UNIVERSIDAD
NACIONAL
DE COLOMBIA

PROYECTO **CULTURAL, CIENTÍFICO Y COLECTIVO** DE NACIÓN

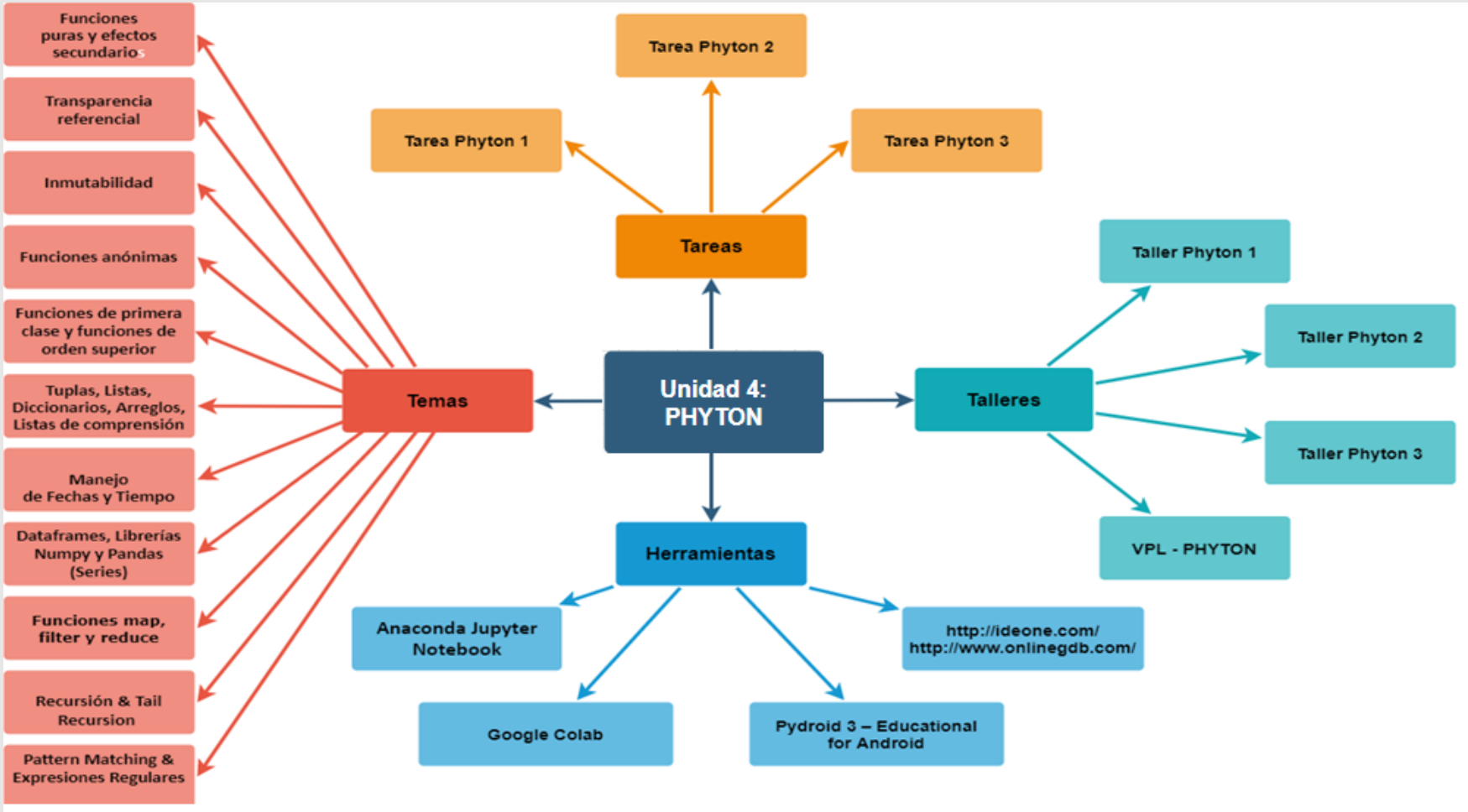
Phyton

Prof. Oscar Mauricio Salazar Ospina
omsalazaro@unal.edu.co

Facultad de Minas
Departamento de Ciencias de la Computación y la Decisión
Facultad de Minas

Unidad 3: Scala
Mapa conceptual

Kahoot! 1



Scala

Generalidades

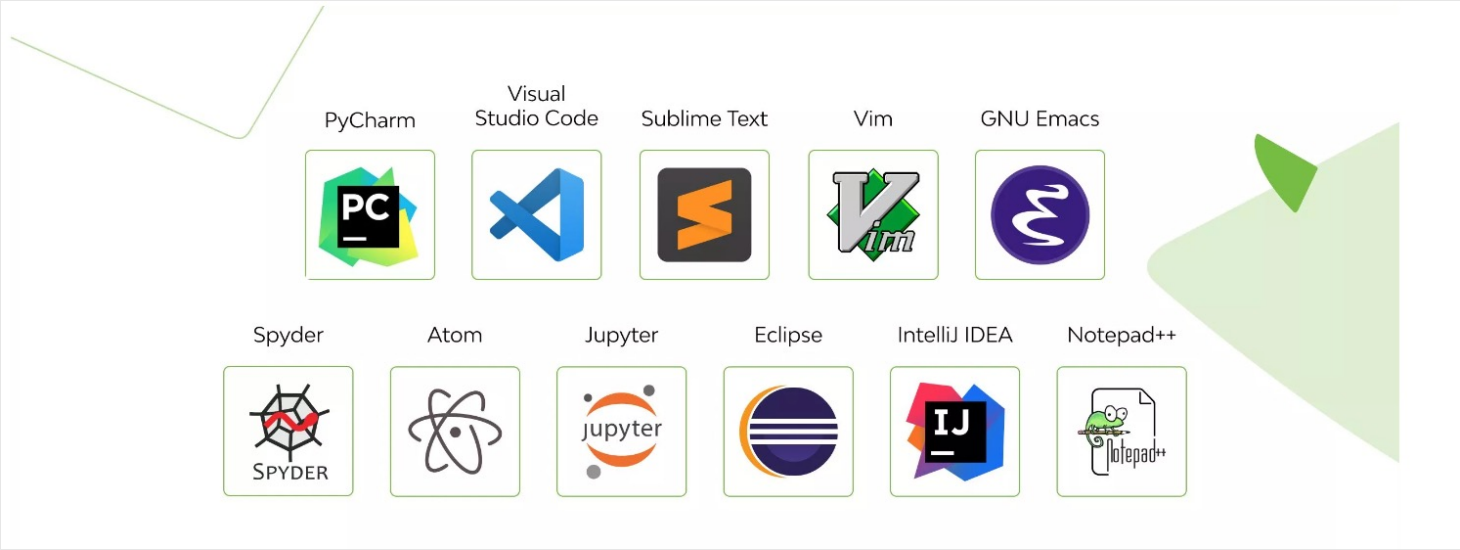
Python es un lenguaje de programación de código abierto el cual se desarrolló en el año 1991, por un equipo de desarrolladores dirigidos por **Guido van Rossum** en Países Bajos.

Van Rossum bautizó al lenguaje de programación basándose en su gran afición por el grupo de comedia británico llamado **Monty Python.**

¿Sabías que Instagram, Youtube, Google, Dropbox, Facebook, Netflix y hasta la Nasa utilizan **Python**?

Scala

Herramientas de desarrollo (IDE)



ONLINE PYTHON

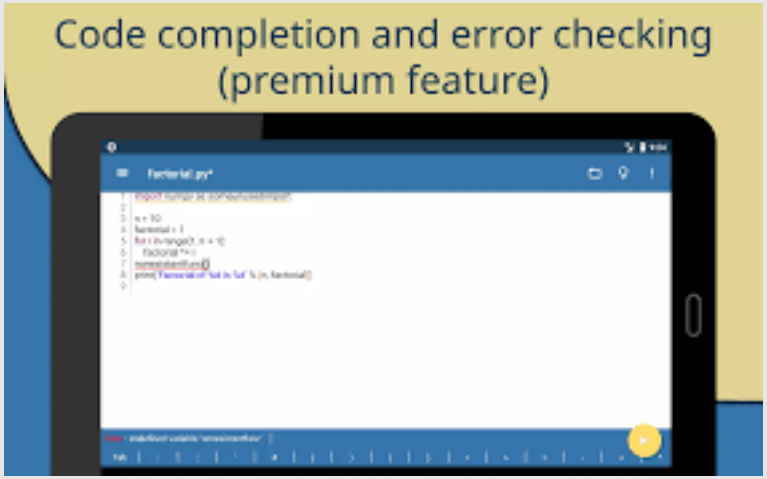
replit

python + colab

OnlineGDB beta

Scala

Herramientas de desarrollo (IDE)



Phyton

Paradigma de programación

Kahoot! 2

Modelo declarativo funcional

- Funciones **puras**, Funciones **anónimas**, Funciones de **primera clase** y de **orden superior**.
- **Transparencia referencial, Inmutabilidad, Recursión** y **Tail recursión**, **Pattern Matching**.

Modelo Orientado a Objetos (POO)

- La característica de describir clases de objetos brinda a los programadores **otra forma de organizar su código en tareas o funciones** que están relacionadas con un tipo particular de objeto.
- **Clase, Objetos, Atributos, Métodos, Instancias.**

Modelo imperativo

- El proceso de subdividir **programas en subprogramas** (piezas más simples) se llama programación estructurada o modelo imperativo de programación.
- Algoritmo: **Datos + Procedimientos (secuencia de instrucciones)**

Phyton

Lenguaje interpretado

Un lenguaje interpretado se caracteriza por **ser convertido a un lenguaje de máquina** a **medida que es ejecutado** (ejm. Ruby, Python y JavaScript)

```
In [2]: print("Hola Mundo")
        print('letras de un texto')
        for i in "TEXTO":
            print(i)
            i = 1
        print(logrado)
        println('FIN')
```

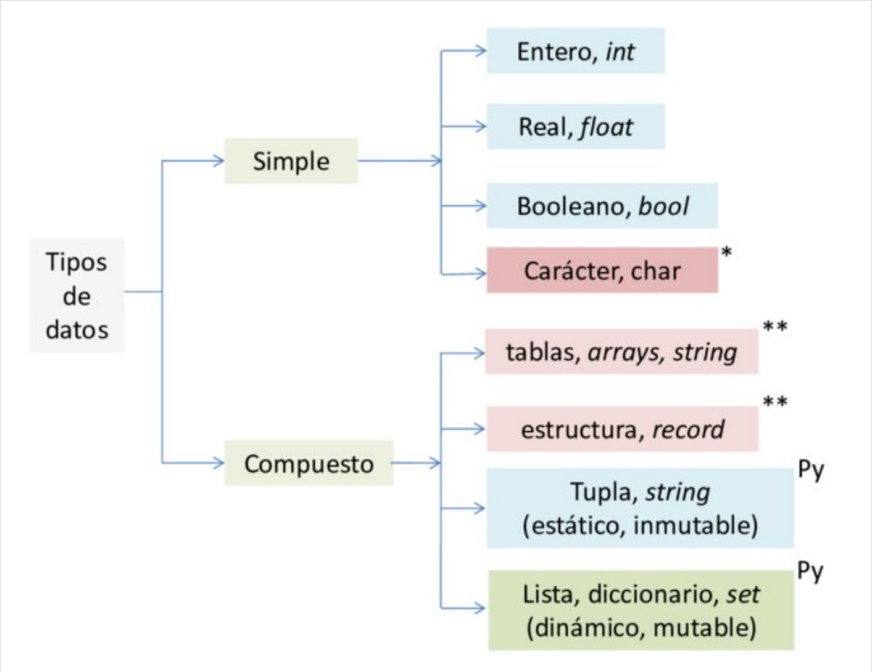
```
Hola Mundo
letras de un texto
T
E
X
T
O

---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-2-b35dda80017a> in <module>
      4     print(i)
      5     i = 1
----> 6 print(logrado)
      7 println('FIN')

NameError: name 'logrado' is not defined
```

Phyton

Tipos de datos



***** El tipo de dato carácter no existe en Python, un carácter simple se representa como cadena de caracteres (string).

****** Estructuras compuestas de lenguajes como C, FORTRAN, Pascal, Matlab, etc.

Py: Estructuras compuestas en Python.

Fuente: Pedro Gomis

Phyton

Tipado

Tipado dinámico

- Es aquel que realiza **la verificación del tipo de las variables durante la ejecución**, al encontrarse un error de tipos durante la ejecución/evaluación, el lenguaje lo detecta y modela (o lo infiere de acuerdo a la operación). Ejm. Python, Ruby.

Tipado fuerte

- Un lenguaje de programación es fuertemente tipado si **no se permiten violaciones de los tipos de datos**, es decir, dado el valor de una variable de un tipo concreto, no se puede usar como si fuera de otro tipo distinto a menos que se haga una conversión.
  Ejm. Python y Java.

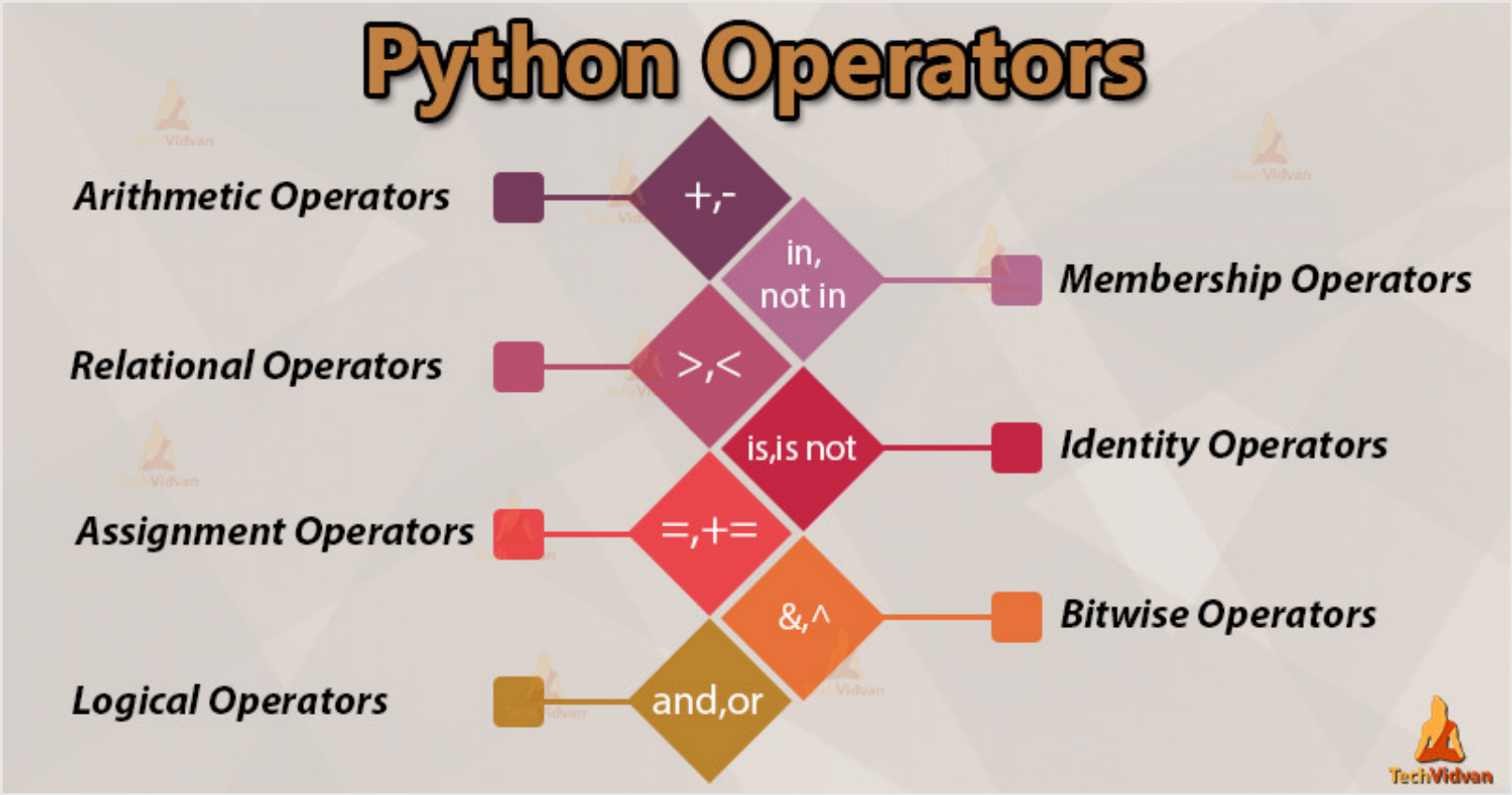
Tipado implícito vs explícito

- Es aquel en el que no se está obligado (implícito) o sí (explícito) a **proveer anotaciones de tipos**, por ejemplo para las variables, parámetros y valores de retorno de un método/ procedimiento / función.
- Se pueden utilizar ambos en Python, aunque según la filosofía de Guido van Rossum (su creador) **"Explícito es mejor que implícito", lo cual es clave pues contribuye a la mantenibilidad del código.**

Phyton

Operadores

Kahoot! 5



Phyton

Operadores

1. Funciones puras y efectos secundarios
2. Transparencia referencial
3. Funciones anónimas
4. Inmutabilidad
5. Funciones de primera clase y funciones de orden superior
6. Recursión y Tail recursión
7. Pattern Matching

Phyton

Funciones anónimas

```scala
val impar = (numero: Int) => numero % 2 != 0
println(impar(5)) // retorna => true

//celsius a fahrenheit
val fahrenheit = (c: Double) => c * 1.8 + 32
println(fahrenheit(30)) // retorna 86.0
```

Scala

```python
impar = lambda numero: numero % 2 != 0
print(impar(5))  # retorna true

fahrenheit = lambda c: c * 1.8 + 32
print(fahrenheit(30))  # retorna 86.0
```

Phyton

Funciones anónimas

```
(c: Double) => {
  c * 1.8 + 32
}
```

- Las **funciones anónimas (FnA)** se implementan en **Python** con las funciones o **expresiones lambda**.

En **Python** el contenido de una **FnA** debe ser **una única expresión** en lugar de un bloque de acciones.

Phyton

Tipado dinámico

```python
numero1 = 6 * 7
print(numero1)  # 42
print(type(numero1))  # <class 'int'>

numero1 = 8 * 9.5
print(numero1)  # 76.0
print(type(numero1)) # <class 'float'>


numero1 = "numero1"
print(numero1)  # numero1
print(type(numero1))  # <class 'str'>
```

La variable numero1 cambia de tipo de **entero** a **float** y luego a **string** durante la ejecución del programa

Phyton

Inmutabilidad

Kahoot! 6

```
x = 8
y = x
print(id(x))
print(id(y))
x = 10
print(id(x))
print(id(y))
y = 1000
print(id(x))
print(id(y))
```

Los números son inmutables

Las listas son mutables

```
x = [1, 2, 3, 4]
y = x
print(id(x))
print(id(y))
x[0] = 5
print(id(x))
print(id(y))
```

Phyton

Inmutabilidad

```
L = [('Santiago', 20), ('Juan', 24), ('Claudia', 21)]
for idx, val in enumerate(L):
  nombre = val[0]
  edad = val[1]
  print("El indice es %d, nombre es %s, y edad es %d" \
        % (idx, nombre, edad))
print(L)
print(L[0][0])
L[0][0] = "Sebastian"
```

```
El indice es 0, nombre es Santiago, y edad es 20
El indice es 1, nombre es Juan, y edad es 24
El indice es 2, nombre es Claudia, y edad es 21
[('Santiago', 20), ('Juan', 24), ('Claudia', 21)]
Santiago
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-19-81448caa8d01> in <module>()
      6 print(L)
      7 print(L[0][0])
----> 8 L[0][0] = "Sebastian"

TypeError: 'tuple' object does not support item assignment
```

Las tuplas
son inmutables

Phyton

Composición

Kahoot! 8,9,10,11

```python
def p(algo): return print(algo)


def multiplo_seis(f, g, i): return f(i) & g(i)


multiplo_dos = lambda i: (i % 2 == 0)
multiplo_tres = lambda i: (i % 3 == 0)
p(multiplo_dos(6))  # True
p(multiplo_tres(6))  # True
p(multiplo_seis(multiplo_dos, multiplo_tres, 6))  # True
```

Phyton

Map

Scala

```scala
val lista = List(1, 2, 3, 4)
println(lista.map(_ * 3))
```

```python
dominio = range(1, 11)
print(list(dominio))  # [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
print(list(map(lambda elemento: elemento * 3, list(dominio))))
```

Phyton

Otras funciones de orden superior

```
dominio = range(1, 11)
print(list(filter(lambda elemento: elemento in [3, 50, 6], list(dominio))))
```

```
print(24 in dominio)
```

```
find = lambda fun, lst: next((x for x in lst if fun(x)), None)
print(find(lambda x: x % 5 == 0, dominio))

findall = lambda fun, lst: [x for x in lst if fun(x)]
print(findall(lambda x: x % 5 == 0, dominio))
```

Phyton

Enumerate - Interpolación

```python
estudiantes = [("Luis", 3.5), ("Pedro", 5.0), ("Diego", 0.6)]

for indice, estudiante in enumerate(estudiantes):
    print('Estudiante %d: %s, nota: %f' % (indice + 1, estudiante[0], estudiante[1]))
```

Phyton

Sorted

```
numeros = [1, 5, 4, 3, 8, 6, 9, 4, 3]
print(sorted(numeros))
```

```
estudiantes = [("Ana", 3.5), ("Pedro", 5.0), ("Diego", 0.6)]
print(sorted(estudiantes))
print(sorted(estudiantes, key=lambda estudiante: estudiante[1]))
print(sorted(estudiantes, key=lambda estudiante: estudiante[1], reverse=True))
```

Phyton

Sorted

```
numeros = [1, 5, 4, 3, 8, 6, 9, 4, 3]
print(sorted(numeros))
```

```
estudiantes = [("Ana", 3.5), ("Pedro", 5.0), ("Diego", 0.6)]
print(sorted(estudiantes))
print(sorted(estudiantes, key=lambda estudiante: estudiante[1]))
print(sorted(estudiantes, key=lambda estudiante: estudiante[1], reverse=True))
```

```
[1, 3, 3, 4, 4, 5, 6, 8, 9]
[('Ana', 3.5), ('Diego', 0.6), ('Pedro', 5.0)]
[('Diego', 0.6), ('Ana', 3.5), ('Pedro', 5.0)]
[('Pedro', 5.0), ('Ana', 3.5), ('Diego', 0.6)]
```

Phyton

Sort vs sorted

Kahoot! 12

```python
estudiantes = [("Luis", 3.5), ("Pedro", 5.0), ("Diego", 0.6)]
numeros = [1, 5, 3, 8, 5, 33, 6, 8]
tupla = (2, 1, 5, 3)

estudiantes.sort()
numeros.sort()

print(estudiantes)
print(numeros)
print(sorted(tupla))
tupla.sort()
print(tupla)
```

```
Traceback (most recent call last):
  File "/Users/omsalazaro/PycharmProjects/pythonProject/error.py", line 12, in <module>
    tupla.sort()
AttributeError: 'tuple' object has no attribute 'sort'
[('Diego', 0.6), ('Luis', 3.5), ('Pedro', 5.0)]
[1, 3, 5, 5, 6, 8, 8, 33]
[1, 2, 3, 5]

Process finished with exit code 1
```

Phyton

Reduce

```python
from functools import reduce

numeros = [1, 5, 3, 8, 5, 33, 6, 8]

print(reduce(min, numeros))
print(reduce(max, numeros))
```

Phyton

Reduce

```python
from functools import reduce

numeros = [1, 5, 3, 8, 5, 33, 6, 8]

print(reduce(min, numeros))
print(reduce(max, numeros))
```

```
1
33
```

Phyton

Operaciones con listas

```
>>> factura = ['pan', 'huevos', 100, 1234]
>>> factura
['pan', 'huevos', 100, 1234]
```

```
>>> factura[0]
'pan'
>>> factura[3]
1234
```

```
>>> len(factura)
4
```

```
>>> factura[-1]
1234
```

```
>>> versiones_plone = [2.1, 2.5, 3.6, 4, 5, 6, 4]
>>> print versiones_plone.index(4)
3
```

Phyton

Operaciones con listas

```
>>> versiones_plone = [2.1, 2.5, 3.6]
>>> print versiones_plone
[2.1, 2.5, 3.6]
>>> versiones_plone.extend([4])
>>> print versiones_plone
[2.1, 2.5, 3.6, 4]
>>> versiones_plone.extend(range(5,7))
>>> print versiones_plone
[2.1, 2.5, 3.6, 4, 5, 6]
```

```
>>> versiones_plone = [2.1, 2.5, 3.6, 4, 5, 6]
>>> print "6 ->", versiones_plone.count(6)
6 -> 1
>>> print "5 ->", versiones_plone.count(5)
5 -> 1
>>> print "2.5 ->", versiones_plone.count(2.5)
2.5 -> 1
```

Phyton

Operaciones con listas

```
>>> versiones_plone = [2.1, 2.5, 3.6, 4, 5, 6]
>>> print versiones_plone
[2.1, 2.5, 3.6, 4, 5, 6]
>>> versiones_plone.insert(2, 3.7)
>>> print versiones_plone
[2.1, 2.5, 3.7, 3.6, 4, 5, 6]
```

```
>>> versiones_plone = [2.1, 2.5, 3.6, 4, 5, 6]
>>> print versiones_plone.pop()
6
>>> print versiones_plone
[2.1, 2.5, 3.6, 4, 5]
```

```
>>> versiones_plone = [2.1, 2.5, 3.6, 4, 5, 6]
>>> print versiones_plone
[2.1, 2.5, 3.6, 4, 5, 6]
>>> versiones_plone.remove(2.5)
>>> print versiones_plone
[2.1, 3.6, 4, 5, 6]
```

Phyton

Operaciones con listas

Kahoot! 13, 14, 15, 16, 17

```
factura = ['pan', 'huevos', 100, 1234, "supermercado"]

print(factura[2:4])
print(factura[1:])
print(factura[-3:])
print(factura[-3:-2])
```

Gracias