

Introducció a Python



Programació
amb Python



Unió Europea
Fons social europeu

Programación y datos



- Herramienta para manipular datos
- Datos = Información
- Números, símbolos, textos, sonidos, imágenes, vídeos...
- Información almacenable en medios digitales

Información manipulable

- Búsquedas en registros
- Solución de problemas
- Gestionar formularios
- Analizar datos y crear gráficos
- Programar videojuegos
- Etc



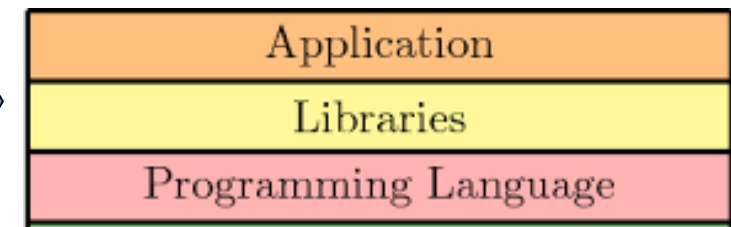
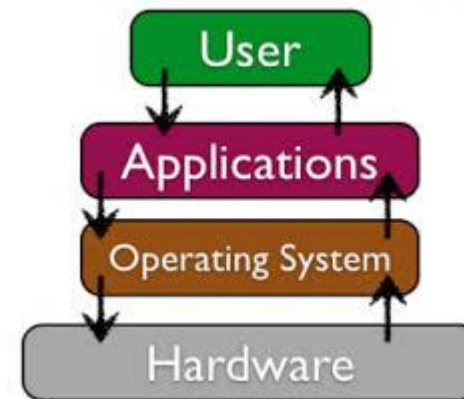
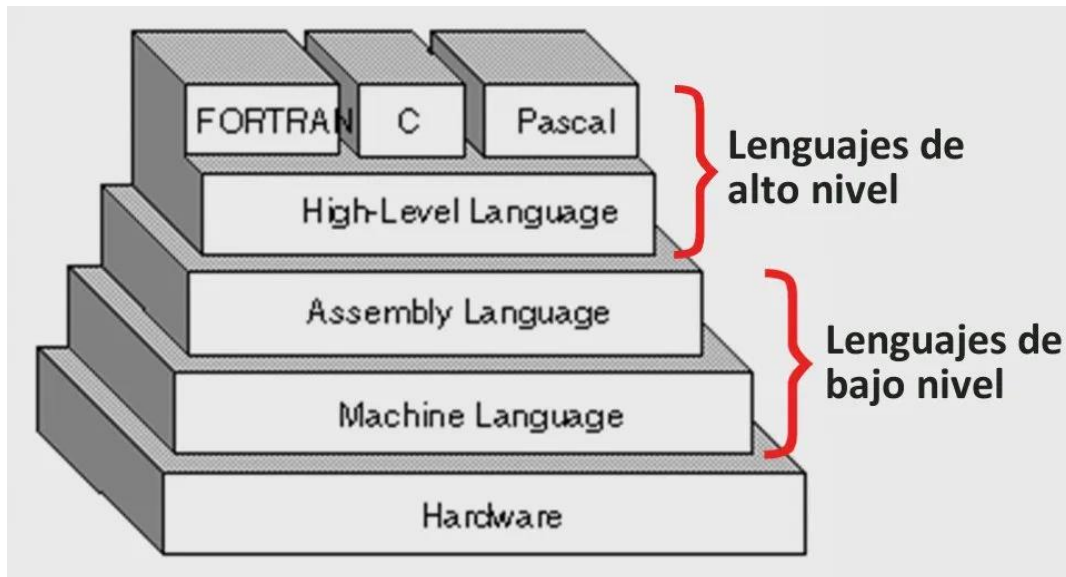
Sin datos la programación no tendría sentido

- Seleccionar,
- Calcular o transformar
- Mostrar o Entrar datos
- Generar animaciones

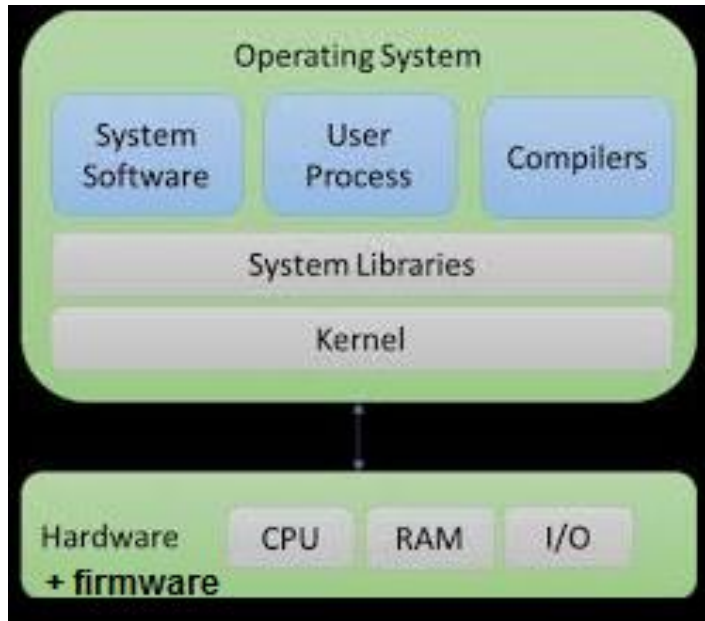
Qué es y cómo actúa un lenguaje de programación ?



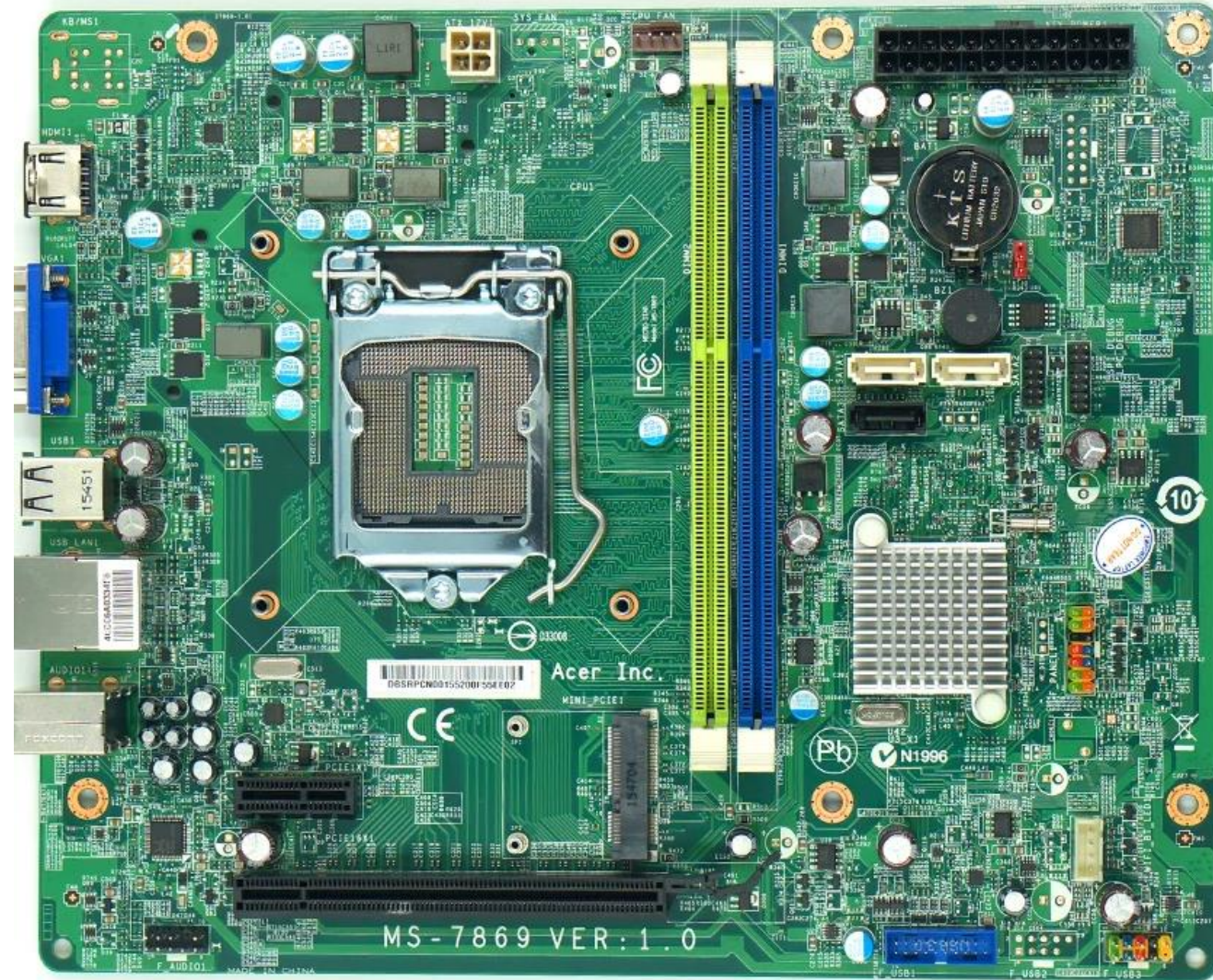
010100100011101101101001110
110101001110101101100011010
01100111101011101011101110



Qué estamos programando ?



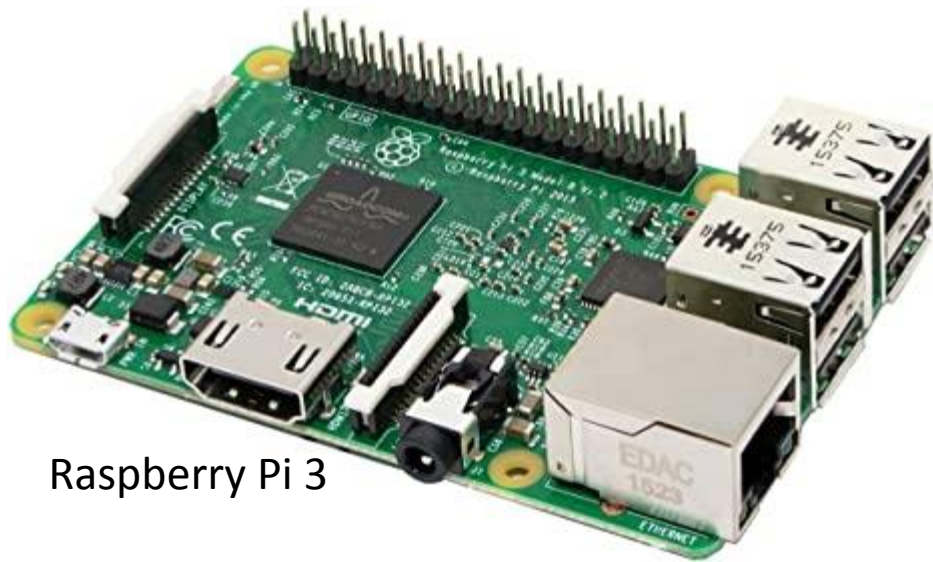
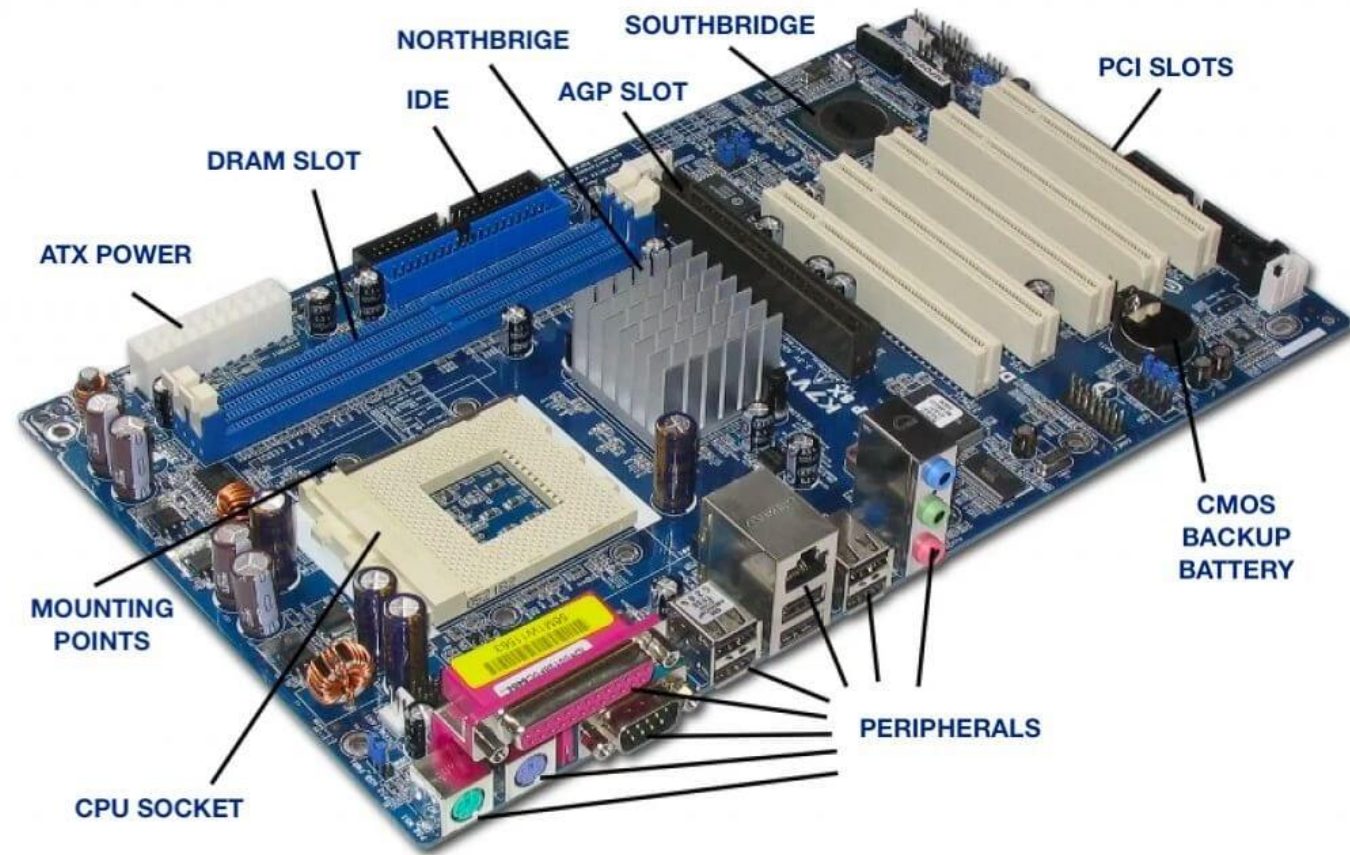
- CPU = Central Process Unit
- RAM = Random Access Memory
- I/O = Input output devices



Cada CPU tiene su propio juego de instrucciones y, en consecuencia, un código de máquina y uno o más lenguajes ensambladores propios. Un programa escrito para una CPU de la marca Intel no funcionará en una CPU diseñada por otro fabricante, como Motorola³.

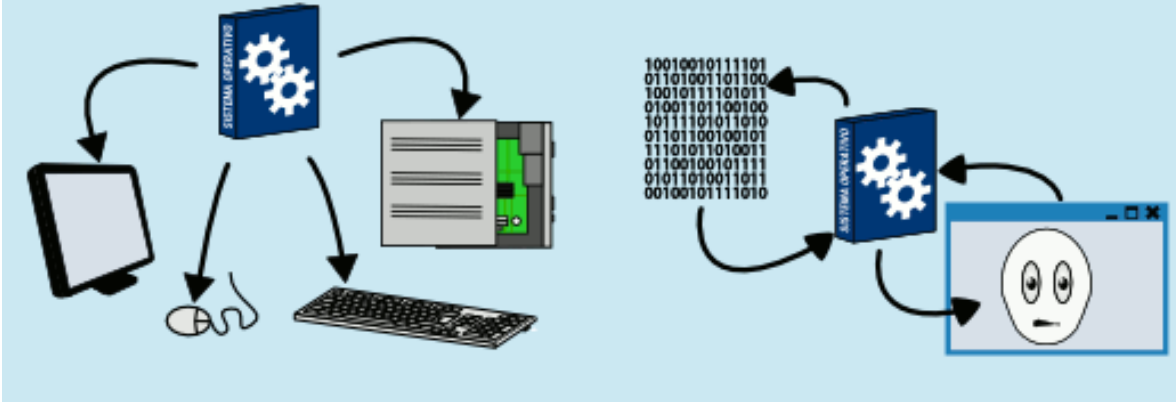
Hardware

```
004012A8 $ 53 PUSH EBX
004012A9 . 56 PUSH ESI
004012AA . 57 PUSH EDI
004012AB . 8BF2 MOV ESI,EDX
004012AD . 8BD8 MOV EBX,EAX
004012AF . 85F6 TEST ESI,ESI
004012B1 . 8BFB MOV EDI,EBX
004012B3 . 74 35 JE SHORT RTRACE.004012EA
004012B5 > 6A 04 PUSH 4
004012B7 . 68 00100000 PUSH 1000
004012BC . 68 00100000 PUSH 1000
004012C1 . 53 PUSH EBX
004012C2 . E8 15870000 CALL <JMP.&KERNEL32.VirtualAlloc>
004012C7 . 85C0 TEST EAX,EAX
004012C9 . 75 0F JNZ SHORT RTRACE.004012DA
004012CB . 8BD3 MOV EDX,EBX
004012CD . 8BC7 MOV EAX,EDI
004012CF . 2BD7 SUB EDX,EDI
004012D1 . E8 1E000000 CALL RTRACE.004012F4
004012D6 . 33C0 XOR EAX,EAX
004012D8 . EB 15 JMP SHORT RTRACE.004012EF
004012DA > 81C3 00100000 ADD EBX,1000
004012E0 . 81EE 00100000 SUB ESI,1000
004012E6 . 85F6 TEST ESI,ESI
004012E8 . 75 CB JNZ SHORT RTRACE.004012B5
004012EA > B8 01000000 MOV EAX,1
004012EF > 5F POP EDI
004012F0 . 5E POP ESI
004012F1 . 5B POP EBX
004012F2 . C3 RETN
```



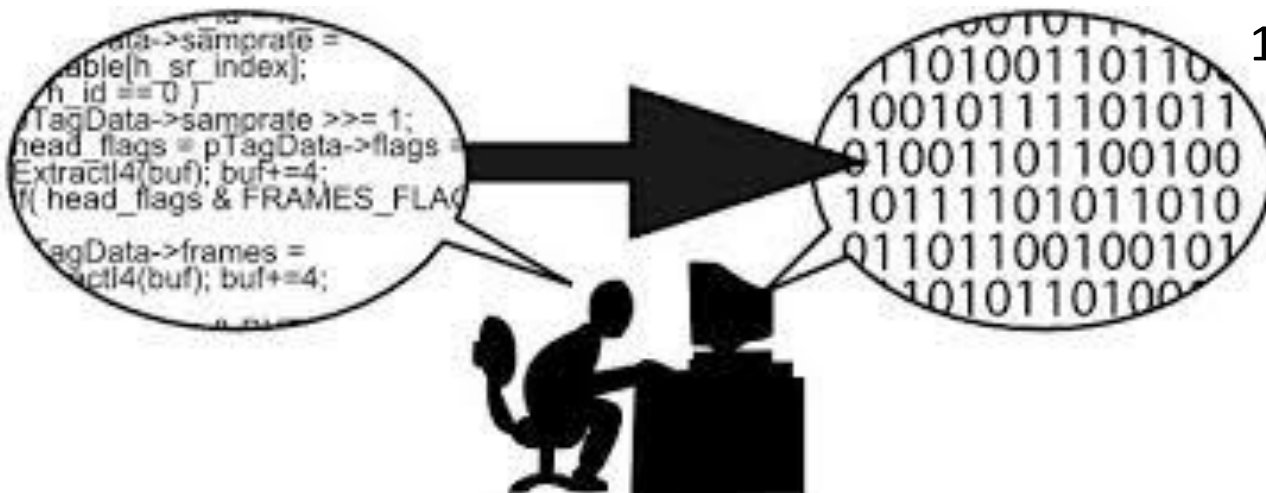
Raspberry Pi 3

FUNCIONES DEL SISTEMA OPERATIVO



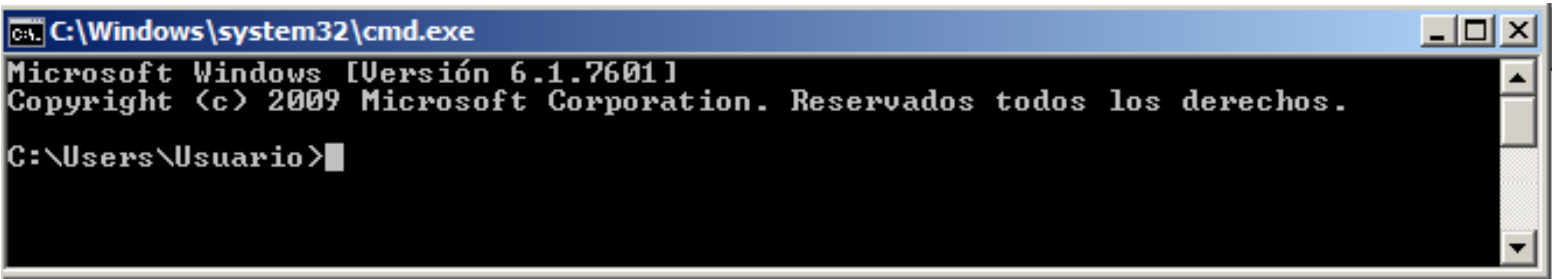
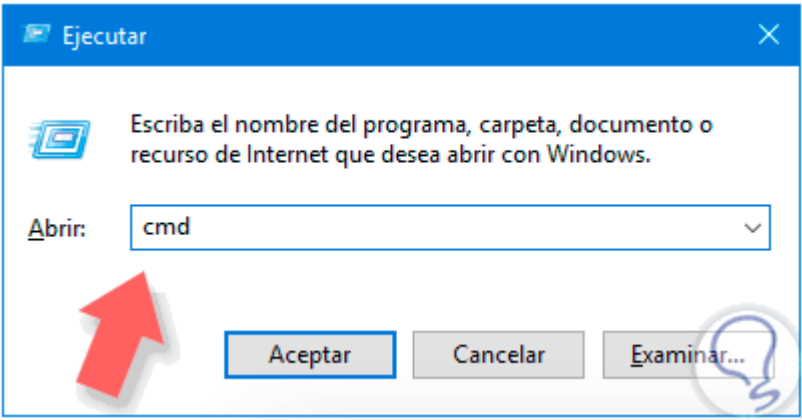
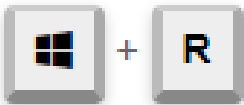
Sistema Operativo

1. Gestión de procesos (cálculos a cpu)
2. Gestión de la memoria principal
3. Gestión del almacenamiento secundario
4. Registro del sistema de archivos
5. Comunicación entre elementos y aplicaciones
6. Gestión del sistema de entrada y salida
7. Gestión de recursos
8. Seguridad
9. Informa del estado del sistema
10. Administración de usuarios

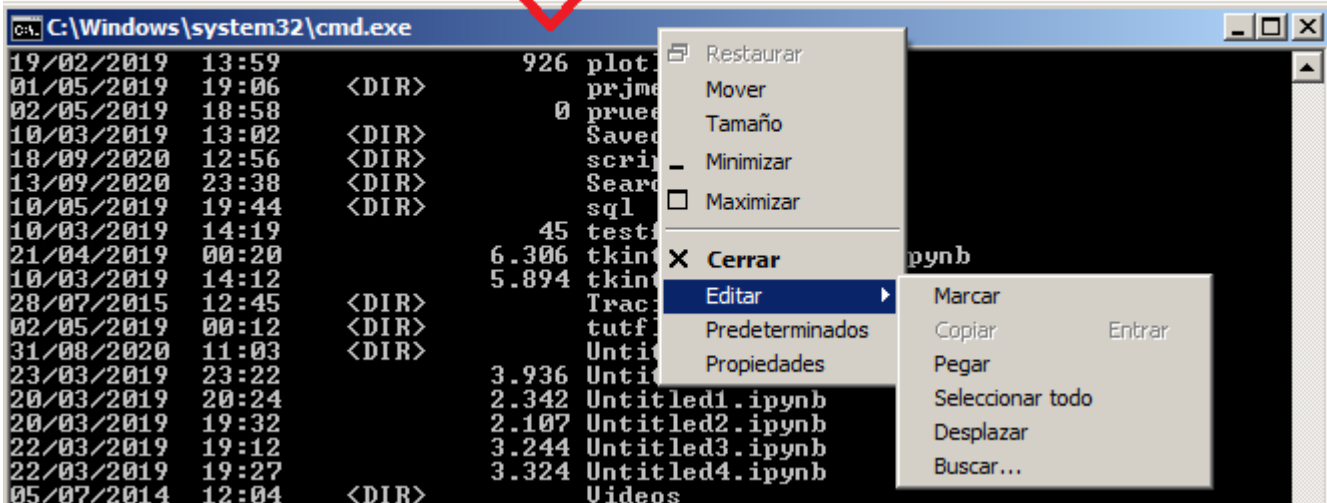


Abrir una ventana cmd de sistema

- En windows 10



dir



Comandos

Teclas

help

flechas

dir

dir

cd

cd

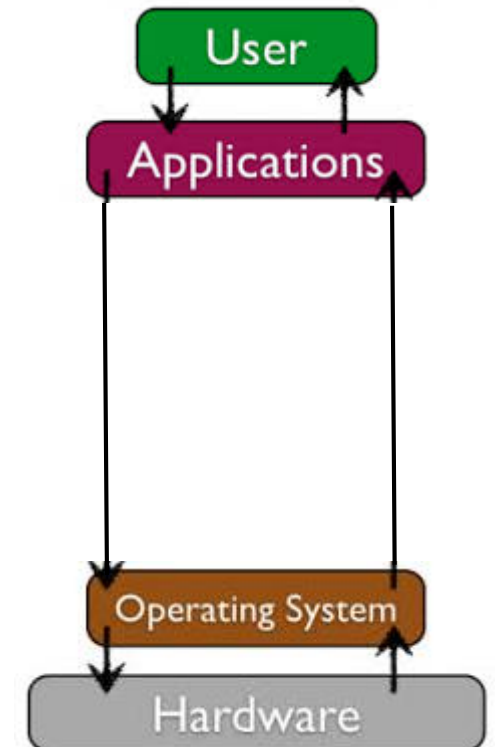
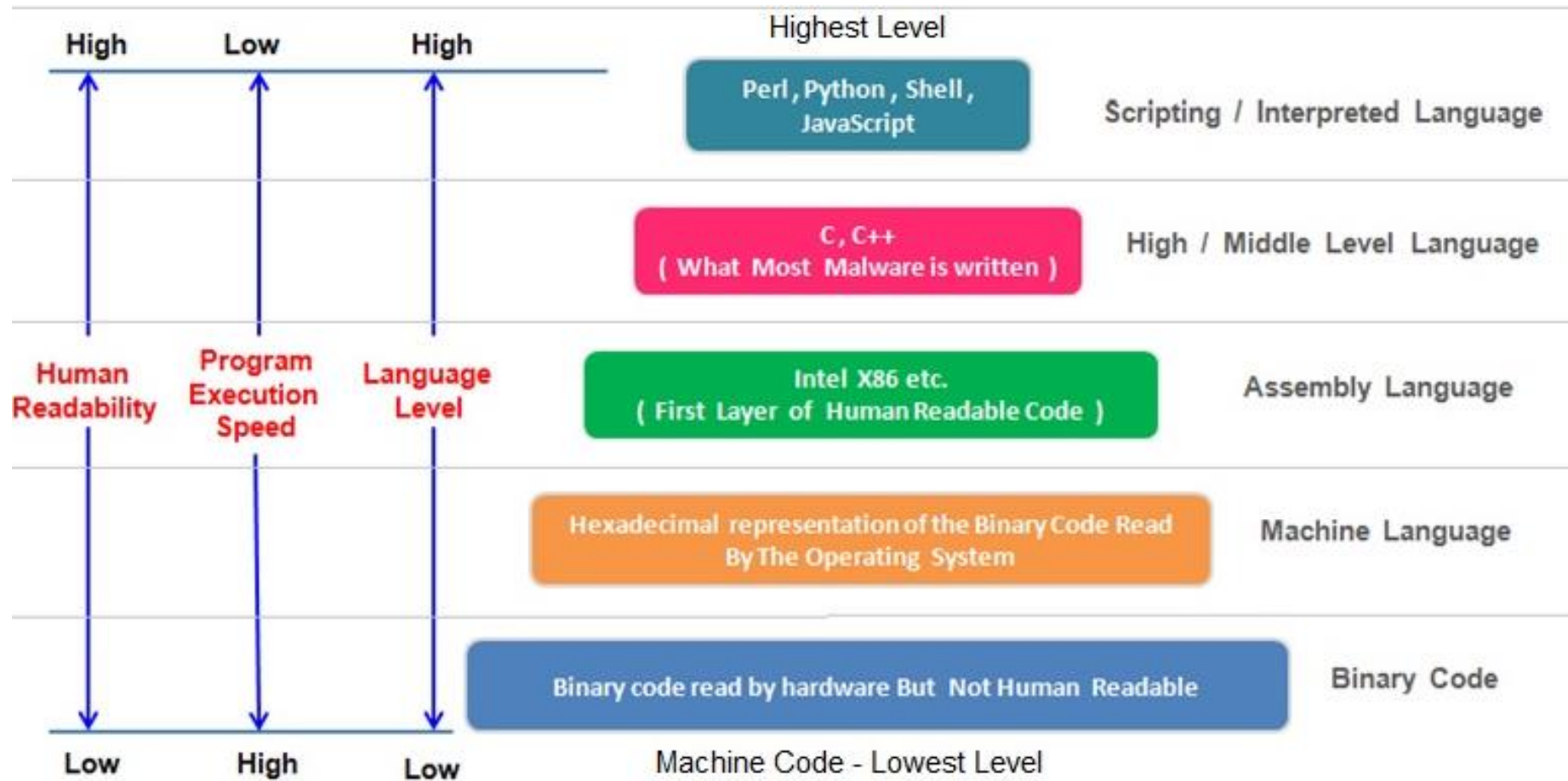
copy

copy

type

type

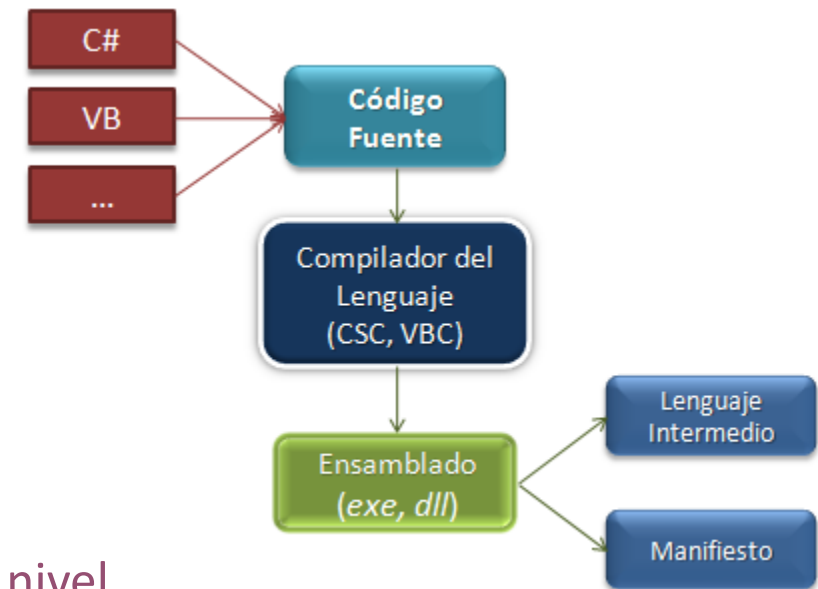
Lenguajes de sistema vs Lenguajes de aplicación



Compiladores e intérpretes

Los lenguajes de alto nivel se traducen automáticamente a código máquina.

Hay dos tipos diferentes de traductores dependiendo de su modo de funcionamiento: *compiladores* e *intérpretes*.



Un **compilador** lee completamente un programa en un lenguaje de alto nivel y lo traduce en su integridad a un programa de código de máquina equivalente.

El programa de código de máquina resultante se puede ejecutar cuantas veces se desee, sin necesidad de volver a traducir el programa original.



Un *intérprete* actúa de un modo distinto:

lee un programa escrito en un lenguaje de alto nivel instrucción a instrucción, traduce cada instrucción a código máquina y la ejecuta inmediatamente.

No hay un proceso de traducción separado por completo del de ejecución. Cada vez que ejecutamos el programa con un intérprete, se repite el proceso de traducción y ejecución, ya que ambos son simultáneos.



¿Qué es Python?



- Python es un lenguaje de programación creado por **Guido van Rossum** a principios de los años 90 cuyo nombre está inspirado en el grupo de cómicos ingleses “Monty Python”.
- Es un lenguaje similar a Perl, pero con una sintaxis muy limpia y que **favorece un código legible**.
- Se trata de un **lenguaje interpretado** o de script, con **tipado dinámico**, fuertemente tipado, **multiplataforma** y **orientado a objetos**.

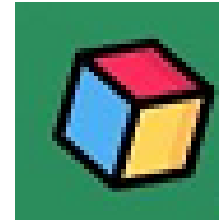


Tipado dinámico : su tipo se determinará en tiempo de ejecución según el tipo del valor al que se asigne

Fuertemente tipado : No se permite tratar a una variable como si fuera de un tipo distinto al que tiene, es necesario convertir de forma explícita dicha variable al nuevo tipo previamente

Multiplataforma

- El intérprete de Python está disponible en multitud de plataformas (**UNIX, Solaris, Linux, DOS, Windows, OS/2, Mac OS**, etc.) por lo que si no utilizamos librerías específicas de cada plataforma nuestro programa podrá correr en todos estos sistemas sin grandes cambios.



Orientado a objetos

- La orientación a objetos es un paradigma de programación en el que los conceptos del mundo real relevantes para nuestro problema se trasladan a clases y objetos en nuestro programa. La ejecución del programa consiste en una serie de interacciones entre los objetos.
- Python también permite la **programación imperativa**,

Variantes de Python

Existen varias implementaciones distintas de Python: CPython, Jython, IronPython, PyPy, etc.

- **CPython** es la más utilizada, la más rápida y la más madura. Cuando la gente habla de Python normalmente se refiere a esta implementación. En este caso tanto el intérprete como los módulos están escritos en C.
- **Jython** es la implementación en Java de Python, mientras que IronPython es su contrapartida en C# (.NET). Su interés estriba en que utilizando estas implementaciones se pueden utilizar todas las librerías disponibles para los programadores de Java y .NET.
- **PyPy**, es una implementación en Python de Python.

Ramas de Python

- Python 3.0 (también llamado "Python 3000" o "Py3K") fue lanzado el 3 de diciembre de 2008.
- Fue diseñado para rectificar fallos de diseño fundamentales en el lenguaje; los cambios requeridos no se pudieron conservar la compatibilidad total con versiones anteriores de 2. x.
- Se abrió una nueva rama 3 con un nuevo número de versión principal.
- En el 2020, tras 12 años, se deja de dar mantenimiento a la rama 2.x.

Instalación de Python

- CPython está instalado por defecto en la mayor parte de las distribuciones Linux y en las últimas versiones de Mac OS.
- Para comprobar si está instalado abre una terminal y escribe : `python`.
- Si está instalado se iniciará la consola interactiva de Python y obtendremos un mensaje parecido al siguiente:

```
Python 2.5.1 (r251:54863, May 2 2007, 16:56:35)  
[GCC 4.1.2 (Ubuntu 4.1.2-0ubuntu4)] on linux2  
Type "help", "copyright", "credits" or "license" for  
more information.
```

Ubuntu

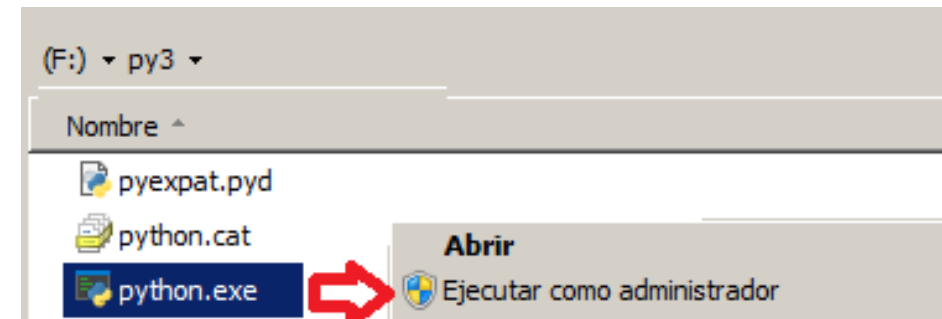
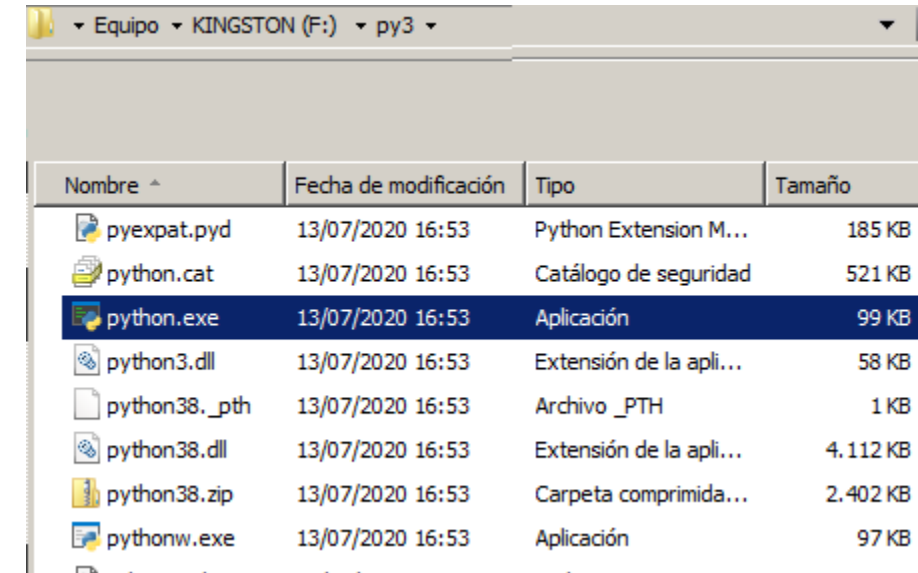
Windows

```
>>>
```

```
Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 22:20:52)  
[MSC v.1916 32 bit (Intel)] on win32  
Type "help", "copyright", "credits" or "license" for more information.  
>>>
```

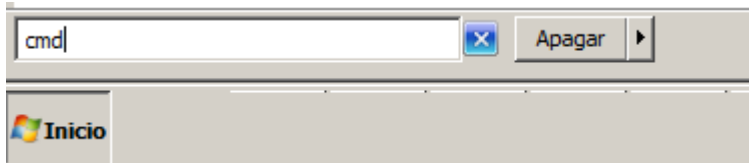
Instalando Python portable en windows

1. Copia la distribución de python que te da el tutor a tu **pen usb**.
2. Escucha la explicación sobre las carpetas del producto



Si no has podido abrir la consola python,

- abre una ventana **cmd** en windows, y sitúate en la **carpeta py3** de tu unidad usb, teclea python



```
C:\Users\Usuario>D:
D:\>cd py3
```

```
C:\Windows\system32\cmd.exe - python
D:\py3>python
Python 3.8.4 (tags/v3.8.4:dfa645a, Jun 10 2020) on win32
Type "help", "copyright", "credits" or "quit()"
>>>
```

3. Observa cual es la versión instalada.
4. Teclea en el **terminal python >>> print ("Hola mundo")**
5. Escucha la explicación para comprender como funciona el terminal de python
6. Vamos a usar python como una calculadora

```
>>> 3 + 10 / 5
```

```
>>> (3 + 10) / 5
```

```
>>> 7 / 3
```

```
>>> 7 // 3
```

```
>>> print("Hola mundo")
```

```
Hola mundo
```

```
>>> print("Hola mundo", " adios")
```

```
Hola mundo adios
```

```
>>> help(print)
```

```
Help on built-in function print in module builtins:
```

```
print(...)
```

```
    print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False)
```

```
    Prints the values to a stream, or to sys.stdout by default.
```

```
    Optional keyword arguments:
```

```
    file: a file-like object (stream); defaults to the current sys.stdout.
```

```
    sep: string inserted between values, default a space.
```

```
    end: string appended after the last value, default a newline.
```

```
    flush: whether to forcibly flush the stream.
```

```
>>> print("Hola mundo", " adios)
```

```
File "<stdin>", line 1
    print ('Hola mundo', 'adios)
SyntaxError: EOL while scanning string literal
```

```
>>> cp = input ("Código postal: ")
```

```
código postal:08003
```

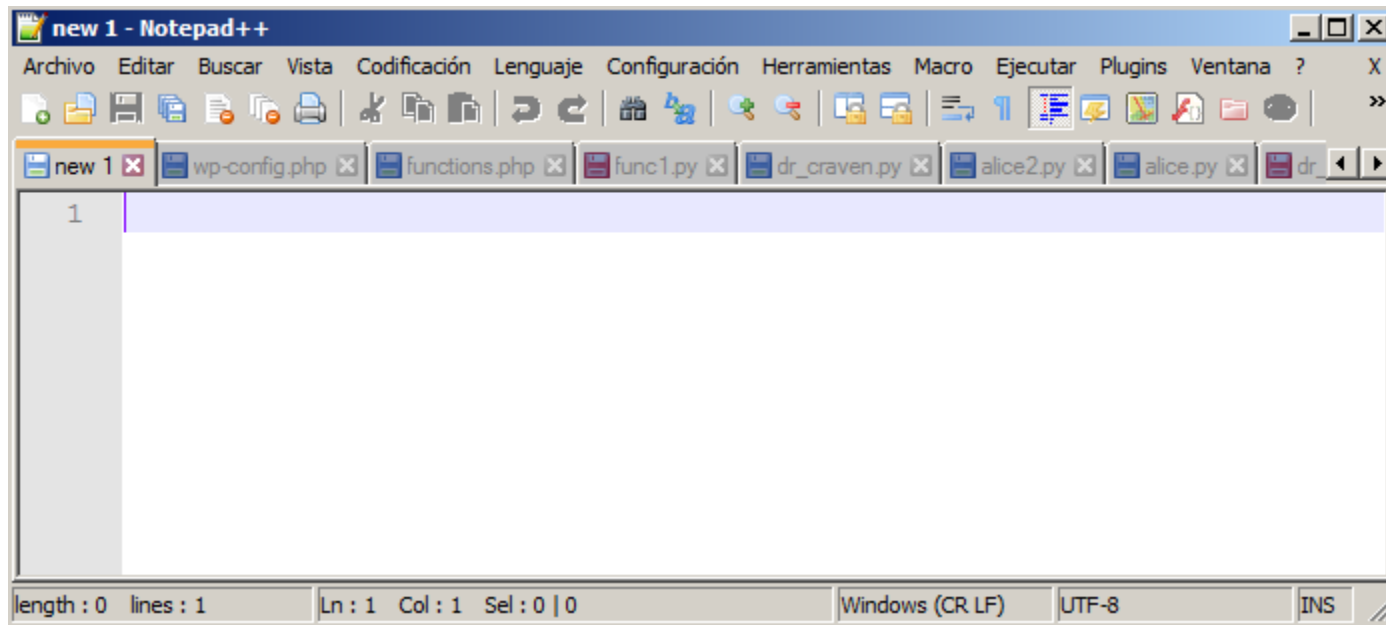
```
>>> cp
```

```
>>> cp
'08003'
```

```
>>> print (cp)
08003
```


Práctica P01

1. Descarga la aplicación [Notepad++](#) o cópiala desde el servidor del tutor. Instálala en tu ordenador.
2. Abre la aplicación y crea un Archivo Nuevo.

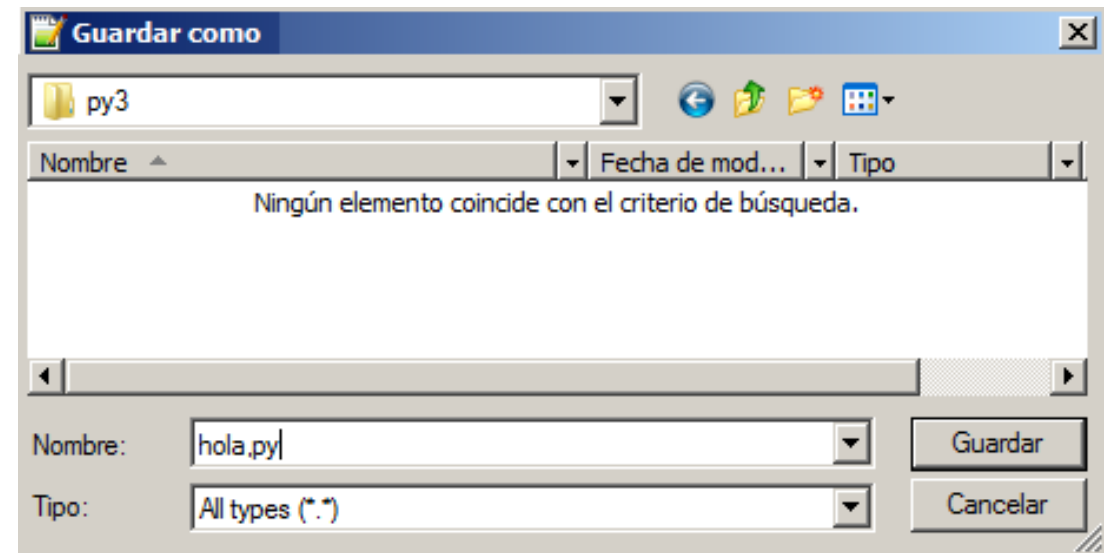
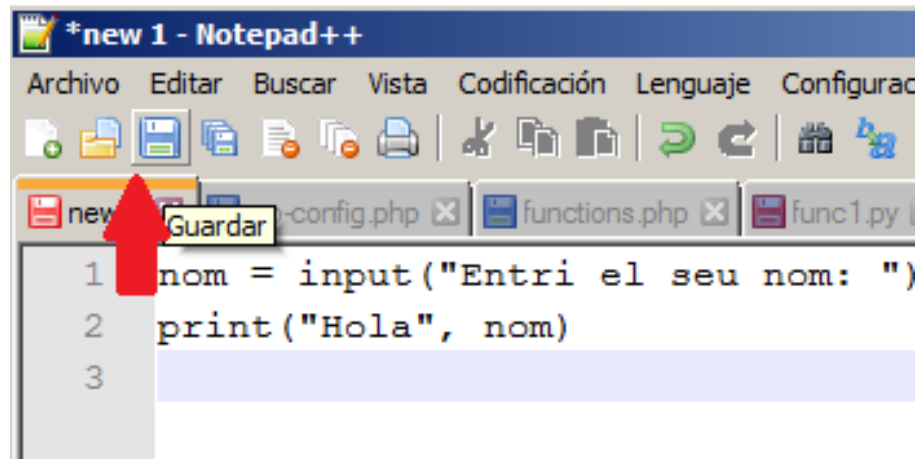


... sigue práctica P01

3. Escribe el siguiente fragmento de código

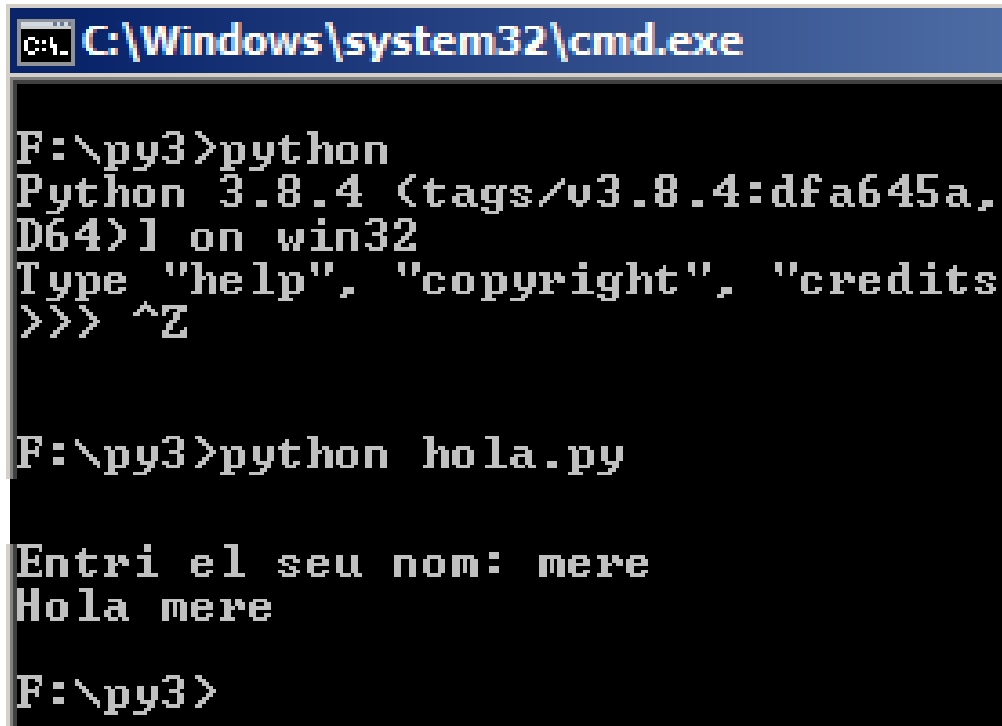
```
nom = input("Entri el seu nom: ")  
print("Hola" , nom)
```

4. Guárdalo como “All types” con el nombre “hola.py” en la carpeta de python de tu usb : d:\py3



... sigue práctica P01

5. Cambia a la ventana donde tienes abierta la consola python
6. Sal del interprete con **^Z**
7. Ejecuta el programa llamando: **python hola.py**



```
C:\Windows\system32\cmd.exe

F:\py3>python
Python 3.8.4 <tags/v3.8.4:dfa645a,
D64>1 on win32
Type "help", "copyright", "credits"
>>> ^Z

F:\py3>python hola.py

Entri el seu nom: mere
Hola mere

F:\py3>
```

Empezando con Python

1. Tipos de datos básicos
2. Tipos de datos contruidos
3. Operadores
4. Palabras Reservadas
5. Funciones básicas (built-in)
6. Sintaxis

Datos básicos



1. Números

2. Textos

Tipo	Clase	Notas	Ejemplo
int	Entero	Precisión fija, convierte a long si necesario	32
float	Decimal	Coma flotante de doble precisión	3.141592
complex	Complejo	Parte real e imaginaria.	(4.5 + 3j)
bool	Booleano	Valores verdadero o falso	True o False
str	Cadena	Inmutable	"Hola"
unicode	Cadena	Versión Unicode de str	u"Hola"

Datos construidos *built-in*

Listas: contenedor ordenado de objetos

Tuplas: similar a listas pero *inmutables*

Diccionarios: mapas clave-valor

Conjuntos: objetos únicos, no ordenado y más!

Tipo	Clase	Notas	Ejemplo
list	Secuencia	Mutable, contiene objetos de diverso tipo	[4, "Hola", 3.14]
tuple	Secuencia	Inmutable, contiene objetos de diverso tipo	(4, "Hola", 3.14)
dict	Diccionario	Pares de clave:valor	{"clave1": 4, "clave2": "Hola"}
set	Conjunto	Mutable, sin orden y sin duplicados	set([4, "Hola", 3.14])

Operadores

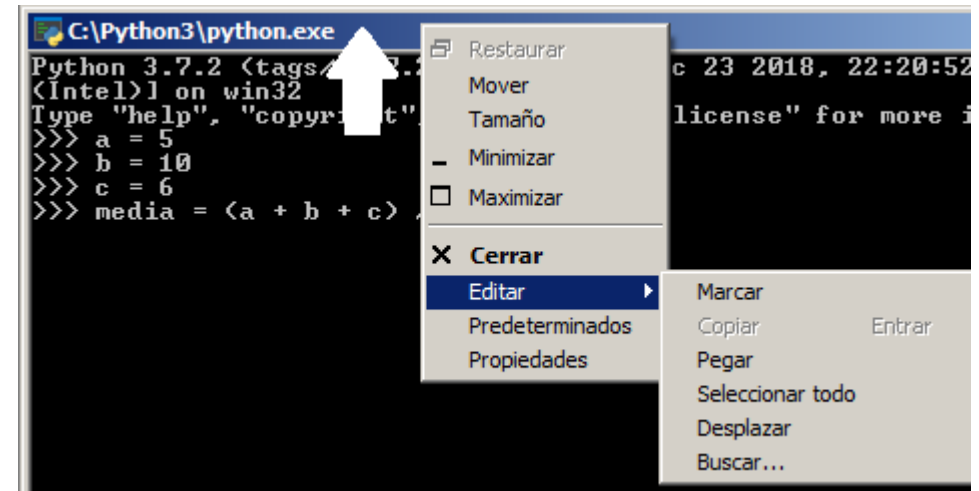
Expresión con operador	Operación
$A + B$	Suma
$A - B$	Resta
$A * B$	Multiplicación
$A \% B$	Resto
A / B	División real
$A // B$	División entera
$A ** B$	Potencia
$A B$	OR(bit)
$A ^ B$	XOR(bit)
$A \& B$	AND(bit)
$A == B$	Igualdad
$A != B$	Desigualdad
$A \text{ or } B$	OR(Lógica)
$A \text{ and } B$	AND(Lógica)
$\text{not } A$	Negación(Lógica)

Ejercicio:

Copia este código a la consola de python

```
a = 5
b = 10
c = 6
media = (a + b + c) / 3
```

Como imprimieras la media ? Cuánto da ?



Ejercicio: con $a = 12$ y $b = 5$

Prueba la división normal y la división entera

Palabras reservadas

<code>False</code>	<code>class</code>	<code>finally</code>	<code>is</code>	<code>return</code>
<code>None</code>	<code>continue</code>	<code>for</code>	<code>lambda</code>	<code>try</code>
<code>True</code>	<code>def</code>	<code>from</code>	<code>nonlocal</code>	<code>while</code>
<code>and</code>	<code>del</code>	<code>global</code>	<code>not</code>	<code>with</code>
<code>as</code>	<code>elif</code>	<code>if</code>	<code>or</code>	<code>yield</code>
<code>assert</code>	<code>else</code>	<code>import</code>	<code>pass</code>	
<code>break</code>	<code>except</code>	<code>in</code>	<code>raise</code>	

Cuando crees un identificador, si lo haces en inglés, respeta las palabras reservadas para el lenguaje.

Qué es un identificador ?

Python funciones básicas (built in core)

Built-in Functions				
abs()	delattr()	hash()	memoryview()	set()
all()	dict()	help()	min()	setattr()
any()	dir()	hex()	next()	slice()
ascii()	divmod()	id()	object()	sorted()
bin()	enumerate()	input() *	oct()	staticmethod()
bool()	eval()	int()	open()	str()
breakpoint()	exec()	isinstance()	ord()	sum()
bytearray()	filter()	issubclass()	pow()	super()
bytes()	float() *	iter()	print() *	tuple()
callable()	format()	len()	property()	type()
chr()	frozenset()	list()	range()	vars()
classmethod()	getattr()	locals()	repr()	zip()
compile()	globals()	map()	reversed()	__import__()
complex()	hasattr()	max()	round() *	

```
nom = input("Entri el seu nom: ")  
print("Hola" , nom)
```

Ejercicio:

Prueba este programa python

```
text = input("Entri un nombre decimal : ")  
num = float (text);  
print ("Arrodoneixo : ", round(num))
```

Sintaxis

- Reglas sintácticas que deciden si una instrucción tiene una forma correcta.
- Las estudiaremos sobre la marcha, solo apunto algunos ejemplos :
 - Los **identificadores** (nombres de variables, funciones, etc)
 - deben empezar con una letra
 - Se consideran diferentes las letras mayúsculas y minúsculas
 - Etc.
 - Las **instrucciones**
 - Si empiezan con el carácter # se consideran comentarios
 - La indentación de la instrucciones determina su dependencia jerárquica
 - Se pueden incluir varias sentencias en una instrucción separando con el carácter “;”
 - Una sentencia se puede escribir en varias líneas usando a final de cada línea \
 - Etc.

Práctica P02

1. Realiza un programa python que pida una medida en centímetros y la convierta y muestre en pulgadas : [cms_a_pulg.py](#)
2. Escribe líneas de comentario indicando el nombre del programa, su descripción, el autor y la fecha de creación.

#-----

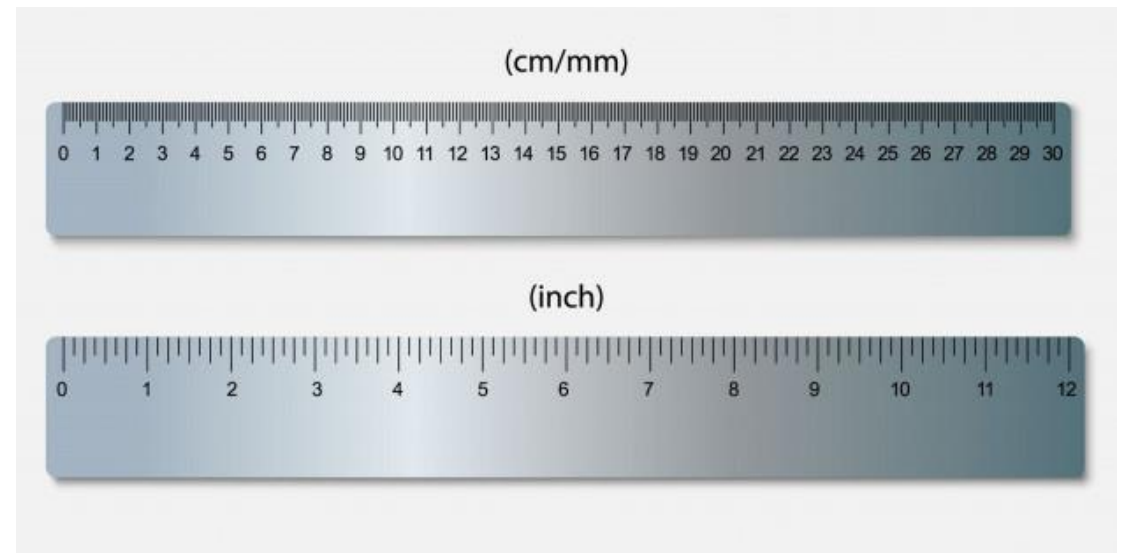
Nombre :

Descripción :

Autor :

Fecha creación :

#-----



Práctica P03

1. Realiza un programa python que calcule los pasos de un robot : [pasos_robot.py](#)
2. Debe pedir una distancia a recorrer (en centímetros) y también el paso del robot en centímetros.
3. Debe mostrar el número de pasos a realizar (procura que no choque ni caiga en abismos).
4. No olvides los comentarios iniciales con el nombre del programa, su descripción, el autor y la fecha de creación.



Práctica P04

Visita estas webs i coméntalas en grupo :

- Web oficial de Python: <https://www.python.org/>
- Documentaciones oficiales en castellano : <https://docs.python.org/es/3/>
- W3 Schools : <https://www.w3schools.com/python/>
- Hektor Profe : <https://docs.hektorprofe.net/python/>
- Fin de la versión 2

<https://www.genbeta.com/desarrollo/20-anos-desarrollo-finaliza-soporte-python-2>

¿ Python por qué ?

- Python es un lenguaje que todo el mundo debería conocer.
- Su **sintaxis simple, clara y sencilla**; el tipado dinámico, el gestor de memoria, la **gran cantidad de librerías disponibles** y la potencia del lenguaje, entre otros, hacen que desarrollar una aplicación en Python sea sencillo, **muy rápido** y, lo que es más importante, divertido.



Desarrollo de
aplicaciones para la
educación

```
nom = input("Entri el seu nom: ")  
print("Hola" , nom)
```