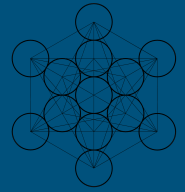
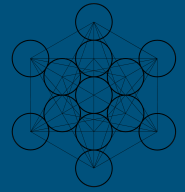

python™

Temas a desarrollar



1. Introducción al Lenguaje de Programación
2. Instalación
3. "Hola Mundo"
    -Entrada Estándar rawInput
    -Salida Estándar rawInput
4. Enteros, reales y operadores aritméticos
5. Booleanos, operadores lógicos y cadenas
6. Listas
7. Tuplas
8. Diccionarios
9. Operadores relacionales
10. Sentencias condicionales
11. Bucles
12. Funciones
13. Clases y Objetos
14. Herencia
15. Herencia múltiple
16. Cadenas y métodos
17. Listas y sus métodos
18. Diccionarios y sus métodos
19. Encapsulación



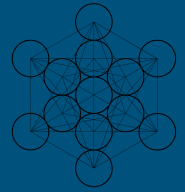
Características

- Lenguaje interpretado
- Tipado dinámicamente
- Fuertemente tipado
- Multiplataforma
- OO

Hola Mundo

Interprete.

```
print(“ ”)
print(“ Hola Mundo”)
x=”Hola Mundo”
x=5
x=5L
```



Entrada/Salida Estándar

```
try:
        valor=raw_input(“Introduce un número: ”)
        valor=int(valor)
except:
        print(“Esto no es un número”)
else:
        print(valor)
```

Entrada/Salida Estándar

```
>>> x=4
>>> y="Hola mundo"
>>> print("mi edad es: %s algo mas"%x)
mi edad es: 4 algo mas
>>> z=3.145678
>>> print("mi edad es: %f algo mas"%z)
mi edad es: 3.145678 algo mas
>>> print("mi edad es: %.3f algo mas"%z)
mi edad es: 3.146 algo mas
>>> print("mi edad es: %d algo mas"%x)
mi edad es: 4 algo mas
```

Entrada/Salida Estándar

```
>>> s = 'Hola mundo.'
>>> str(s)
'Hola mundo.'
>>> repr(s)
"'Hola mundo.'"
>>> str(1.0/7.0)
'0.142857142857'
>>> repr(1.0/7.0)
'0.14285714285714285'
>>> x = 10 * 3.25
>>> y = 200 * 200
```

```
>>> s = 'El valor de x es ' + repr(x) + ', y es ' +
repr(y) + '...'
>>> print s
El valor de x es 32.5, y es 40000...
>>> # El repr() de una cadena agrega apóstrofos
y barras invertidas
... hola = 'hola mundo\n'
>>> holas = repr(hola)
>>> print holas
'hola mundo\n'
```

Entrada/Salida Estándar

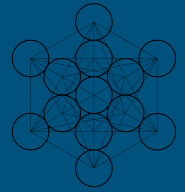
```
>>> # El argumento de repr() puede ser
cualquier objeto Python:
... repr((x, y, ('carne', 'huevos')))
"(32.5, 40000, ('carne', 'huevos'))"

for x in range(1, 35):
        print (repr(x).rjust(5), repr(x*x).rjust(5),
        (repr(x*x*x).rjust(5)))
```

el método rjust() de los objetos cadena llena de espacios a la izquierda, Hay métodos similares ljust() y center()

Entrada/Salida Estándar

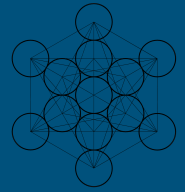
```
>>> '12'.zfill(5)
'00012'
>>> '-3.14'.zfill(7)
'-003.14'
>>> '3.14159265359'.zfill(5)
'3.14159265359'
>>> '3.14'.zfill(5)
'03.14'
```

el método zfill(), el cual rellena una cadena numérica a la izquierda con ceros



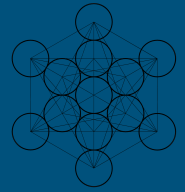
Entrada/Salida Estándar

```
>>> print 'Somos los {} quienes decimos "{}!"'.format('caballeros', 'Nop')
Somos los caballeros quienes decimos "Nop!"
>>> print '{0} y {1}'.format('carne', 'huevos')
carne y huevos
>>> print '{1} y {0}'.format('carne', 'huevos')
huevos y carne
```



```
name = input("What's your name? ")
print("Nice to meet you " + name + "!")
age = input("Your age? ")
print("So, you are already " + str(age) + " years old, " + name + "!")
```

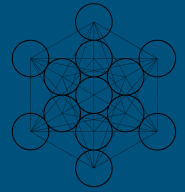
Números Enteros, Reales y Operadores aritméticos



Números y operadores

```
>>> 2 + 2
4
>>> 50 - 5*6
20
>>> (50 - 5*6) / 4
5.0
>>> 8 / 5  # la división regresa un float
1.6
```

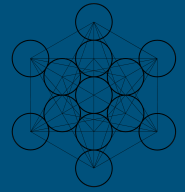
```
>>> 17 / 3  # clásica división
5.666666666666667
>>> 17 // 3  # división descartando la fracción
5
>>> 17 % 3  # el % regresa el residuo
2
>>> 5 * 3 + 2  # mantiene jerarquía de operadores
17
```



```
>>> 5 ** 2  # 5 cuadrado
25
>>> 2 ** 7  # 2 a la potencia 7
128
```

reales - x10 exponente n

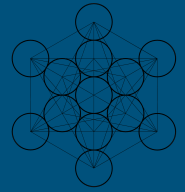
```
>>> real=0.673
>>> real=0.67e-3
>>> print (real)
0.00067
```



Boleanos, Operadores Lógicos y Cadenas

Operadores Lógicos y Valores Boleans

```
>>> xF=False
>>> xT=True
>>> bAND=xT and xF
>>> bOR=xT or xF
>>> bNOT = not xT
>>> print (bAND)
False
>>> print(bOR)
True
>>> print(bNOT)
False
```



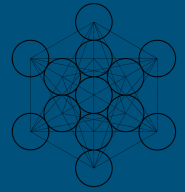
Cadenas

```
a=’Hola ’
b=”Mundo”
c=”””” Cruel
Saludos
a
Todos
”””
```

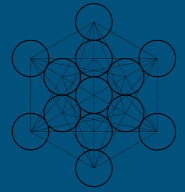
Concatenar y Repetir

```
>>> x= “texto ” *  3'
>>> print (x)
texto texto texto
```

```
>>> print (a,b,c)
Hola  Mundo  Cruel
Saludos
a
Todos
>>> print (a+b+c)
Hola Mundo Cruel
Saludos
a
Todos
```



```
word = 'Python'
>>> word[0]  # caracter en posición 0
'P'
>>> word[5]  # caracter en posición 5
'n'

>>> text = ('Put several strings within parentheses '
...         'to have them joined together.')
>>> text
'Put several strings within parentheses to have them joined together.'
```



función len()

```
>>> s = 'supercalifragilisticexpialidocious'
>>> len(s)
34
```

Listas

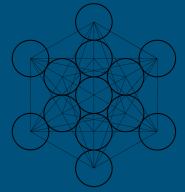
```
>>> l=[2,"tres",True,["uno",10]]
>>> squares = [1, 4, 9, 16, 25]
>>> squares
[1, 4, 9, 16, 25]

>>> squares[0]
1
>>> squares[-1]
25
>>> squares[-3:]
[9, 16, 25]
```

concatenación

```
>>> squares + [36, 49, 64, 81, 100]
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

elemento “65” esta equivocado

```
>>> cubes = [1, 8, 27, 65, 125]
>>> 4 ** 3  #es 64, no 65!
64
>>> cubes[3] = 64  # reemplazando...
>>> cubes
[1, 8, 27, 64, 125]
```

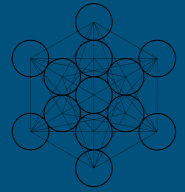


Función append()

```
>>> cubes.append(216)  # agregando el cubo de 6
>>> cubes.append(7 ** 3)  # y el cubo de 7
>>> cubes
[1, 8, 27, 64, 125, 216, 343]
```

Funcion len()

```
>>> letters = ['a', 'b', 'c', 'd']
>>> len(letters)
4
```



```
>>> letters = ['a', 'b', 'c', 'd', 'e', 'f', 'g']
>>> letters
['a', 'b', 'c', 'd', 'e', 'f', 'g']
>>> letters[2:5] = ['C', 'D', 'E'] # Reemplazando valores
>>> letters
['a', 'b', 'C', 'D', 'E', 'f', 'g']
>>> letters[2:5] = []# now remove them
>>> letters
['a', 'b', 'f', 'g']
>>> letters[:] = [] #limpiando lista, mandando una lista vacia
>>> letters
[]
```

Más de Listas

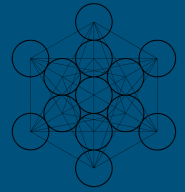
list.**append**(x)
Add an item to the end of the list. Equivalent to a[len(a):] = [x].
list.**extend**(iterable)
Extend the list by appending all the items from the iterable. Equivalent to a[len(a):] = iterable.
list.**insert**(i, x)
Insert an item at a given position. The first argument is the index of the element before which to insert, so a.insert(0, x) inserts at the front of the list, and a.insert(len(a), x) is equivalent to a.append(x).

list.**remove**(x)
Remove the first item from the list whose value is x. It is an error if there is no such item.
list.**pop**([i])
Remove the item at the given position in the list, and return it. If no index is specified, a.pop() removes and returns the last item in the list.

Más de Listas

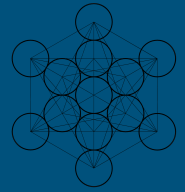
list.**clear**()
Remove all items from the list.
Equivalent to del a[:].
list.**index**(x[, start[, end]])
Return zero-based index in the list of the first item whose value is x.
Raises a ValueError if there is no such item.
list.**count**(x)
Return the number of times x appears in the list.

list.**sort**(key=None, reverse=False)
Sort the items of the list in place (the arguments can be used for sort customization, see sorted() for their explanation).
list.**reverse**()
Reverse the elements of the list in place.
list.**copy**()
Return a shallow copy of the list.
Equivalent to a[:].



Más de Listas

```
>>> fruits = ['orange', 'apple', 'pear', 'banana', 'kiwi', 'apple', 'banana']
>>> fruits.count('apple')
2
>>> fruits.count('tangerine')
0
>>> fruits.index('banana')
3
>>> fruits.index('banana', 4)  # Find next banana starting a position 4
6
```



Más de Listas

```
>>> fruits.reverse()
>>> fruits
['banana', 'apple', 'kiwi', 'banana', 'pear', 'apple', 'orange']
>>> fruits.append('grape')
>>> fruits
['banana', 'apple', 'kiwi', 'banana', 'pear', 'apple', 'orange', 'grape']
>>> fruits.sort()
>>> fruits
['apple', 'apple', 'banana', 'banana', 'grape', 'kiwi', 'orange', 'pear']
>>> fruits.pop()
'pear'
```

Tuplas

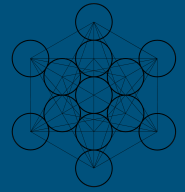
```
 t = 12345, 54321, 'hello!'
>>> t[0]
12345
>>> t
(12345, 54321, 'hello!')
>>> u = t, (1, 2, 3, 4, 5) #anidadas
>>> u
((12345, 54321, 'hello!'), (1, 2, 3, 4, 5))
```

```
>>> # Son immutables
... t[0] = 88888
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'tuple' object does not
support item assignment
>>> # pero pueden contener objetos
mutables
... v = ([1, 2, 3], [3, 2, 1])
>>> v
([1, 2, 3], [3, 2, 1])
```

Tuplas

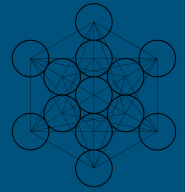
Para tuplas de 0 y 1 elemento:

```
>>> vacia= ()
>>> simple= 'hello',    # <-- note la coma
>>> len(vacia)
0
>>> len(simlpe)
1
>>> simple
('hello',)
```



Sets(conjuntos)

```
>>> conjunto = set(["un", "conjunto", "hecho", "en", "python", 3.6])
>>> conjunto
{3.6, 'un', 'hecho', 'conjunto', 'en', 'python'}

>>> conjunto = set("hola mundo!")
>>> conjunto
{' ', 'm', 'd', 'u', 'h', 'l', '!', 'o', 'n', 'a'}
>>>
```



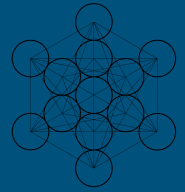
Sets(conjuntos)

```
>>> a = set('abracadabra')
>>> b = set('alacazam')
>>> a                                  # unique letters in a
{'a', 'r', 'b', 'c', 'd'}
>>> a - b                              # letters in a but not in b
{'r', 'd', 'b'}
>>> a | b                              # letters in a or b or both
{'a', 'c', 'r', 'd', 'b', 'm', 'z', 'l'}
>>> a & b                              # letters in both a and b
{'a', 'c'}
>>> a ^ b                              # letters in a or b but not both
{'r', 'd', 'b', 'm', 'z', 'l'}
```

Diccionarios

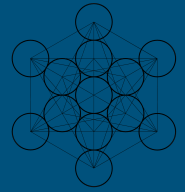
```
>>> tel = {'jack': 4098, 'sape': 4139}
>>> tel['guido'] = 4127
>>> tel
{'sape': 4139, 'guido': 4127, 'jack': 4098}
>>> tel['jack']
4098
>>> del tel['sape']
>>> tel['irv'] = 4127
>>> tel
{'guido': 4127, 'irv': 4127, 'jack': 4098}
```

```
>>> list(tel.keys())
['irv', 'guido', 'jack']
>>> sorted(tel.keys())
['guido', 'irv', 'jack']
>>> 'guido' in tel
True
>>> 'jack' not in tel
False
```



Diccionarios

```
>>> knights = {'gallahad': 'the pure', 'robin': 'the brave'}
>>> for k, v in knights.items():
...     print(k, v)
...
gallahad the pure
robin the brave
```

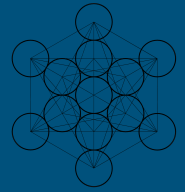


Diccionarios

```
>>> questions = ['name', 'quest', 'favorite color']
>>> answers = ['lancelot', 'the holy grail', 'blue']
>>> for q, a in zip(questions, answers):
...     print('What is your {0}?  It is {1}.'.format(q, a))
...
What is your name?  It is lancelot.
What is your quest?  It is the holy grail.
What is your favorite color?  It is blue.
```

Operadores Relacionales

Operador	Descripción	Ejemplo
==	¿Son iguales a y b?	r=5==3 #r es False
!=	¿Son distintos a y b?	r=5!=3 #r es True
<	¿Es a menor que b?	r=5<3 #r es False
>	¿Es a mayor que b?	r=5>3 3 r es True
<=	¿Es a menor o igual que b?	r=5<=5 #r es True
>=	¿Es a mayor o igual que b?	r=5>=3 #r es True



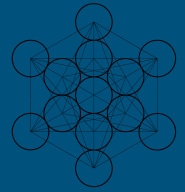
Sentencias Condicionales

Herramientas de control

```
>>> edad = int(input("¿Cuántos años tiene? "))
¿Cuántos años tiene? 17
>>> if edad < 18:
    print("Es usted menor de edad")
else:
    print("Es usted mayor de edad")

Es usted menor de edad
>>>
```

Herramientas de control

```
edad = int(input("¿Cuántos años tiene? "))
>>> if edad >= 18:
    print("Es usted mayor de edad")
elif edad < 0:
    print("No se puede tener una edad
negativa")
else:
    print("Es usted menor de edad")


Es usted mayor de edad
>>>
```

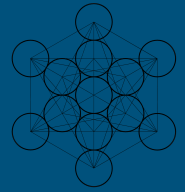


if

Herramientas de control

```
>>> words = ['cat', 'window', 'defenestrate']
>>> for w in words:
...     print(w, len(w))
...
cat 3
window 6
defenestrate 12
```

for



Herramientas de control

```
>>> for i in "AMIGO":
    print(f"Dame una {i}")

Dame una A
Dame una M
Dame una I
Dame una G
Dame una O
>>> print("¡AMIGO!")
¡AMIGO!
>>>
```

for

Herramientas de control

```
>>> for i in range(10):
    print("Hola ", end="")
```

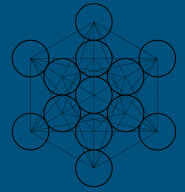
Hola Hola Hola Hola Hola Hola Hola Hola Hola Hola

for

Herramientas de control

```
>>> for i in range(5):
...     print(i)
...
0
1
2
3
4
```

range



Herramientas de control

```
>>> x = range(10)
>>> x
range(0, 10)
>>> list(x)
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> range(7)
range(0, 7)
>>> list(range(7))
[0, 1, 2, 3, 4, 5, 6]
```

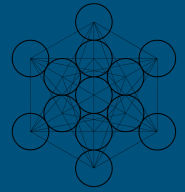
range

Herramientas de control

valores de **inicio** y **fin(range(m, n)) Si n es menor o igual que m, se crea un range vacío.**

```
>>> list(range(5, 10))
[5, 6, 7, 8, 9]
>>> list(range(-5, 1))
[-5, -4, -3, -2, -1, 0]
```

range

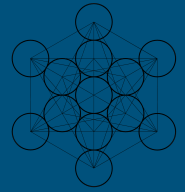


Herramientas de control

valores de **inicio, fin y salto(range(m, n, p))**

```
>>> list(range(5, 21, 3))
[5, 8, 11, 14, 17, 20]
>>> list(range(10, 0, -2))
[10, 8, 6, 4, 2]
```

range



Herramientas de control

Concatenar

```
>>> range(3) + range(5) #error
>>> list(range(3)) + list(range(5))
[0, 1, 2, 0, 1, 2, 3, 4]

>>> range(1, 3) + range(3, 5)#error
>>> list(range(1, 3)) + list(range(3, 5))
[1, 2, 3, 4]
```

range

Herramientas de control

```
i = 1
while i <= 3:
    print(i)
    i += 1
print("Programa terminado")
```

while

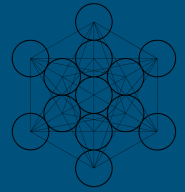
Herramientas de control

```
for letra in "Python":
    if letra == "h":
        break
    print ("Letra actual : " + letra)
```

brake

continue

pass



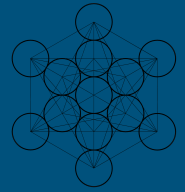
Herramientas de control

```
>>> for num in range(2, 10):
        if num % 2 == 0:
            print("es par", num)
            continue
        print("no es par", num)
es par 2
no es par 3
es par 4
no es par 5
es par 6
no es par 7
es par 8
no es par 9
```

brake

continue

pass



Herramientas de control

```
>>> for letra in "Python":
    if letra == "h":
        pass
    print ("Letra actual :" + letra)
```

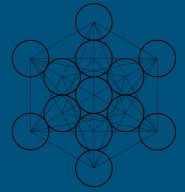
```
Letra actual :P
Letra actual :y
Letra actual :t
Letra actual :h
Letra actual :o
Letra actual :n
```

No hace nada, solo se pasa

brake

continue

pass



Funciones

```
>>> def fib(n):    # write Fibonacci series up to n
    a, b = 0, 1
    while b < n:
        print(b, end=' ')
        a, b = b, a+b
    print()

>>> fib(30)
1 1 2 3 5 8 13 21
```

Funciones

```
>>> def algo(n1=0,n2=0):
        print (n1 + n2)

>>> algo(3)
3
>>> algo()
0
>>> algo(3,4)
7
>>>
```

Clases y Objetos

```
class Humano:
    def __init__(self, edad):
        self.edad=edad #atributo edad
        # print ("Soy un nuevo objeto")

    def hablar(self,mensaje):
        print (mensaje)

pedro=Humano(45)
raul=Humano(21)
```

```
print ("Soy Pedro y tengo ", pedro.edad)
print ("Soy Raul y tengo ", raul.edad)

pedro.hablar("Saludos")
raul.hablar("mucho gusto")
```

Clases y Objetos

```
class Humano:
    def __init__(self, edad):
        self.edad=edad #atributo edad
        # print ("Soy un nuevo objeto")

    def hablar(self,mensaje):
        print (mensaje)

pedro=Humano(45)
raul=Humano(21)
```

```
print ("Soy Pedro y tengo ", pedro.edad)
print ("Soy Raul y tengo ", raul.edad)

pedro.hablar("Saludos")
raul.hablar("mucho gusto")
```

```
class Humano:
    def __init__(self, edad):
        self.edad=edad #atributo edad
    def hablar(self,mensaje):
        print (mensaje)

class
IngSistemas(Humano):#...(Humano)
    def programar(self,lenguaje):
        print ("Voy a programar en ",
lenguaje)

class LicDerecho(Humano):
    def estudiarCaso(self,de):
        print("debo estudiar el caso de ", de)
```

```
pedro=IngSistemas(45)
raul=LicDerecho(21)

print ("Soy Pedro y tengo ",
pedro.edad)
print ("Soy Raul y tengo ", raul.edad)

pedro.programar("Python")
raul.estudiarCaso("Pedro")

pedro.hablar("Saludos")
raul.hablar("mucho gusto")
```

Herencia

```
>>>

Soy Pedro y tengo  45
Soy Raul y tengo  21
Voy a programar en  Python
debo estudiar el caso de  Pedro
Saludos
mucho gusto
>>>
```

Herencia múltiple

```
.
.
class estudioso(IngSistemas,LicDerecho):
    pass

pedro=IngSistemas(45)
raul=LicDerecho(21)
pepe=estudioso(25)

pepe.hablar("Soy pepe multiple")
pepe.programar("Java")
pepe.estudiarCaso("Juan")
```

Encapsulación

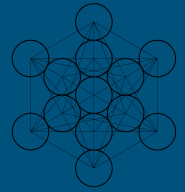
```
class Prueba(object):
    def __init__(self):
        self.__priv="Soy att privado"
        self.priv="Soy un att Publico"
    def __metPrivado(self):
        print("soy met Privado")

    def metPublico(self):
        print ("Soy Met Publico")

obj=Prueba()

#print(obj.priv)
```

```
#print(obj.priv)
#print(obj.__priv)

print(obj.metPublico())
print(obj.__metPublico())
```

Encapsulación

```
.
.
.
def getPrivado(self):
        return self.__priv

obj=Prueba()

#print(obj.priv)
print(obj.getPrivado())
```

Encapsulación



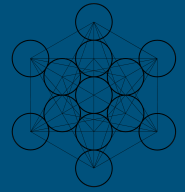
```
.
..
def getPrivado(self):
        return self.__priv
 def setPrivado(self,valor):
        self.__priv=valor

obj=Prueba()

#print(obj.priv)
obj.setPrivado("Soy att privado extranjero")
print(obj.getPrivado())
```

Cadenas y Métodos

Las cadenas son objetos

```
>>> x="Hola Mundo"
>>> len(x)
10
>>> print(x.count("o"))
2
>>> print(x.count("o",0,5))
1
>>> print(x.lower())
hola mundo
>>> print(x.upper())
HOLA MUNDO
>>>
```

```
>>> print(x.replace("o","x"))
Hxla Mundx
>>> print(x.replace("o","f",1))
Hfla Mundo
>>> print(x.split('a'))
['Hol', ' Mundo']
>>> print(x.split('o'))
['H', 'la Mund', '']
>>> print(x.split('o',1))
['H', 'la Mundo']
>>> print(x.find("a"))
3
>>> print(x.rfind("o"))
9
>>> t=("H","o","l","a")
>>> s=";"
>>> print(s.join(t))
H;o;l;a
```



Salida

```
>>> print('C:\some\name')  # \n significa nueva linea!
C:\some
ame
>>> print(r'C:\some\name')  # anotando r antes de las comillas
C:\some\name
```