In [ ]:
```
#  Esta celda es exclusivo para cuestiones de impresion dentro del notebook
from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = "all"
```

Introducción a Python

- ## CPython 3.7
- ## 3.8 finales de Octubre 2019

PEP 373

Indice TIOBE

TIOBE programming Community Index

StackOverflow

Stack Overflow Developer Survey 2019

- ## Por qué Python?

Sintaxis sencilla

Baterias incluidas

Open Source

Compañías que usan Python

- ## Qué es Python?

> Python es un lenguaje de programación interpretado cuya filosofía hace hincapié en una sintaxis que favorezca un código legible. Se trata de un lenguaje de programación multiparadigma.
> , Es un lenguaje interpretado, de tipado fuerte y dinámico y multiplataforma.

Wikipedia

Interpretado

Multiparadigma

- Orientado a Objetos
- Imperativa
- Funcional

Multifuncional

- Web
- Data Science
- Machine Learning
- Aplicaciones GUI
- Robotica
- IOT
- etc

Multiplataforma

- Windows
- Linux
- Mac OS
- etc

- ## PEP 8

Que es el PEP 8?

Python Enhancement Proposals(Propuesta de Mejora de Python)
Guia de estilo para codigo

1. Estilo de codigo
2. Nombre de variables, metodos y clases
3. Comentarios y documentacion
4. etc.

In [18]:
```
import this
```

```
The Zen of Python, by Tim Peters

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!
```

> Bello es mejor que feo.
> Explicito es mejor que implicito.
> Simple es mejor que complejo.
> Complejo es mejor que complicado.
> Plano es mejor que anidado.
> Disperso es mejor que denso. La legibilidad cuenta.
> Los casos especiales no son tan especiales como para quebrantar las reglas.
> Lo práctico gana a lo puro.

> Los errores nunca deberían dejarse pasar silenciosamente.
> A menos que hayan sido silenciados explícitamente.
> Frente a la ambigüedad, rechaza la tentación de adivinar.
> Debería haber una y preferiblemente solo una manera obvia de hacerlo.
> Aunque esa manera puede no ser obvia al principio a menos que usted sea holandés.
> Ahora es mejor que nunca.
> Aunque nunca es a menudo mejor que ya mismo.

> Si la implementación es difícil de explicar, es una mala idea.
> Si la implementación es fácil de explicar, puede que sea una buena idea.
> Los espacios de nombres (namespaces) son una gran idea ¡Hagamos más de esas cosas!

Tim Peters, El Zen de Python

- ## Instalación

- ### Windows

- ### Unix (Linux/Mac OS)

```
sudo apt update
sudo apt install software-properties-common
sudo apt install python3.7
```

- ## Interprete (REPL)


```
>>> python
```

```
>>> python3
```

```
>>> python3.7
```

función `help()`

La función help devuelve la documentación (docstring) de clases, funciones y metódos que pasen por argumento

In [13]:
```
help(str.capitalize)
```

```
Help on method_descriptor:

capitalize(self, /)
    Return a capitalized version of the string.

    More specifically, make the first character have upper case and the res
t lower
    case.
```

función `dir()`

la función dir devuelve el nombre de las variables del entorno, si no tiene argumento devuelve las variables del entorno (namespace) de donde se llama, como argumento puede recibir clases, funciones/metodos o módulos, y devuelve las variables que componen a estas

In [15]:
```
print(dir(str))
```

```
['__add__', '__class__', '__contains__', '__delattr__', '__dir__', '__doc__
', '__eq__', '__format__', '__ge__', '__getattribute__', '__getitem__', '__
getnewargs__', '__gt__', '__hash__', '__init__', '__init_subclass__', '__le
__', '__len__', '__lt__', '__mod__', '__mul__', '__ne__', '__n
ew__', '__reduce__', '__reduce_ex__', '__repr__', '__rmod__', '__rmul__',
'__setattr__', '__sizeof__', '__str__', '__subclasshook__', 'capitalize',
'casefold', 'center', 'count', 'encode', 'endswith', 'expandtabs', 'find',
'format', 'format_map', 'index', 'isalnum', 'isalpha', 'isascii', 'isdecima
l', 'isdigit', 'isidentifier', 'islower', 'isnumeric', 'isprintable', 'isspa
ce', 'istitle', 'isupper', 'join', 'ljust', 'lower', 'lstrip', 'maketran
s', 'partition', 'replace', 'rfind', 'rindex', 'rjust', 'rpartition', 'rspl
it', 'rstrip', 'split', 'splitlines', 'startswith', 'strip', 'swapcase', 't
itle', 'translate', 'upper', 'zfill']
```

Hola Mundo

In [6]:
```
print("Hola Mundo")
```

```
Hola Mundo
```

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

la función imprime n valores separados por default por un espacio y al final del renglón terminado por un salto de linea. La salida se imprime en consola y el buffer se conserva.

In [10]:
```
saludo = "Hola"
nombre = "Arturo"
complemento = "como estas?"
print(saludo, nombre, complemento,"***", end='.')
```

```
Hola Arturo como estas? ***.
```

`input(prompt=None)`

Lee una cadena de texto desde la linea de comando, podemos agregar una cadena la cual saldra en la linea de comandos.

In [14]:
```
edad = int(input("Cual es tu edad? "))
edad
type(edad)
```

```
Cual es tu edad? 21
```

Out[14]:
```
int
```

Variables y constantes

Las variables en Python no requieren de especificar el tipo de dato que se va a emplear, pero es posible usar notaciones (typing) como guia, estas no se infieren en tiempo de ejecucion en Cpython, si desea que los tipos de datos sean inferidos puede usar herramientas como mypy

Para que el identificador de la variable sea valida debe iniciar con un caracter minusculo o mayusculo, o un guion bajo, seguido de estos mismos caracteres o numeros, de lo contrario se arrogara un SyntaxsError. Por convención en el PEP 8, las variables se escriben con minusculas, separadas por guiones bajos; el nombre debe ser alusivo al uso que se le debe dar.

Las constantes en Python no existen, sin embargo, por convención se escriben en mayusculas, separados por guión bajo, para indicar a los desarrolladores de la intención de uso.

In [16]:
```
#  Nuevos ejemplos de variables
entero = 10
flotante = 29.87
numeros_pares = [2, 4, 6, 8, 10]
directorio_locales = ['pizzeria', '5 de Mayo 213',
                      'abarrotes', 'Revolucion 70 B']

#  Malos ejemplos de variables
variable_buena = False  # ???

#  Ejemplos de constantes
PI = 3.1416
PI = 4
PI
```

Out[16]:
```
4
```

In [ ]:
```
#  Uso de typing 3.6+
contador_eventos: int = 0
correo_electronico: str = "curso.python@correos.com"

#  Un soltero tipo de dato
promedio: int = 89.6
promedio

#  Para tipos mas complejos
from typing import List

calificaciones: List[float]= [90.7, 60.6, 74.23, 100.0]
calificaciones
```

Función main

Que usamos como main?

```
if __name__ == '__main__':
```

Nosotros podemos usar un archivo de Python como:

> 1. Archivo: Tipicamente, un archivo Python es cualquier archivo que contenga código. La mayoria de los archivos Python tienen la extensión.py

> 1. Script: Un script Python es un archivo que intenta ejecutar desde la linea de comandos para realizar una tarea.

> 1. Módulo: Un módulo Python es un archivo que desea importar desde otro módulo o script, o desde el intreprete interactivo. Puede leer más sobre los modulos en la documentación de Python.

Real Python

Que es `__name__`?

Respuesta corta: una variable del namespace principal

Respuesta larga: una variable del namespace principal que indica si el archivo .py es el que se esta ejecutando

In [17]:
```
__name__
```

Out[17]:
```
'__main__'
```

Usos de `if __name__ == '__main__':`

- Indicar el inicio de nuestro codigo
- Realizar casos de prueba en nuestros modulos
- Optimizar codigo*