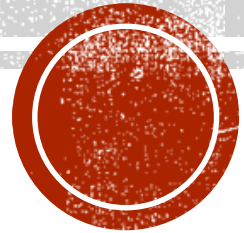
# PYTHON

**Christian Camilo Urcuqui López, MSc**




UNIVERSIDAD ICESI | A OTRO NIVEL

# PRESENTACIÓN


Christian Camilo Urcuqui López

Ing. Sistemas, Magister en Informática y Telecomunicaciones

Big Data Professional

Big Data Scientist

Deep Learning Specialization

Grupo de investigación i2t

Líder de investigación y desarrollo

Ciberseguridad y ciencia de datos aplicada

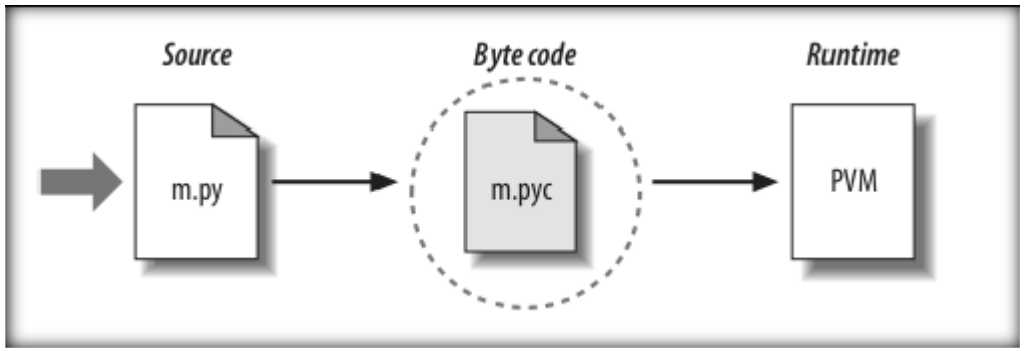
ccurcuqui@icesi.edu.co

# COMPETENCIAS

- Describir el lenguaje de programación Python y su aplicación en proyectos de inteligencia artificial.
- Utilizar el entorno de trabajo Jupyter Notebook.
- Utilizar GitHub para el control de versiones de proyectos de software.

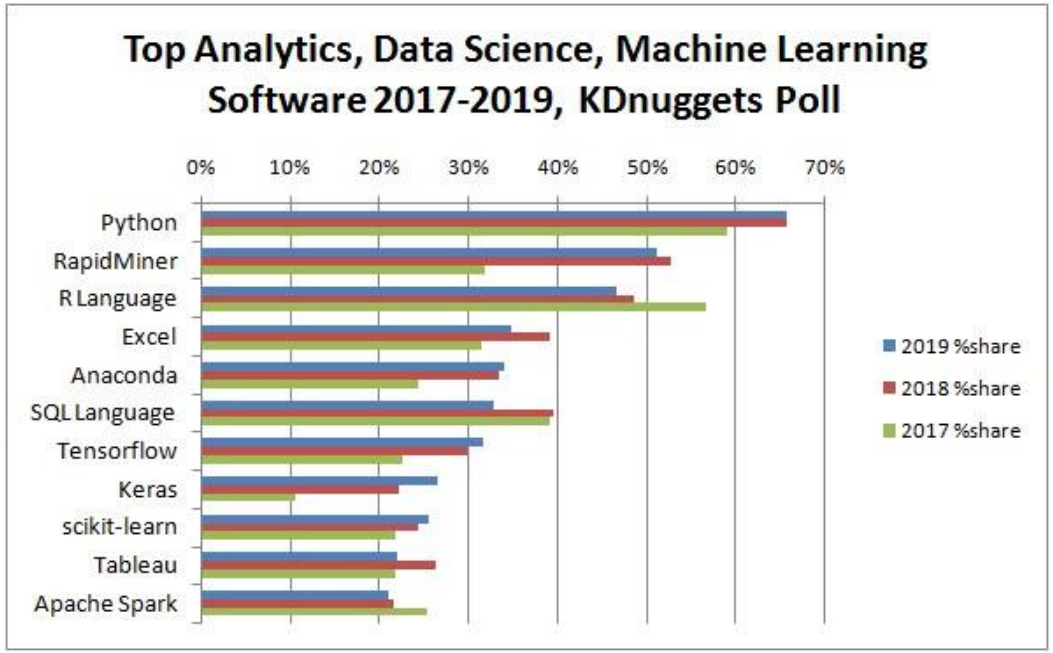
# PYTHON

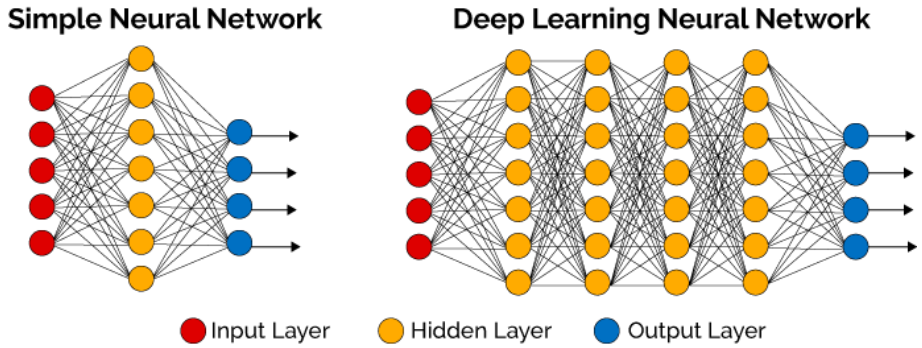
- Es un lenguaje popular orientado a objetos
- Es un lenguaje de libre acceso
- Puede ser utilizado para el desarrollo de programas standalone y scripts que pueden ser aplicados a muchos contextos o problemas.
- No es lenguaje tan rápido como C y C++




# PYTHON

- Según la KDnuggets [1], Python esta en el primer lugar en su top 10
- Es uno de los lenguajes más utilizados para las redes neuronales profundas



# PYTHON

- Es un lenguaje orientado a objetos
- Es *open source.*
- Es portable.
- Tiene una comunidad de expertos que día a día trabaja en nuevas utilidades.

# PYTHON

- **Interfaces para sistemas operativos**, la librería estándar de Python incorpora mecanismos de POSIX (Portable Operating System Interface), es decir, se pueden desarrollar programas que permiten la comunicación entre el sistema operativo y el entorno.
- **GUI**, Python incorpora una interfaz orientada a objetos a través de la API Tk GUI [1] (llamada Tkinter), la cual permite a los programas de Python implementar GUI (Graphical User Interface) portables con un aspecto y comportamiento nativo.
- **Internet Scripting**, existe una variedad de librerías que permiten el desarrollo de utilidades en Python para tareas de redes tanto a nivel de cliente y servidores, por ejemplo comunicación a través de sockets para el envío de paquetes TCP y UDP.

[1] http://www.tcl.tk/

# PYTHON

- **Programación de base de datos**, se provee una persistencia a los objetos y además existen librerías para la gestión de base de datos como Oracle, MySQL, PostgresSQL, entre otros.
- **Programación numérica**, Numpy es un paquete científico que incluye herramientas para operaciones con arreglos N dimensionales, operaciones lineales, transformadas de Fourier, entre otros.
- **Juegos, IA, XML y más…**, debido a que es un lenguaje abierto, las distintas comunidades han desarrollado y aportado con librerías para múltiples dominios, por ejemplo para aprendizaje de máquina esta scikit learn y para análisis de tráfico de red esta pyshark (utiliza los mecanismos ofrecidos por Wireshark).

# IDE

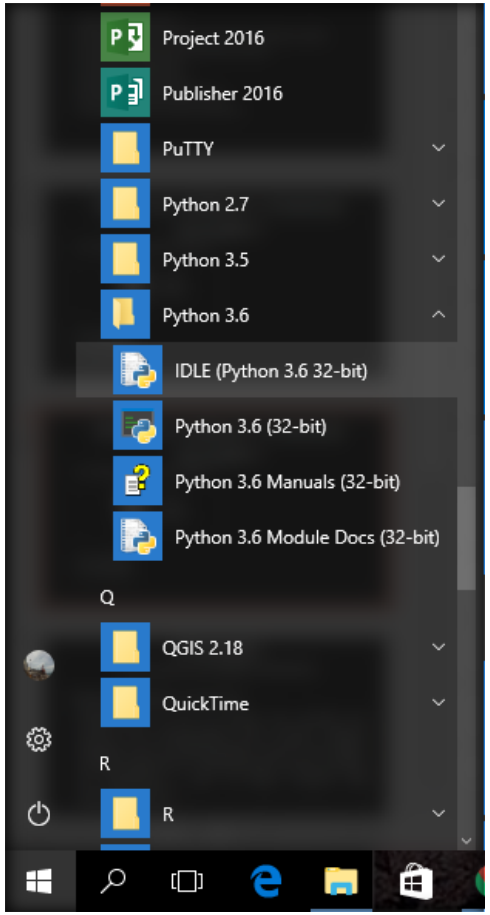
Entorno de desarrollo integrado (Integrated Development Environment)

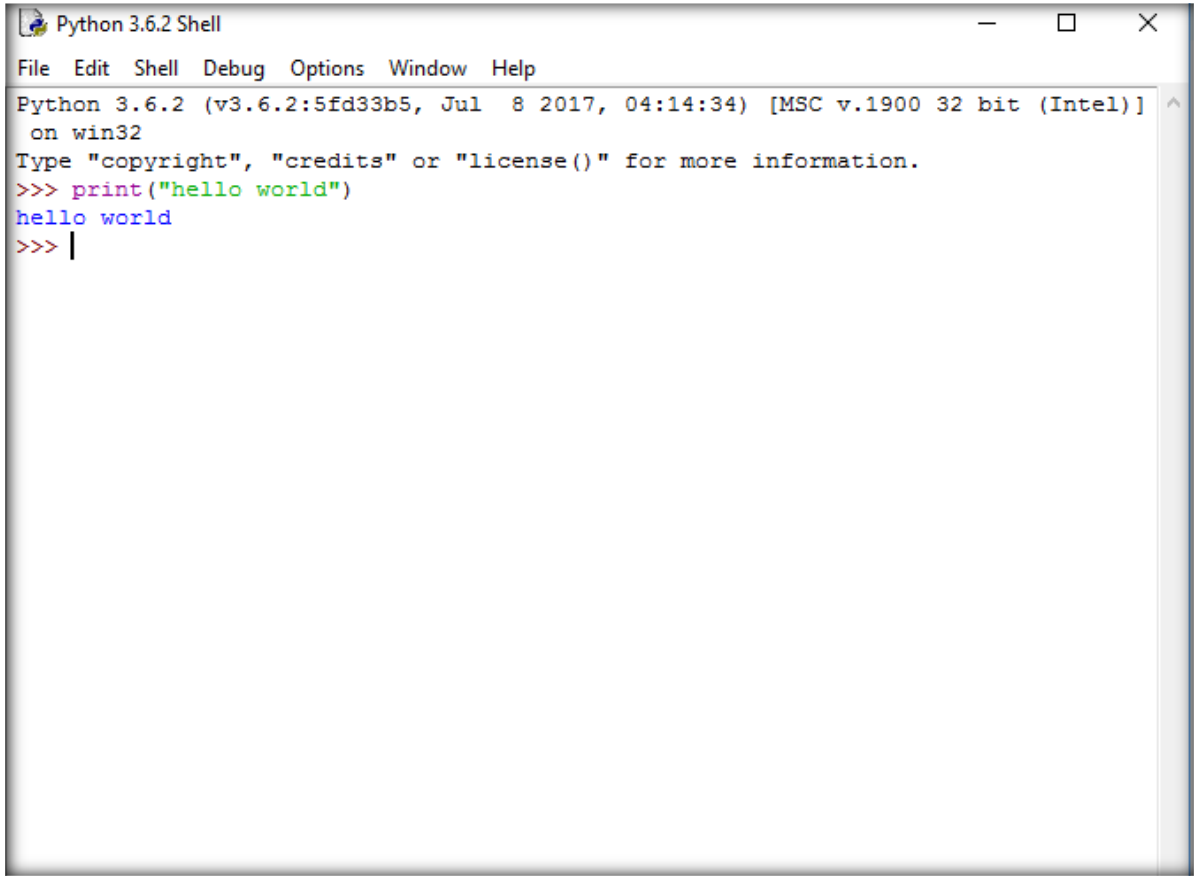
Es un editor de texto plano para programar que permite probar, compilar/interpretar/ejecutar.

# NUESTRAS HERRAMIENTAS



# PYTHON POR CONSOLA



# PYTHON - DIFERENCIA ENTRE VERSIONES

Python 3.6.2 Shell

File Edit Shell Debug Options Window Help

```
Python 3.6.2 (v3.6.2:5fd33b5, Jul  8 2017, 04:14:34) [MSC v.1900 32 bit (Intel)]
 on win32
Type "copyright", "credits" or "license()" for more information.
>>> print("hello world")
hello world
>>> print 2*10
SyntaxError: Missing parentheses in call to 'print'
>>> print(2*10)
20
>>> def hello(parameter):
        print("hello "+parameter)


>>> hello("world")
hello world
>>>
```

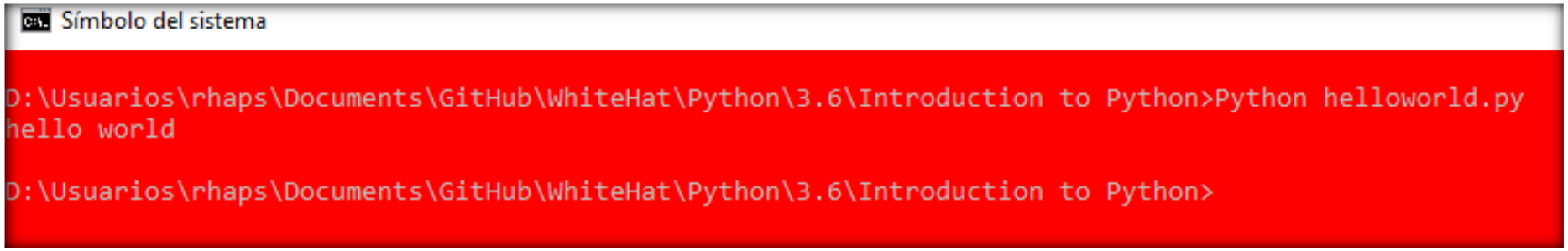
# PYTHON - VERSIONES

```
Símbolo del sistema
Microsoft Windows [Versión 10.0.14393]
(c) 2016 Microsoft Corporation. Todos los derechos reservados.

D:\Usuarios\rhaps>Python -V
Python 3.5.2

D:\Usuarios\rhaps>
```

# EJECUTE UN SCRIPT PYTHON




# EJECUTE UN SCRIPT PYTHON

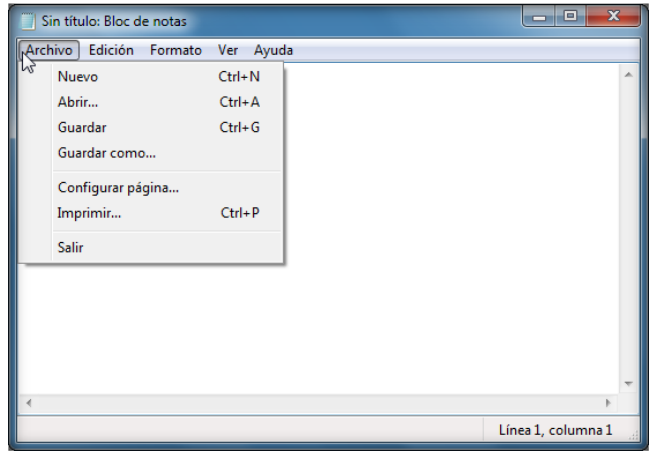
- Podemos almacenar las salidas de un script en un nuevo archivo

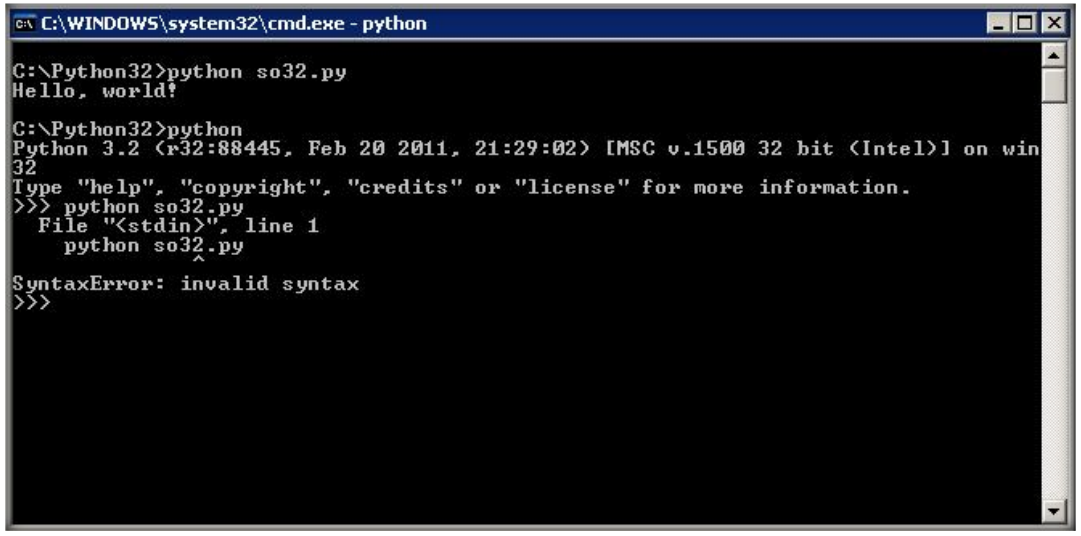
```
Python helloworld.py > output.txt
```

# PYTHON


PC
PyCharm
Python IDE
for Professional Developers


Visual Studio
TM


Sin título: Bloc de notas
Archivo
Edición
Formato
Ver
Ayuda
Nuevo Ctrl+N
Abrir... Ctrl+A
Guardar Ctrl+G
Guardar como...
Configurar página...
Imprimir... Ctrl+P
Salir
Línea 1, columna 1


C:\WINDOWS\system32\cmd.exe - python
C:\Python32>python so32.py
Hello, world!
C:\Python32>python
Python 3.2 (r32:88445, Feb 20 2011, 21:29:02) [MSC v.1500 32 bit (Intel)] on win
32
Type "help", "copyright", "credits" or "license" for more information.
>>> python so32.py
File "<stdin>", line 1
python so32.py
^
SyntaxError: invalid syntax
>>>

# ESTRUCTURAS

- Todo lo que procesamos en Python es considerado como objeto

*Table 4-1. Built-in objects preview*

| Object type | Example literals/creation |
|---|---|
| Numbers | `1234, 3.1415, 3+4j, 0b111, Decimal(), Fraction()` |
| Strings | `'spam', "Bob's", b'a\x01c', u'sp\xc4m'` |
| Lists | `[1, [2, 'three'], 4.5], list(range(10))` |
| Dictionaries | `{'food': 'spam', 'taste': 'yum'}, dict(hours=10)` |
| Tuples | `(1, 'spam', 4, 'U'), tuple('spam'), namedtuple` |
| Files | `open('eggs.txt'), open(r'C:\ham.bin', 'wb')` |
| Sets | `set('abc'), {'a', 'b', 'c'}` |
| Other core types | Booleans, types, None |
| Program unit types | Functions, modules, classes (Part IV, Part V, Part VI) |
| Implementation-related types | Compiled code, stack tracebacks (Part IV, Part VII) |

# ADVERTENCIA

PYThON ES sensible A LAS MAYUSCULAS Y minúsculas

# VARIABLES

- Declaración de variables.

number = 100

factor = 1.1

text =  "hello world“

Com = 1j

- Utilice el método type e imprima sus resultados.

# IMPORTANDO PAQUETES Y AYUDA

- Dado el caso que el paquete solo traiga métodos y constantes podemos utilizar **import** más el nombre de la herramienta.
- Importemos el paquete math

```
import math
```

- Ahora utilice la ayuda y obtenga la documentación de math

```
?math
```

```
help(math)
```

- Ahora busque entre las constantes e imprima el número pi

# IMPORTANDO PAQUETES Y AYUDA

1. Importe el paquete **random**, busque en la documentación y genere un número aleatorio
2. Utilice el método **choice()** de random e ingrésele por parámetro una lista, es decir:

[1, 4, 9]

# IMPORTANDO PAQUETES Y AYUDA

- Explore el contenido del archivo my_tools.py
- La forma de utilizar un paquete que tiene clases y métodos es a través de **from,** luego importamos la clase que necesitamos con **import**.

**import** numpy **as** np

**from** my_tools **import** Tools **as** tl

# CARACTERES

- Podemos declarar una cadena de caracteres con “” o ‘’, por ejemplo:

```
s = "spam"
print(type(s))
s = 'spam'
print(type(s))
```

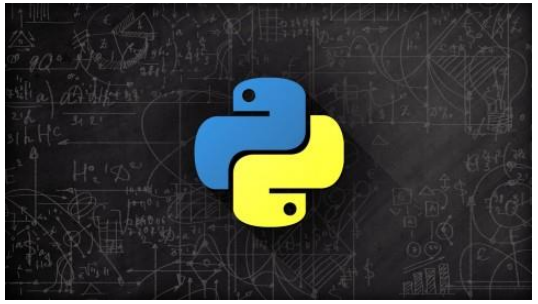
- Podemos obtener la cantidad de caracteres de una cadena con el método **len()**

```
len(s)
```

# CARACTERES - RECORRIDOS

- Digite e imprima los resultados de las siguientes líneas de código:

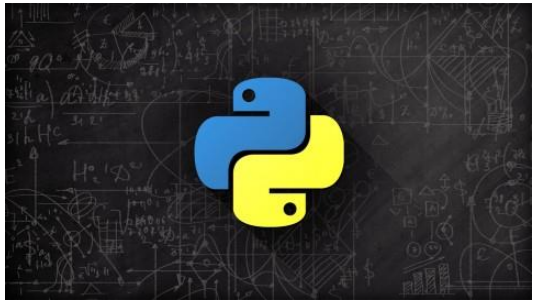
```
s = "spam"
s[0]
s[-1]
s[1:3] # Slicing
s[0] = "z"
```



# MÉTODOS DE LA CLASE STR

- Utilice la ayuda de str y aplique los métodos que permitan realizar las siguientes operaciones sobre la cadena de caracteres **“hello world”**:

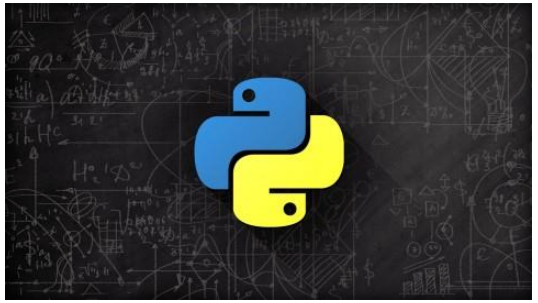
1. Retorne el primer índice de la cadena “ world”.
2. Reemplazar la palabra “world” por “Python”.
3. Transformar “hello Python” a mayúsculas.



# MÉTODOS DE LA CLASE STR

- Utilice la ayuda de str y aplique los métodos que permitan realizar las siguientes operaciones sobre la cadena de caracteres **“aaa,bbb,ccccc,dd”**:

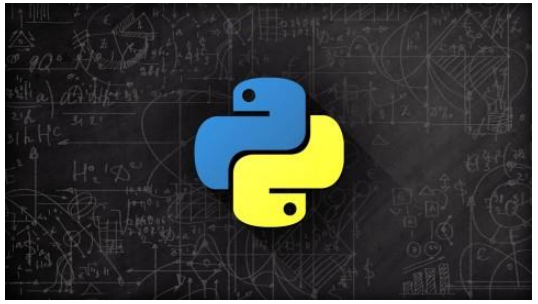
1. Dividir el carácter por el símbolo “,” y que cree una lista con sus elementos.



# FORMATEANDO UNA EXPRESIÓN

- Podemos imprimir los valores de un conjunto de variables en conjunto con un texto.

```
spam = "spam"
SPAM = "SPAM"
spam + "," + "eggs, and "+ SPAM
```

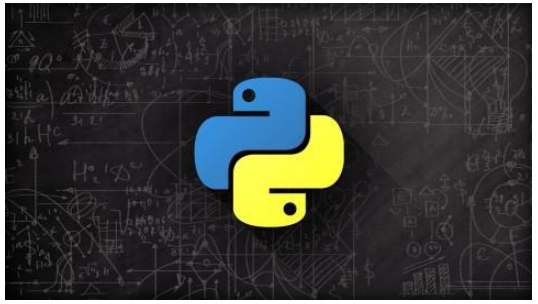


# FORMATEANDO UNA EXPRESIÓN

- Podemos imprimir los valores de un conjunto de variables en conjunto con un texto.

```
"%s, eggs, and %s" % ('spam', "SPAM")
```

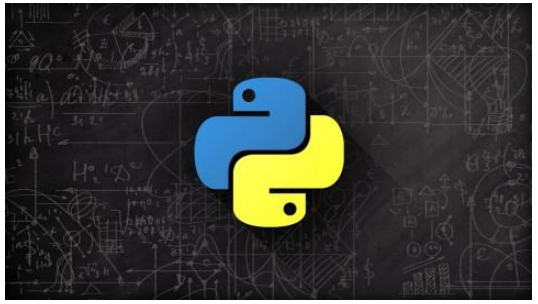
Tupla de valores a imprimir



# FORMATEANDO UNA EXPRESIÓN

- Podemos imprimir los valores de un conjunto de variables en conjunto con un texto.

"**{}**, eggs, and **{}**".**format('spam', "SPAM")**



# LISTAS

- Las listas las definimos con [], son estructuras que permiten almacenar durante el tiempo de vida del Kernel un conjunto de datos de distinto tipo.

lista = [1, 4, 5, “hello”, 7]

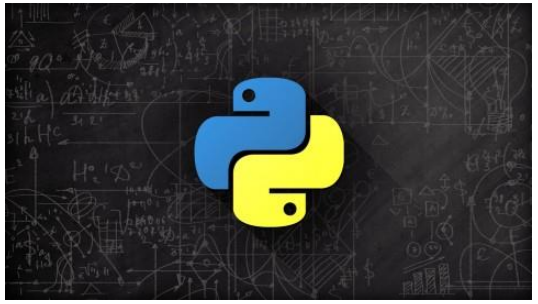
- Revise la documentación de la clase lista y realice las siguientes operaciones:

1. Ingrese el valor 21 a la lista [1, 4, 5, “hello”, 7].
2. Organice la lista [10, 2, 4, 1] de manera ascendiente.
3. Remueva el valor 4 de la lista [1, 4, 5, “hello”, 7].

# DICCIONARIOS

- Son estructuras hash table en Python que nos permiten almenar por tiempo del vida del kernel datos asociados a unas llaves.

```
# the next code line makes a dictionary
services = {'ftp':21, 'ssh':22, 'smtp':25, 'http':80}
# prints the keys
print(services.keys())
print()
# prints the items
print(services.items())
print()
# prints a value associated to a key
print(services['ftp'])
```



# FECHAS

**datetime**, Python tiene un paquete dedicado a tareas relacionadas con fechas, algunos de sus métodos son:

- datetime.date
- datetime.datetime
- datetime.MAXYEAR
- datetime.MINYEAR
- datetime.year

- https://docs.python.org/3.7/library/datetime.html

# OPERADORES

- **Multiplication**, in Python it's representation is *
- **Sum**, in Python it's representation is +
- **substract**, a - b
- **divide**, a / b
- **floor divide**, a // b
- **raise**, a ** b
- **AND**, a & b
- **OR**, a | b
- **EQUAL**, a == b
- **Difference**, a != b
- **XOR**, a ^ b

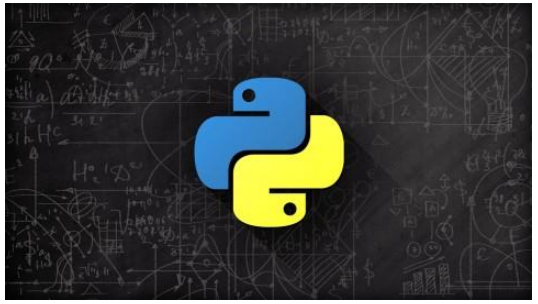
# OPERADORES

Existen operadores aritméticos y asociativos en Python.

- **in** es un palabra reservada que significa una operación que retorna si un objeto aparece en otro, es decir…

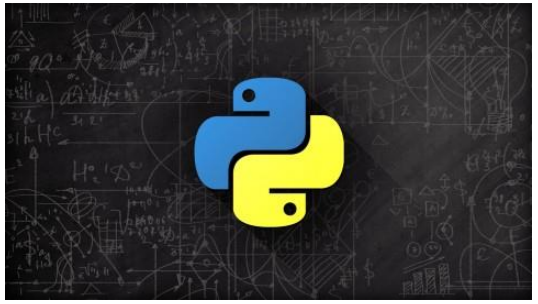
"world" **in** "hello world“

[2,3] **in** [1,[2,3],3,4]



# FUNCIONES

- Python has a lot of packages and different functions, one important source to get their documentation is at Python's docs (https://docs.python.org/3/library/). In this section we took some functions in order to illustrate how use them in Python.
- type(), it allows us to know the type of the variables
- str(), int(), float() and bool(), these ae the mechanisms which allow us to change a variable type to another one.



# FUNCIONES

- Preste atención a como se define una función.
- Un método siempre inicia con la palabra reserve **def** y luego se define su nombre.
- Luego del nombre de van los parámetros dentro de parenthesis y separados por comas.
- Una vez definidos los parámetros finalizamos con un **:**
- Toda línea de código que vaya dentro del método debe tener una sangría

```
def append_alement(some_list, element):
    some_list.append(element)
```

# FUNCIONES

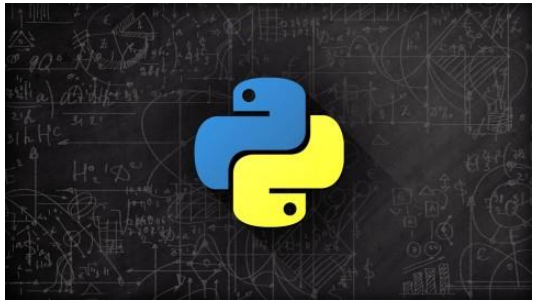
```
# we can return a value or multiple values with our function
   def k():
      a = 1
      b = 3
      c = 7
      return a, b, c

   a, b, c = k()
   print(a)
```

# REPETITIVAS

```
result = 0
for i in range(100):
    result +=i
print(result)
```

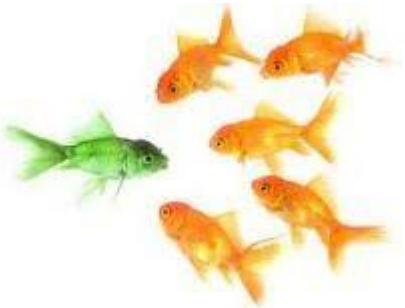
```
animals = ["dog", "cat", "mouse"]
for i in animals:
    print(i, end="/")
print("\n")
for i in animals:
    print(i)
```



# EXCEPCIONES

Ejecute la siguiente sentencia

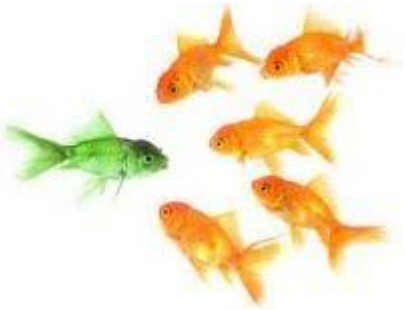
131/0



# EXCEPCIONES

Podemos hacer control de excepciones de la siguiente forma

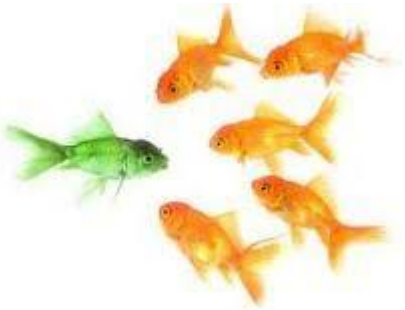
```
try:
    print("[+] 1337/0 = " + str(1337/0))
except:
    print("[-] Error. ")
```



# EXCEPCIONES

Podemos también acceder a la información de la excepción.

```
try:
    print("[+] 1337/0 = " + str(1337/0))
except Exception as e:
    print("[-] Error = " + str(e))
```



# CLASES

```python
from numpy import exp, sum


class Tools(object):

    def softmax(x):
        """
        Compute the softmax values for x.
        Returns
        -------
        probabilities : the probability representation of x
        """
        """""""""
        return exp(x) / sum(exp(x), axis=0)

    def fibonacci(self=None):
        """
        this method returns the first ten numbers of the Fibonacci
        Returns
        -------
        fibonacci : the ten numbers of the fibonacci method
        """
        """""""""
        a, b = 0, 1
        while a < 10:
            print(a)
            a, b = b, a + b
```

# INPUT

- Podemos solicitar ingreso de datos por consola a los usuarios a través del método **input**

```
print(input("Escriba su nombre: "))
```

# Beginner's Python Cheat Sheet

## Variables and Strings

*Variables are used to store values. A string is a series of characters, surrounded by single or double quotes.*

Hello world

```
print("Hello world!")
```

Hello world with a variable

```
msg = "Hello world!"
print(msg)
```

Concatenation (combining strings)

```
first_name = 'albert'
last_name = 'einstein'
full_name = first_name + ' ' + last_name
print(full_name)
```

## Lists

*A list stores a series of items in a particular order. You access items using an index, or within a loop.*

Make a list

```
bikes = ['trek', 'redline', 'giant']
```

Get the first item in a list

```
first_bike = bikes[0]
```

Get the last item in a list

```
last_bike = bikes[-1]
```

Looping through a list

```
for bike in bikes:
    print(bike)
```

Adding items to a list

```
bikes = []
bikes.append('trek')
bikes.append('redline')
bikes.append('giant')
```

Making numerical lists

```
squares = []
for x in range(1, 11):
    squares.append(x**2)
```

## Lists (cont.)

List comprehensions

```
squares = [x**2 for x in range(1, 11)]
```

Slicing a list

```
finishers = ['sam', 'bob', 'ada', 'bea']
first_two = finishers[:2]
```

Copying a list

```
copy_of_bikes = bikes[:]
```

## Tuples

*Tuples are similar to lists, but the items in a tuple can't be modified.*

Making a tuple

```
dimensions = (1920, 1080)
```

## If statements

*If statements are used to test for particular conditions and respond appropriately.*

Conditional tests

```
equals            x == 42
not equal         x != 42
greater than      x > 42
  or equal to     x >= 42
less than         x < 42
  or equal to     x <= 42
```

Conditional test with lists

```
'trek' in bikes
'surly' not in bikes
```

Assigning boolean values

```
game_active = True
can_edit = False
```

A simple if test

```
if age >= 18:
    print("You can vote!")
```

If-elif-else statements

```
if age < 4:
    ticket_price = 0
elif age < 18:
    ticket price = 10
else:
    ticket_price = 15
```

## Dictionaries

*Dictionaries store connections between pieces of information. Each item in a dictionary is a key-value pair.*

A simple dictionary

```
alien = {'color': 'green', 'points': 5}
```

Accessing a value

```
print("The alien's color is " + alien['color'])
```

Adding a new key-value pair

```
alien['x_position'] = 0
```

Looping through all key-value pairs

```
fav_numbers = {'eric': 17, 'ever': 4}
for name, number in fav_numbers.items():
    print(name + ' loves ' + str(number))
```

Looping through all keys

```
fav_numbers = {'eric': 17, 'ever': 4}
for name in fav_numbers.keys():
    print(name + ' loves a number')
```

Looping through all the values

```
fav_numbers = {'eric': 17, 'ever': 4}
for number in fav_numbers.values():
    print(str(number) + ' is a favorite')
```

## User input

*Your programs can prompt the user for input. All input is stored as a string.*

Prompting for a value

```
name = input("What's your name? ")
print("Hello, " + name + "!")
```

Prompting for numerical input

```
age = input("How old are you? ")
age = int(age)

pi = input("What's the value of pi? ")
pi = float(pi)
```

# BIBLIOGRAFÍA


- Lutz, M. (2013). *Learning Python: Powerful Object-Oriented Programming*." O'Reilly Media, Inc.".