor de código Python, se recomienda utilizar a gente que ya tiene experiencia en la programación con Python. Este IDE es libre de costo. <http://pydev.sourceforge.net/>

Thonny: IDE muy amigable y gratuito, se recomienda para la gente que recién comienza en la programación en Python. <http://thonny.org/>

SublimeText: Es uno de los IDE universales, para Python no tiene ningún problema, contiene un sinfín de complementos tanto para mejorar la programación, como también de personalización. Este IDE tiene 2 versiones, Free y uno pagada, aunque trabajar en la versión Free es totalmente recomendado. <https://www.sublimetext.com/>

Atom: IDE universal, gratuito creado por GitHub, es muy potente y también tiene las mismas ventajas de SublimeText, se puede personalizar con complementos para la programación o diseño. <https://atom.io/>