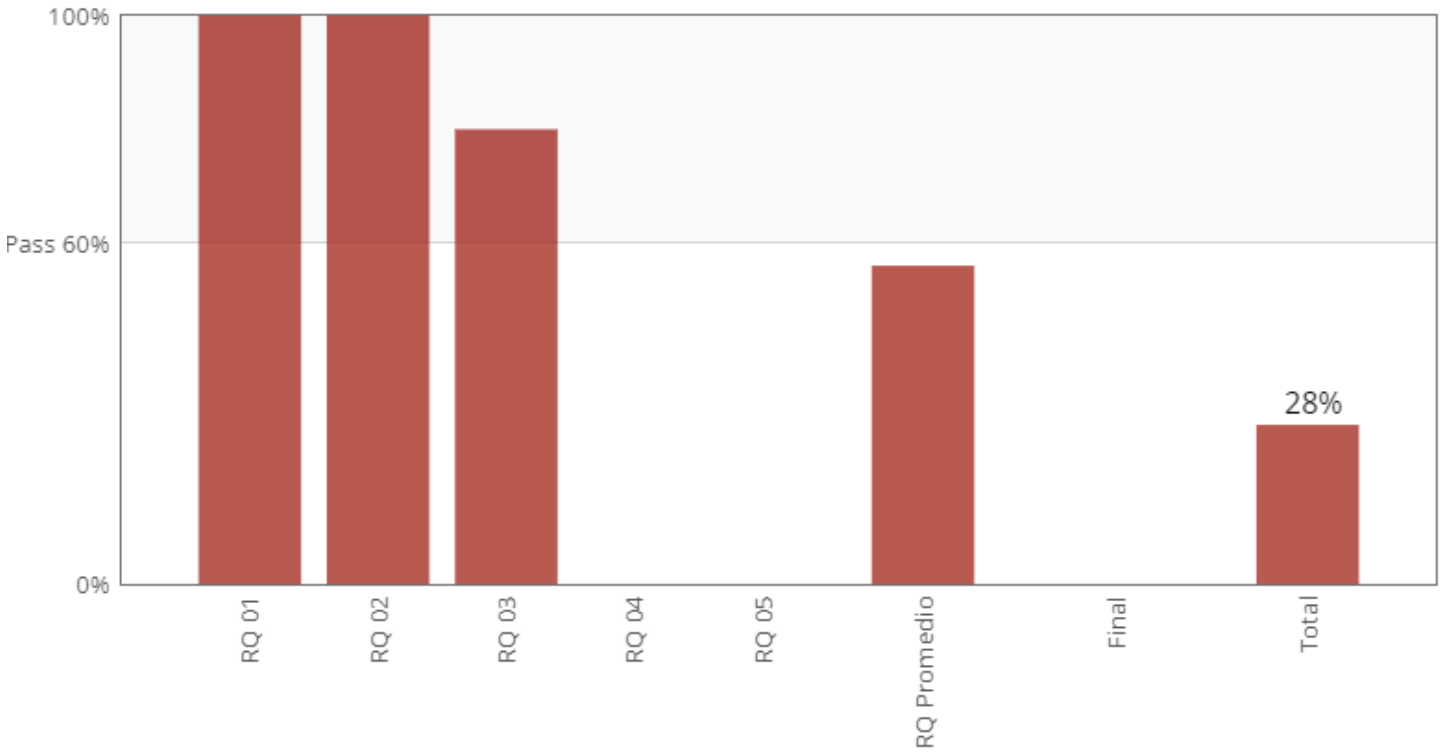


M O D U L O 1

----------**Contenido**----------

A statement or expresión is a instruction tath computer execute

Una frase o una instrucción es una instrucción que ejecuta la computadora

El statmenent o funcion Print()

Print (“Hello World”)

El valor entre los paréntesis es llamado argumento

Parta hacer comentarios ponemos # (hash simbol)

Un error de sintaxis o syntactic error es cunado pýthon no comprende el código por un error de escritura

Un error de semántica o semantic error es cuando la lógica de lo escrito esta mal

----------**Jupyter lab / Google Colabs**----------

#Imprimir Hola mundo

print(“Hello World”)

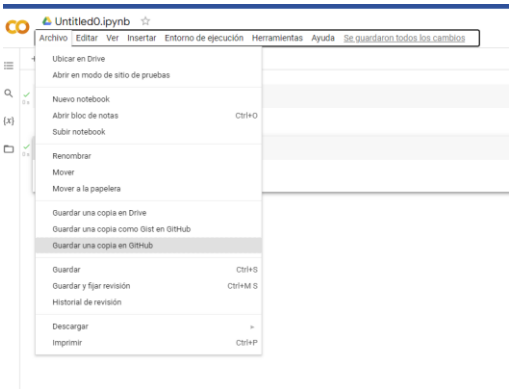
#Imprimir Hola Mundo en dos renglones diferentes

print(“Hello\nWorld”)

```
In [6]: #Imprimir Hola mundo
        print("Hello World")

        Hello World

In [5]: #Imprimir Hola Mundo en dos renglones diferentes
        print("Hello\nWorld")

        Hello
        World

In [ ]:
```


```
In [6]: #Imprimir Hola mundo
        print("Hello World")

        Hello World

In [5]: #Imprimir Hola Mundo en dos renglones diferentes
        print("Hello\nWorld")

        Hello
        World

In [ ]:
```


Nota para guardar de colab a github, tienes que seleccionar



Y después vinvular tu cuenta de colab con la de github

Si estas trabajando en jupyter, solo guarda el archivo y súbelo a tu repositorio de github

----------Quiz----------

Question 1

1/1 punto (no calificado)

Using the **Practice Lab**, find the result of executing `print("Hello\nWorld!")` statement.

- ○ Hello\nWorld!
- ○ Hello World!
- ◉ Hello
  World!

✔

Enviar Ha realizado 2 de 2 intentos

✔ Correcto (1/1 punto)

Question 2

1/1 punto (no calificado)

What is the output of executing the following statement: `# print('Hello World!')` ?

- ○ Hello World!
- ○ "Hello World!"
- ◉ There is no output as it is a comment.

✔

Guardar

Enviar Ha realizado 1 de 2 intentos

✔ Correcto (1/1 punto)

----------**Contenido**----------

Los tipos de información es la forma en que se representan diversos tipos de datos

- Enteros o Integers (35) pueden ser positivos y negativos
- Flotantes o floats (35.568) pueden ser positivos y negativos con punto decimal
- Cadenas o strings (Hola)
- Boleanos o booleans(true or false), es negativo o falso

Typecating, nos permite cambiar de un tipo a otro

Ponemos el tipo al que queremos cambiar y entre paréntesis el valor en su tipo original:

Para cambiar un entero a flotante: float(2), el resultado es 2.0

Para cambiar de floatante a entero: int(1.1), el resultado es 1, obsérvese que se pierde información

Para cambiar una cadena a entero: int (‘1’), el resultado es 1, considerar que no se puede hacer lo mismo con texto, por ejemplo, poner una letra nos devolverá un error

Para cambiar un boleano a entero: int(True), nos devolverá 1, el false nos devolverá cero, y viceversa de entero a boleano

----------Jupyter lab / Google Colabs----------

#Encontrar el tipo de la 4nformación

#”Hello World””1.1”

type (“Hello World”)

#Encontrar el tipo de la 4nformación

#1.1”

type (1.1)

#Casto o convertir el valor primero a entero int a booleano bool

#”1”

bool(int(“1”))

----------Quiz----------

Question 1

1/1 punto (no calificado)

What is the type of the following: int(1.0)?

- float
- (•) int
- bool

✔

Guardar

Enviar Ha realizado 1 de 2 intentos

✔ Correcto (1/1 punto)

Question 2

1/1 punto (no calificado)

Enter the code to convert the number 1 to a Boolean.

bool(1) ✔

Guardar

Enviar Ha realizado 1 de 2 intentos

✔ Correcto (1/1 punto)

----------**Contenido**----------

Las expresiones describen un tipo de operaciones que las computadoras ejecutan

Expresiones de operaciones matemáticas

+ suma

- Substraction

* Multiplication

/ division

// división entera, con resultado redondeado

Los números son llamados operands u operadons, y los símbolos son llamados operadores u operators

La realizaci[on sigue las reglas de las convenciones matemáticas, es decir priorizando multiplicacion y division y después suma y resta. Pero primero se le da priorida a las operaciones entre parentesis

Las variables guardan valores, les ponemos el nombre deseado y asignamos valor usando el operador de asignacion el símbolo =

En las variables no se puede poner espacio, usamos guion bajo, por ejemplo: Mi_variable=10

Las variables pueden ir cambiando y contenerse a sí mismas, por ejemplo:

asignamos un valor inicial de x=10

después lo reemplazamos x=x+15, ahora x vale 25, y así sucesivamente

----------Jupyter lab / Google Colabs----------

```
#Imprimir una salida

print('Hello, Python!')

#Revisar nuestra version de python

import sys

print(sys.version)

#Escribir comentarios

print('Hello, Python!') #Esta linea imprime una cadena

#Escribir un codigo erroneo

print("Hello, Python!)

#Escribir tres cadenas, una con error, para ver el orden

print("This will be printed")

frint("This will cause an error")

print("This will NOT be printed")

#Encontrar los tipos de informaci[on

type(12)

type(2.14)

type("Hello, Python 101!")

type(-1)

type(4)

type(0)

type(1.0)

type(0.5)

type(0.56)

type(False)
```

```
type(True)
type (6/2)
type(6//2)
#Conocer la informacion del sistema sobre el tipo flotante
sys.float_info
#Convertir los valores y ver a que tipo cambiaron
float(2)
type(float(2))
int(1.1)
type(int(1.1))
int('1')
float('1.2')
str(1)
str(1.2)
int(True)
bool(1)
bool(0)
float(True)
#Convertir un valor con error
int('1 or 2 people')
#Calculos
#Cuantas horas tiene un minuto
160/60
#Guardar valor en variables e imprimirlos
x = 43 + 60 + 16 + 41
x
y = x / 60
y
total_min = 43 + 42 + 57
total_min
total_hours = total_min / 60
total_hours
```

```
x = 3 + 2 * 2

y = (3 + 2) * 2

z = x + y

z
```

```
In [2]: #Imprimir una salida
        print('Hello, Python!')

Hello, Python!
```

```
In [3]: #Revisar nuestra version de python
        import sys
        print(sys.version)

3.10.7 (tags/v3.10.7:6cc6b13, Sep  5 2022, 14:08:36) [MSC v.1933 64 bit (AMD6
4)]
```

```
In [4]: #Escribir comentarios
        print('Hello, Python!') #Esta linea imprime una cadena

Hello, Python!
```

```
In [5]: #Escribir un codigo erroneo
        print("Hello, Python!)

  Cell In [5], line 2
    print("Hello, Python!)
          ^
SyntaxError: unterminated string literal (detected at line 2)
```

In [6]:
```
#Escribir tres cadenas, una con error, para ver el orden
print("This will be printed")
frint("This will cause an error")
print("This will NOT be printed")
```

```
This will be printed

---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In [6], line 3
      1 #Escribir tres cadenas, una con error, para ver el orden
      2 print("This will be printed")
----> 3 frint("This will cause an error")
      4 print("This will NOT be printed")

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

In [12]:
```
#Encontrar los tipos de informaci[on
type(12)
type(2.14)
type("Hello, Python 101!")
type(-1)
type(4)
type(0)
type(1.0)
type(0.5)
type(0.56)
type(False)
type(True)
type (6/2)
type(6//2)
```

Out[12]:
```
int
```

In [13]:
```
#Conocer la informacion del sistema sobre el tipo flotante
sys.float_info
```

Out[13]:
```
sys.float_info(max=1.7976931348623157e+308, max_exp=1024, max_10_exp=308, min=
2.2250738585072014e-308, min_exp=-1021, min_10_exp=-307, dig=15, mant_dig=53, e
psilon=2.220446049250313e-16, radix=2, rounds=1)
```

In [14]:
```
#Convertir los valores y ver a que tipo cambiaron
float(2)
type(float(2))
int(1.1)
type(int(1.1))
int('1')
float('1.2')
str(1)
str(1.2)
int(True)
bool(1)
bool(0)
float(True)
```

Out[14]:
```
1.0
```

In [15]:
```
#Convertir un valor con error
int('1 or 2 people')
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In [15], line 2
      1 #Convertir un valor con error
----> 2 int('1 or 2 people')

ValueError: invalid literal for int() with base 10: '1 or 2 people'
```

In [16]:
```
#Calculos
#Cuantas horas tiene un minuto
160/60
```

Out[16]:
```
2.6666666666666665
```

In [17]:
```
#Guardar valor en variables e imprimirlos
x = 43 + 60 + 16 + 41
x
y = x / 60
y
total_min = 43 + 42 + 57
total_min
total_hours = total_min / 60
total_hours
x = 3 + 2 * 2
y = (3 + 2) * 2
z = x + y
z
```

Out[17]:
```
17
```

----------Quiz----------

No hay quiz

----------Contenido----------

Sring Mehods y String Operators

Una cadena o string, es una secuencia de caracteres, y se escriben entre comillas dobles o simples. Pueden ser letas o caracteres.

Cada letra tiene una posici[on

Si a la variable Name le damos el valor “Michael Jackson”, para obtener la letra m tenemos que acceder así>

Name[0] y esto nos devolverá la letra M, ya que es la que está en la posicion 0

También se puede hacer la indezacion inversa, es decir accediendo pero para atrás

Para acceder a varos valores,usamos slicing, o ponemos un rango desde que posición hasta que posición

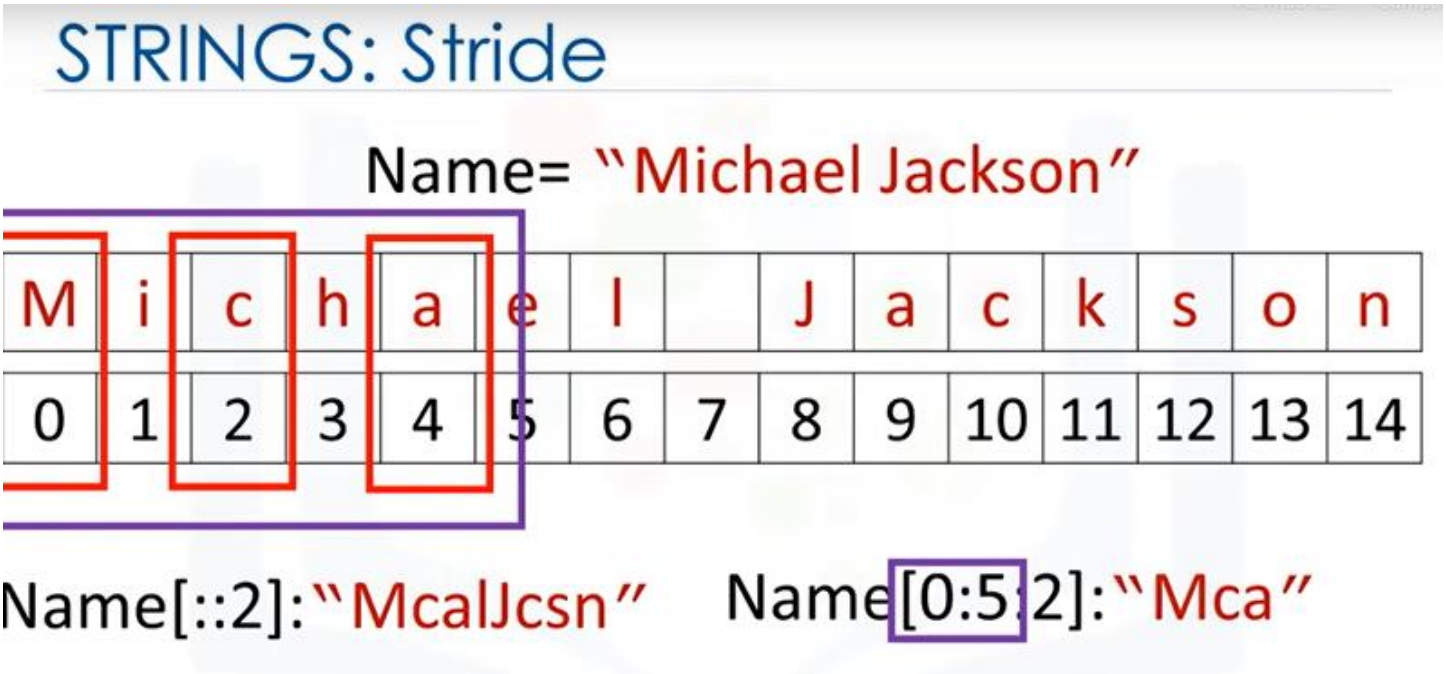
Name[0:4]

Para dar saltos, usamos el stride que indica que se darán saltos cada determinado número>

Name [::2], esto devolverá todas los valores en cada dos posiciones

Podemos hacer stride, pero cortado, poniendo el valor inicial, el valor final, y los saltos

Name[0:5:2]



Para obtener la longitud usamos el comando len, ejemplo len("Hola") dará como resultado 4

Para combinar o concatenar cadenas usamos el símbolo +, ejemplo>

Nombre =Carlos, Saludo= Nombre + "Es el mas mejor", Saludo será igual a Carlos Es el mas mejor

Para replicar cadenas, usamos el símbolo de multiplicaci[on *

"Carlos"*3 , dará como resultado CarlosCarlosCarlos

Las cadenas son inmutables, ósea que no se pueden cambiar de valor ósea no se puede asignar así Nombre[0]="F", esto no dará como resultado Farlos, sino que dará un error

\, diagonal invertida representan secuencias de escape

\n representa enter o pasara la siguiente renglón

\t, es un espacio de tabulador

\\, Si necesitamos escribir solo una diagonal invertida, debemos de poner 2, y solo aparece una, o ponemos la letra r antes de la primera comilla doble de la cadena.

Los métodos

El método mayúscula o uppper(), cambia todo a mayúscula, ejemplo>

A="Hola", B=A.upper(), El valor de B ahora es "HOLA"

El método reemplazar o replace(, rremplaza una cadena por otra

Al usar el método, ponemos primero la palabra a reemplazarse y después la palabra que la sustituirá, ejemplo>

A='Carlos Sanchez', B=A.replace('Carlos', 'Juan'), el resultado será B igual 'Juan Sanchez'

El método encontrar, devuelve la posicion donde inicia un texto buscado también llamada substring, porque está dentro de una cadena, ejemplo

Si tenemos la variable Nombre="Raul Lopez", usamos el método Nombre.find('Lopez'), el resultado será el numero 6, porque lopez inicia en la posicion 5 de la cadena "Raul Lopez"

----------**Jupyter lab / Google Colabs**----------

```
#Crear diferentes strings de diversas formas

'Michael Jackson'

"Michael Jackson"

'@#2_#]&*^%$'

"1 2 3 4 5"

#Imprimir cadenas

print('Hola Mundo')

name = "Michael Jackson"

name

#Imprimir elementos con indices en diferentes posiciones

name="Rafael Oliveiros"

print(name[0])

print(name[6])

print(name[8])

#Imprimir elementos en posiciones invertidas

name="Rafael Oliveiros"

print(name[-2])

print(name[-9])

print(name[-1])

#Onbtener la longitud de una cadena

len("Michael Jackson")

#Obtener elementos d euna cadena en diferentes posiciones, slice

name='Pedrito Sola'

name[0:4]

name[3:8]

#Obtener elementos de una cadena dando determinados saltos

name="Thor Odinson"

name[::2]

#Obtener elementos de una cadena dando determinados saltos, pero dentro de un rango seleccionado

name='Septiembre patrio'

name[0:5:2]
```

```
#Concatenar dos o mas cadenas e imprimirlas
nombre='Carlos'
frase=nombre + 'es el mejors'
frase
#multiplicar una cadena
5 * 'frase repetida '
#Usar 3 secuencias de escape
print(" Michael Jackson \n is the best" )
print(" Michael Jackson \t is the best" )
print(" Michael Jackson \\ is the best" )
print(r" Michael Jackson \ is the best" )
#USar operadores de cadenas
a = "Hola mundo"
print("en minuscula es:", a)
b = a.upper()
print("en mayuscula es", b)
g='carlos'
g.upper()

a = "OScar is the best"
b = a.replace('OScar', 'Carlos')
b
g.replace('ar','al')

name = "Michael Jackson"
name.find('el')
g.find('al')
#Ver que ocurre cuando no encuentra
name.find('Jasdfasdasdf')
#imrprimir diagonal invertida
print("\\\\\\")
print(r"\ ")
```

In [1]:
```
#Crear diferentes strings de diversas formas
'Michael Jackson'
"Michael Jackson"
'@#2_#]&*^%$'
"1 2 3 4 5"
```

Out[1]: '1 2 3 4 5'

In [2]:
```
#Imprimir cadenas
print('Hola Mundo')
name = "Michael Jackson"
name
```

```
Hola Mundo
```

Out[2]: 'Michael Jackson'

In [3]:
```
#Imprimir elementos con indices en diferentes posiciones
name="Rafael Oliveiros"
print(name[0])
print(name[6])
print(name[8])
```

```
R

l
```

In [4]:
```
#Imprimir elementos en posiciones invertidas
name="Rafael Oliveiros"
print(name[-2])
print(name[-9])
print(name[-1])
```

```
o
O
s
```

In [5]:
```
#Onbtener la longitud de una cadena
len("Michael Jackson")
```

Out[5]: 15

In [6]:
```
#Obtener elementos d euna cadena en diferentes posiciones, slice
name='Pedrito Sola'
name[0:4]
name[3:8]
```

Out[6]: 'rito '

In [7]:
```
#Obtener elementos de una cadena dando determinados saltos
name="Thor Odinson"
name[::2]
```

Out[7]: 'To dno'

In [8]:
```
#Obtener elementos de una cadena dando determinados saltos, pero dentro de un ra
name='Septiembre patrio'
name[0:5:2]
```

Out[8]: 'Spi'

In [9]:
```
#Concatenar dos o mas cadenas e imprimirlas
nombre='Carlos'
frase=nombre + 'es el mejors'
frase
```

Out[9]: 'Carloses el mejors'

In [10]:
```
#multiplicar una cadena
5 * 'frase repetida '
```

Out[10]: 'frase repetida frase repetida frase repetida frase repetida frase repetida '

In [11]:
```
#Usar 3 secuencias de escape
print(" Michael Jackson \n is the best" )
print(" Michael Jackson \t is the best" )
print(" Michael Jackson \\ is the best" )
print(r" Michael Jackson \ is the best" )
```

```
 Michael Jackson
 is the best
 Michael Jackson         is the best
 Michael Jackson \ is the best
 Michael Jackson \ is the best
```

In [12]:
```
#USar operadores de cadenas
a = "Hola mundo"
print("en minuscula es:", a)
b = a.upper()
print("en mayuscula es", b)
g='carlos'
g.upper()

a = "OScar is the best"
b = a.replace('OScar', 'Carlos')
b
g.replace('ar','al')

name = "Michael Jackson"
name.find('el')
g.find('al')
```

```
en minuscula es: Hola mundo
en mayuscula es HOLA MUNDO
```

Out[12]:
```
-1
```

In [13]:
```
#Ver que ocurre cuando no encuentra
name.find('Jasdfasdasdf')
```

Out[13]:
```
-1
```

In [14]:
```
#imrprimir diagonal invertida
print("\\\\\\")
print(r"\ ")
```

```
\\\
\
```

----------Quiz----------

QUESTION 1

1/1 punto (no calificado)

Consider the following string:

```
Numbers = "0123456"
```

How would you obtain the even elements?

- ○ Numbers[::3]
- ◉ Numbers[::2]
- ○ Numbers[::6]

✔

Guardar

Enviar Ha realizado 1 de 2 intentos

✔ Correcto (1/1 punto)

QUESTION 2

1/1 punto (no calificado)

What is the result of the following line of code:

```
"0123456".find('1')
```

- ○ 0
- ◉ 1
- ○ '1'
- ○ True

✔

Guardar

Enviar Ha realizado 1 de 2 intentos

✔ Correcto (1/1 punto)

`3 + 2 * 2`

- ○ `10`
- ◉ `7`
- ○ `9`
- ○ `12`

✔

Enviar Ha realizado 2 de 2 intentos

Review Question 2

1/1 punto (calificado)

In Python, if you executed `name = 'Lizz'`, what would be the output of `print(name[0:2])`?

- ○ `Lizz`
- ○ `L`
- ○ `Liz`
- ◉ `Li`

✔

Enviar Ha realizado 2 de 2 intentos

✔ Correcto (1/1 punto)

Review Question 3

1/1 punto (calificado)

In Python, if you executed `var = '01234567'`, what would be the result of `print(var[::2])`?

- (•) `0246`
- ( ) `1357`
- ( ) `1234567`
- ( ) `8903`

✔

Enviar Ha realizado 2 de 2 intentos

✔ Correcto (1/1 punto)

Review Question 4

1/1 punto (calificado)

In Python, what is the result of the following operation `'1'+'2'`?

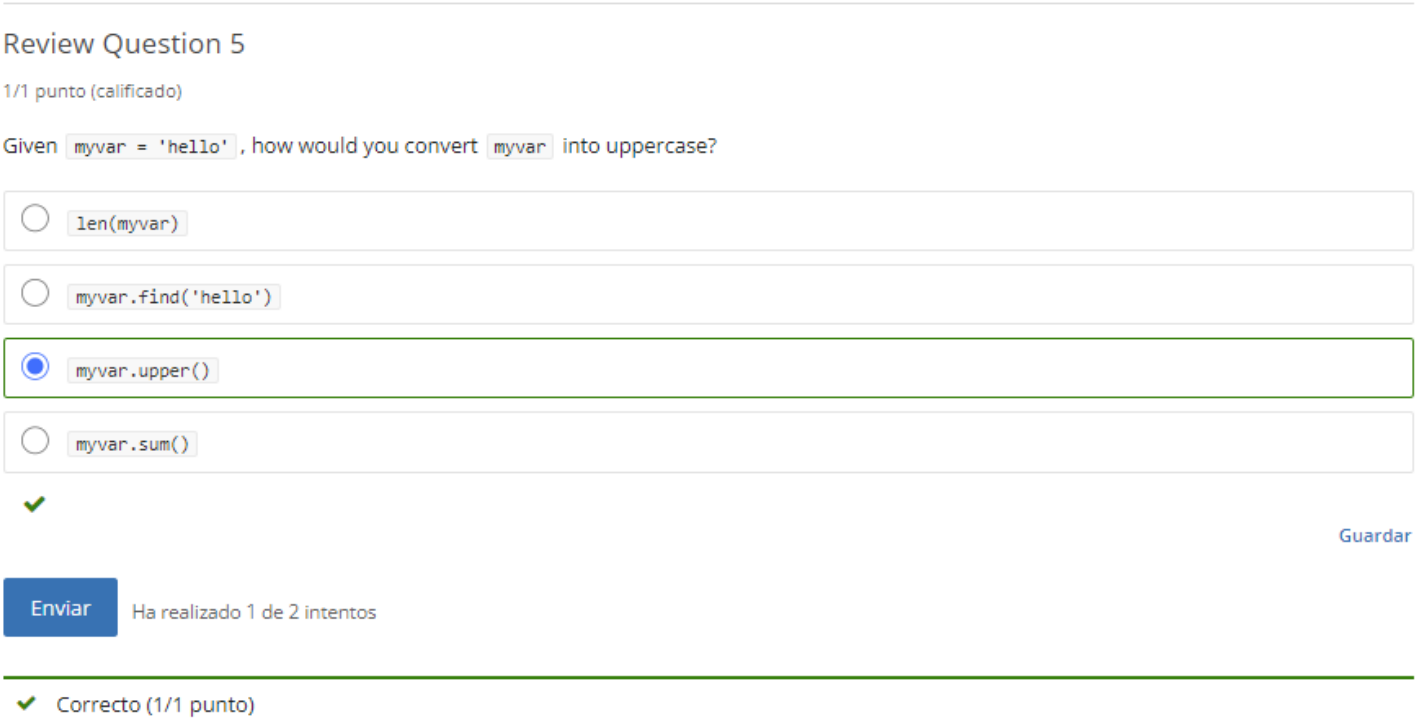
- ( ) `'2'`
- ( ) `'3'`
- (•) `'12'`
- ( ) `3`

✔

Guardar

Enviar Ha realizado 1 de 2 intentos

✔ Correcto (1/1 punto)



M O D U L O 2

----------Contenido----------

List and tuples

Los tuples son una secuencia ordenada de datos separados por comas, se escriben siempre entre paréntesis

Los tuples pueden tener diferentes tipos de de datos, desde cadenas, hasta enteros y flotantes, ejemplo:

Tuple_1=(“Carlos”, 10, 1.5), de esta variable el tipo se tuple

Los tuples tienen índices que permiten acceder a los datos que tienen almacenados. Para el ejemplo anterior, “Carlos” está en el índice 0, 10 está en el índice 1 y así sucesivamente. Para acceder se hace esto

Tuple_1[0], nos dará como respuesta “Carlos” , también se puede acceder de forma inversa, en este caso el ultimo valor esta en la posición -1, por lo tanto para acceder a “Carlos”, escribimos Tuple_1[-3]

Los tuples también se pueden combinar o concatenar con el signo +, como si fueran strings

Los tuples también se pueden rebanar o aplicárseles slicing. Por ejemplo, para obtener los primeros tres valores usamos

Tuple_1[0:3], esto dará los tres primeros valores, es decir, el índice 0,1 y 2.

También podemos usar el comando len, para saber la longitud y la cantidad de elementos

Los tuples son inmutables, no los podemos modificar

Podemos usar la función ordenar o sort, para organizar un tuple, pero este tiene que almacenarse en una nueva variable, por ejemplo:

Calificaciones=(10,9,8,5,7,9,10), y calificaciones_ordenadas=sorted(Calificaciones)

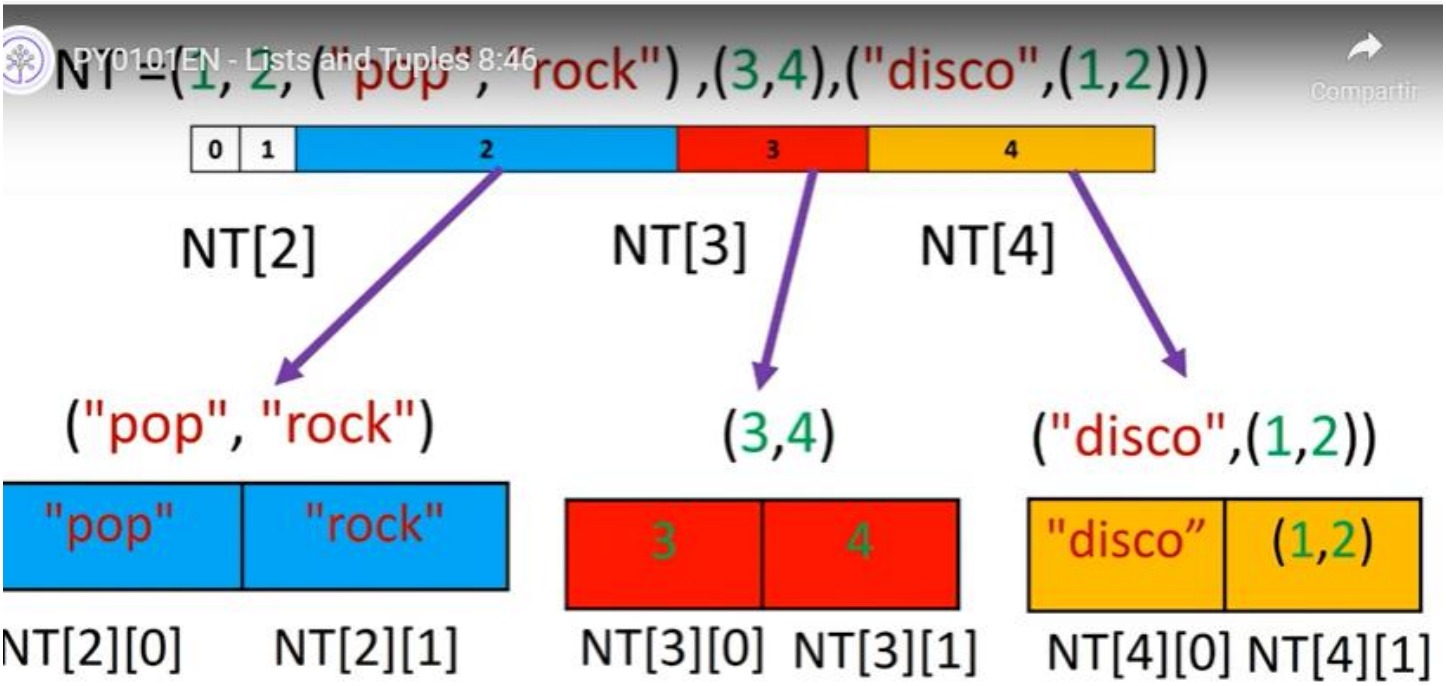
Los tuples pueden ser anidados, ejemplo:

Nombres=(“juan”,”oscar”,”pedro”), calificaciones=(10,9,8), anidado=(“ramon”, (5,15,20), 25,(“juan”,2),”oscar”)

Si accedemos al índice anidado[1], el resultado será 5,15 y 20, porque cuenta como 1 solo

Podemos usar dos valores como si se deseara acceder a una matriz, por ejemplo

Para acceder a 15 en nuestro ejemplo, tenemos que entrara al segundo valor del segundo valor, por lo que seria anidado[1][1], esto daría como resultado 15.



Se puede acceder hasta diferentes niveles, dependiendo el nivel de anidamiento que se tenga

Las listas son otro tipo de estructuras de datos, que se ordenan en secuencias, pero a diferencia de los tuples, las listas van entre corchetes [] y no entre paréntesis ()

Las listas si son mutables, ósea que se pueden reemplazar los valores que contienen, también pueden contener cadena, enteros y boleando, y también pueden contener otras listas y también otros tuples, respetando las reglas de parentesios y corcheas según corresponda

Las litas tienen las mismas reglas de acceso o index rules que los tuples, también se les puede aplicar slicing y también se concatenan con el signo +, pero este ultimo concatenamiento en una nueva variable

Otra forma de concatenar listas o aumentaralas, pero en la misma variable es por el método extend

Lista=[1,2,3,4,5] , y Lista.extend([6,7,8]) , dando como resultado Lista=[1,2,3,4,5,6,7,8]

El método append, también muta la lista, pero en vez de añadirlos por separaciones por coma, añade los valores anidándolos en un solo espacio dentro de la lista

Lista=[1,2,3,4,5] , y Lista.append([6,7,8]) , dando como resultado Lista=[1,2,3,4,5,[6,7,8]]

Si el append es de solo un valor, este valor se añade normal, separado por coma

A diferencia de los tuples, podemos reemplazar los valores de las listas, por ejemplo

A=[0,1,2,3], A[2]=”Carlos”, el resultado es A=[0,1,”Carlos”,3]

Parar remover elemento de la lista usando el comando del, para esto usamos la posición de la lista que deseamos eliminar como un arguemtno, por ejemplo:

A=[0,1,2,3], del(A[0]), el resultado es A=[1,2,3]

Podemos convertir una cadena en una lista, con el método Split, el cual hace una lista con todos los elementos del texto separados por un espacio, por ejemplo “Carlos es el mejor”.splis(), da como resultado la lista [Carlos,es,el,mejor]

En caso de que no haya espacios podemos seleccionar el valor que al detectarse separe una string, por ejemplo, que se separe cada que haya una coma, seria “pedro,omar,c,d”.split(“,”). Esto dará como resultado [“pedro”,”omar”,”c”,”d”]

El aliasing ocurre cuando una lista hace referencia a los valores de otra lista, y cuando los valores de la lista de origen cambian, también cambian los valores de las listas que la refieren

La clonación o clone, sirve para copiar los valores de una lista, sin estar sujetos al aliasing, ósea será independiente.

El comando help, nos ayuda obtener ayuda solo tenemos que escribir help(poner aquí el objeto, lista o tuple), ejemplo Help(A)

Aclaración del slice

Variable[punto de inicio : punto final contado desde el inicio en 1, no desde el punto de inicio]

Index permite conocer la ubicación de un elemento dentro de una cadena

Ojo, index, no entra en los anidados de los tuplets, agarra los anidados como 1 solo elemento

----------Jupyter lab / Google Colabs----------

#Crear un tuple e imprimirlo

Tuple_ejemplo=("cesar",1.5,5,46,98,"ramon")

Tuple_ejemplo

#Imprimir el tipo de tuple

type(Tuple_ejemplo)

#Imprimir cada valor del tuple

Tuple_ejemplo[0]

Tuple_ejemplo[1]

Tuple_ejemplo[2]

Tuple_ejemplo[3]

Tuple_ejemplo[4]

Tuple_ejemplo[5]

#imptimr el tipo de cada valor del tuple

type(Tuple_ejemplo[0])

type(Tuple_ejemplo[1])

type(Tuple_ejemplo[2])

type(Tuple_ejemplo[3])

```
type(Tuple_ejemplo[4])

type(Tuple_ejemplo[5])

#Usar la indezacion negativa

Tuple_ejemplo[-1]

#Concatenar dos tuples

Tuple1 = (1,2,3,4,5,6)

Tuple2 = ("Victoria","Francisca","Namora")

Tuple3 = Tuple1 + Tuple2 + ("Etcetera",1)

Tuple3

#Utilizar slice y medir la longitud del tuple

Tuple3[2:4]

len(Tuple3)

#Ordenar un tuple

Tuple4=(65,54,87,4,5,1,2,3,654,98)

Tuple5=sorted(Tuple4)

Tuple5

#Hacer tuple anidada e imprimir elementos

Tuple6=(1,2,3,("Cuatro","Cinco"),6,7,("Ocho","Nueve",(10,11)))

print("El primer elemento del tuple es:", Tuple6[0])

print("El primer elemento del tuple anidado es:", Tuple6[3][1])

print("El primer elemento del tuple de segundo anidado es:", Tuple6[6][2][0])

#Encontrar la posicion o el indice de un elemento

Tuple6.index(7)
```

In [2]:
```
#Crear un tuple e imprimirlo
Tuple_ejemplo=("cesar",1.5,5,46,98,"ramon")
Tuple_ejemplo
```

Out[2]:
```
('cesar', 1.5, 5, 46, 98, 'ramon')
```

In [3]:
```
#Imprimir el tipo de tuple
type(Tuple_ejemplo)
```

Out[3]:
```
tuple
```

In [4]:
```
#Imprimir cada valor del tuple
Tuple_ejemplo[0]
Tuple_ejemplo[1]
Tuple_ejemplo[2]
Tuple_ejemplo[3]
Tuple_ejemplo[4]
Tuple_ejemplo[5]
```

Out[4]:
```
'ramon'
```

In [5]:
```
#imptimr el tipo de cada valor del tuple
type(Tuple_ejemplo[0])
type(Tuple_ejemplo[1])
type(Tuple_ejemplo[2])
type(Tuple_ejemplo[3])
type(Tuple_ejemplo[4])
type(Tuple_ejemplo[5])
```

Out[5]:
```
str
```

In [6]:
```
#Usar la indezacion negativa
Tuple_ejemplo[-1]
```

Out[6]:
```
'ramon'
```

In [12]:
```
#Concatenar dos tuples
Tuple1 = (1,2,3,4,5,6)
Tuple2 = ("Victoria","Francisca","Namora")
Tuple3 = Tuple1 + Tuple2 + ("Etcetera",1)
Tuple3
```

Out[12]:
```
(1, 2, 3, 4, 5, 6, 'Victoria', 'Francisca', 'Namora', 'Etcetera', 1)
```

In [19]:
```
#Utilizar slice y medir la longitud del tuple
Tuple3[2:4]
len(Tuple3)
```

Out[19]:
```
11
```

In [22]:
```
#Ordenar un tuple
Tuple4=(65,54,87,4,5,1,2,3,654,98)
Tuple5=sorted(Tuple4)
Tuple5
```

Out[22]:
```
[1, 2, 3, 4, 5, 54, 65, 87, 98, 654]
```

In [25]:
```
#Hacer tuple anidada e imprimir elementos
Tuple6=(1,2,3,("Cuatro","Cinco"),6,7,("Ocho","Nueve",(10,11)))
print("El primer elemento del tuple es:", Tuple6[0])
print("El primer elemento del tuple anidado es:", Tuple6[3][1])
print("El primer elemento del tuple de segundo anidado es:", Tuple6[6][2][0])
```

```
El primer elemento del tuple es: 1
El primer elemento del tuple anidado es: Cinco
El primer elemento del tuple de segundo anidado es: 10
```

In [30]:
```
#Encontrar la posicion o el indice de un elemento
Tuple6.index(7)
```

Out[30]:
```
5
```

In [ ]:
```

```

```
#Crear una lista e imprimirla
Lista=["Camion",1,2,3,"Vaca"]
Lista
#Imprimir con idnices positivos y negativos
print("EL primer elemento es",Lista[0])
print("Primer elemento con indice inverso es",Lista[-5])
#Rebanar una lista
Lista2=["Perro",23,56,("Casa","Oscar"),89,91,["zapatos",3]]
Lista2[1:4]
#Extender una lista de dos formas diferntes
Lista2.extend([4,29])
Lista2
Lista2.append([35,295])
Lista2
```

```
#Modificar y eliminar elementos de la lista
Lista2[0]='Gato'
print ("La lista vieja",Lista2)
del(Lista2[1])
print("La lista nueva",Lista2)
#Dividir cadena de dos formas diferentes
"Hola buenos dias".split()
"Hola buenos, dias, que tal".split(",")
#Copiar una lista con aliasing
Lista3 = Lista2
Lista3
#Clonar una lista, es decir sin aliasing
Lista4=Lista3[:]
Lista4
#Concatenar listas
Lista5=Lista3+Lista2
Lista5
```

In [1]:
```
#Crear una lista e imprimirla
Lista=["Camion",1,2,3,"Vaca"]
Lista
```

Out[1]:
```
['Camion', 1, 2, 3, 'Vaca']
```

In [2]:
```
#Imprimir con idnices positivos y negativos
print("EL primer elemento es",Lista[0])
print("Primer elemento con indice inverso es",Lista[-5])
```

```
EL primer elemento es Camion
Primer elemento con indice inverso es Camion
```

In [5]:
```
#Rebanar una lista
Lista2=["Perro",23,56,("Casa","Oscar"),89,91,["zapatos",3]]
Lista2[1:4]
```

Out[5]:
```
[23, 56, ('Casa', 'Oscar')]
```

In [7]:
```
#Extender una lista de dos formas diferntes
Lista2.extend([4,29])
Lista2
Lista2.append([35,295])
Lista2
```

Out[7]:
```
['Perro',
 23,
 56,
 ('Casa', 'Oscar'),
 89,
 91,
 ['zapatos', 3],
 4,
 29,
 4,
 29,
 [35, 295]]
```

In [11]:
```
#Modificar y eliminar elementos de la lista
Lista2[0]='Gato'
print ("La lista vieja",Lista2)
del(Lista2[1])
print("La lista nueva",Lista2)
```

```
La lista vieja ['Gato', ('Casa', 'Oscar'), 89, 91, ['zapatos', 3], 4, 29, 4, 2
9, [35, 295]]
La lista nueva ['Gato', 89, 91, ['zapatos', 3], 4, 29, 4, 29, [35, 295]]
```

In [12]:
```
#Dividir cadena de dos formas diferentes
"Hola buenos dias".split()
```

Out[12]:
```
['Hola', 'buenos', 'dias']
```

In [15]:
```
"Hola buenos, dias, que tal".split(",")
```

Out[15]:
```
['Hola buenos', ' dias', ' que tal']
```

In [16]:
```
#Copiar una lista con aliasing
Lista3 = Lista2
Lista3
```

Out[16]:
```
['Gato', 89, 91, ['zapatos', 3], 4, 29, 4, 29, [35, 295]]
```

In [17]:
```
#Clonar una lista, es decir sin aliasing
Lista4=Lista3[:]
Lista4
```

Out[17]:
```
['Gato', 89, 91, ['zapatos', 3], 4, 29, 4, 29, [35, 295]]
```

In [18]:
```
#Concatenar listas
Lista5=Lista3+Lista2
Lista5
```

Out[18]:
```
['Gato',
 89,
 91,
 ['zapatos', 3],
 4,
 29,
 4,
 29,
 [35, 295],
 'Gato',
 89,
 91,
 ['zapatos', 3],
 4,
 29,
 4,
 29,
 [35, 295]]
```

----------Quiz----------

QUESTION 1

1/1 punto (no calificado)

How do you access the last element of the following tuple: `A=(0,1,2,3)` ? Select all possible correct answers.

- [x] A[3]
- [ ] A[0]
- [x] A[-1]
- [ ] A[end]

✔

Note: Make sure you select all of the correct options—there may be more than one!

Enviar Ha realizado 2 de 2 intentos

✔ Correcto (1/1 punto)

QUESTION 2

1/1 punto (no calificado)

Consider the list: `B=["a","b","c"]`.

What is the result of the following `B[1:]`?

- ( ) []
- ( ) 'b'
- (•) ['b', 'c']
- ( ) ['a','b', 'c']

✔

Guardar

Enviar Ha realizado 1 de 2 intentos

✔ Correcto (1/1 punto)

----------Contenido----------

Los sets, son un tipo de colección en pyton, son similares a las listas y tuples, pues les puedes meter varios tipos de datos, la diferencia es que los sets, no están ordenados, no almacenan la posición de sus elementos.

Para definir un set se usan los corchetes curvos

Cuando se crea un set, de una lista de elementos repetivos, estos se hacen uno solo.

Para convertir una lista en un set, se usa el typecasting, usando la función set(). Ejemplo Mi_set=set(Mi_lista)

Se debe considerar que los sets, unifican los elementos repetidos o duplicados de una lista

Las operaciones en los sets

Para añadir un elemento usamos add, ejemplo: Set1.add(“Carlos”)

Considerar que si añadimos un elemento que ya está, este en realidad no se añadirá, porque no puede haber elementos repetidos

Para remover, es lo mismo, pero con el método remove

Usando el comando in, podemos verificar si un set contiene un elemento “Carlos” in Mi_Set, nos dira si dentro de mi set esta la cadena Carlos, con un True or false

Los sets también aceptan operadores matemáticos

Para unir dos sets usamos el ampersand &, lo que nos devuelve únicamente los valores comunes dentro de ambos sets, y no considera aquellos que están únicamente dentro de un set u otro

Set3= Set1 & Set2

Para unir dos sets, considerando todos los elementos, no únicamente los comunes usamos el comando unión

Set1.union(Set2)

Si tenemos un set con varios elementos y queremos revisar si estos elementos están dentro de otro set, usamos es comando issubset, que es como un in, pero para varios elementos. Ejemplo

Set3.issubset(album1), esto vertificaria si los elementos del set 3, están contenidos en el set 1, nos devuelve true or false

También hay issuperset, pero no esta explicado

Difference, encontrar los elementos que sean únicos en un set

Set_unico.difference(Set_a_comparar)

----------Jupyter lab / Google Colabs----------

#Convertir una lista en un set

Lista=['a','b','c']

Mi_set=set(Lista)

Mi_set

#Anadir un elemento a un set

Set1={'A','B','C'}

Set1.add('Casa')

Set1

#Hacer una interseccion entre dos sets

Set2={'A','B','C','z','g','f'}

Set1&Set2

Set1.intersection(Set2)

In [1]:
```
#Convertir una lista en un set
Lista=['a','b','c']
Mi_set=set(Lista)
Mi_set
```

Out[1]: {'a', 'b', 'c'}

In [2]:
```
#Anadir un elemento a un set
Set1={'A','B','C'}
Set1.add('Casa')
Set1
```

Out[2]: {'A', 'B', 'C', 'Casa'}

In [3]:
```
#Hacer una interseccion entre dos sets
Set2={'A','B','C','z','g','f'}
Set1&Set2
Set1.intersection(Set2)
```

Out[3]: {'A', 'B', 'C'}

#Crear un set e imprimirlo

set1 = {"pop", "rock", "soul", "hard rock", "rock", "R&B", "rock", "disco"}

set1

#Convertir una lista en un set y anadir elementos

Set_de_musica=set(['pop','cumbia','salsa'])

Set_de_musica.add('nerengue')

Set_de_musica

#Buscar un elemento y eliminarlos

'nerengue' in Set_de_musica

Set_de_musica.remove('nerengue')

Set_de_musica

#unir dos sets y buscar la interseccion

Set2={'pop','bumbia','regue','ska'}

Set_de_musica,set2

interseccion=Set_de_musica&set2

intersección

#Buscar los elementos que estan solo en un set usando difference

Set_de_musica.difference(Set2)

#Unir dos sets

```
Set_de_musica.union(Set2)

#Usar issubset y issuperset

set(Set_de_musica).issuperset(Set2)

set(Set2).issubset(Set_de_musica)

#sumar elementos de un set y compararlos

A = [1, 2, 2, 1]

B = set([1, 2, 2, 1])

print("the sum of A is:", sum(A))

print("the sum of B is:", sum(B))
```

In [2]:
```
#Crear un set e imprimirlo
set1 = {"pop", "rock", "soul", "hard rock", "rock", "R&B", "rock", "disco"}
set1
```

Out[2]:
```
{'R&B', 'disco', 'hard rock', 'pop', 'rock', 'soul'}
```

In [4]:
```
#Convertir una lista en un set y anadir elementos
Set_de_musica=set(['pop','cumbia','salsa'])
Set_de_musica.add('nerengue')
Set_de_musica
```

Out[4]:
```
{'cumbia', 'nerengue', 'pop', 'salsa'}
```

In [5]:
```
#Buscar un elemento y eliminarlos
'nerengue' in Set_de_musica
Set_de_musica.remove('nerengue')
Set_de_musica
```

Out[5]:
```
{'cumbia', 'pop', 'salsa'}
```

In [8]:
```
#unir dos sets y buscar la interseccion
Set2={'pop','bumbia','regue','ska'}
Set_de_musica,set2
interseccion=Set_de_musica&set2
interseccion
```

Out[8]:
```
{'pop'}
```

In [9]:
```
#Buscar los elementos que estan solo en un set usando difference
Set_de_musica.difference(Set2)
```

Out[9]:
```
{'cumbia', 'salsa'}
```

In [10]:
```
#Unir dos sets
Set_de_musica.union(Set2)
```

Out[10]:
```
{'bumbia', 'cumbia', 'pop', 'regue', 'salsa', 'ska'}
```

In [11]:
```
#Usar issubset y issuperset
set(Set_de_musica).issuperset(Set2)
set(Set2).issubset(Set_de_musica)
```

Out[11]:
```
False
```

In [12]:

```
#sumar elementos de un set y compararlos
A = [1, 2, 2, 1]
B = set([1, 2, 2, 1])
print("the sum of A is:", sum(A))
print("the sum of B is:", sum(B))
```

```
the sum of A is: 6
the sum of B is: 3
```

In [ ]:

```

```

----------Quiz----------

QUESTION 1

1/1 punto (no calificado)

What is the result of the following lines of code:

```
S={'A','B','C'}

U={'A','Z','C'}

U.union(S)
```

- (●) {'A', 'B', 'C', 'Z'}
- ( ) {'A','Z','C'}
- ( ) {'A','B','C'}
- ( ) {'A'}

✔

Guardar

Enviar Ha realizado 1 de 2 intentos

✔ Correcto (1/1 punto)

QUESTION 2

1/1 punto (no calificado)

What is the intersection of set `s` and `u`?

S={'A','B','C'}

U={'A','Z','C'}

- {'A','B','C'}
- {'A','B','C','Z'}
- (•) {'A', 'C'}
- {'Z'}

Guardar

Enviar Ha realizado 1 de 2 intentos

✔ Correcto (1/1 punto)

----------Contenido----------

Los Diccionarios son un tipo de colección, mientras que las listas tienen valores o elementos y cada valor tiene un índice de posición, los diccionarios tienen un valor (elemento) y una llave que es un índice con etiqueta, es decir no tiene que set enteros.

Los diccionarios tienen valores mutables, inmutables y duplicados

Para crear diccionarios, usamos los corchetes curvos igual que los sets, la forma de definirlos es la siguiente:

Diccionario={“llave”:”contenido”}

Mi_Diccionario = {'llave1':20,'llavero2':[5,6,7],'canciones':(“Salsa”, “Cumbia”)}

Para acceder un valor, buscamos su llave dentro del diccionario de la siguiente forma

DICT[“Llavebuscada”], pero si igualamos a algo, es para añadir un nuevo valor al diccionario

Para añadir:

DICT[“Llavedenuevovalor”]=”NuevoValor”, y para eliminar también usamos el índice y el comando del

del(DICT[“Llavedenuevovalor”])

Para buscar un valor en el diccionario, también usamos el comando in

“Llavedenuevovalor” in DICT, devuelve true or false

Para desplegar todas las llaves de un diccionario usamos el método keys()

DICT.keys() y para los valores, el método values() , es decir DICT.values()

----------Jupyter lab / Google Colabs----------

#Crear un diccionario y encontrar un valor

Diccionario={'indice1':10, 'indice2':'Perro'}

Diccionario['indice1']

#Encontrar los indices o llaves del diccionario

Diccionario.keys()

```
In [1]:
1 #Crear un diccionario y encontrar un valor
2 Diccionario={'indice1':10, 'indice2':'Perro'}
3 Diccionario['indice1']
```

```
Out[1]: 10
```

```
In [2]:
1 #Encontrar los indices o llaves del diccionario
2 Diccionario.keys()
```

```
Out[2]: dict_keys(['indice1', 'indice2'])
```

#Crear un diccionario de datos y acceder a sus valores de 2 formas diferentes

Diccionario = {"llave1": 156, "key2": "2", "key3": ['oso', 'lobo', 'camion'], "key4": (4, 4, 4), ('key5'): 5, (0, 'Carlos'): 6}

Diccionario

Diccionario['llave1']

Diccionario[(0,'Carlos')]

#Obtener los valores y las llaves de un diccionario

Diccionario.keys()

Diccionario.values()

#Anadirvalores del diccionario

Diccionario['Pinata'] = 2021

Diccionario['Pinata']

#Buscar un valor en el diccionario

'Pinata' in Diccionario

#Eliminar valores del diccionario

del(Diccionario['Pinata'])

'Pinata' in Diccionario

In [2]:
```
#Crear un diccionario de datos y acceder a sus valores de 2 formas diferente
Diccionario = {"llave1": 156, "key2": "2", "key3": ['oso', 'lobo', 'camion']
Diccionario
Diccionario['llave1']
Diccionario[(0,'Carlos')]
```

Out[2]: 6

In [5]:
```
#Obtener los valores y las llaves de un diccionario
Diccionario.keys()
Diccionario.values()
```

Out[5]: dict_values([156, '2', ['oso', 'lobo', 'camion'], (4, 4, 4), 5, 6])

In [6]:
```
#Anadirvalores del diccionario
Diccionario['Pinata'] = 2021
Diccionario['Pinata']
```

Out[6]: 2021

In [7]:
```
#Buscar un valor en el diccionario
'Pinata' in Diccionario
```

Out[7]: True

In [8]:
```
#Eliminar valores del diccionario
del(Diccionario['Pinata'])
'Pinata' in Diccionario
```

Out[8]: False

----------Quiz----------

QUESTION 1

1/1 punto (no calificado)

Consider the following dictionary:

```
D = {'a':0,'b':1,'c':2}
```

What is the result of the following: `D.values()` ?

- ○ dict_keys([0, 1, 2])
- ◉ dict_values([0, 1, 2])
- ○ dict_values(['a', 'b', 'c'])
- ○ dict_keys(['a', 'b', 'c'])

✔

Guardar

Enviar Ha realizado 1 de 2 intentos

✔ Correcto (1/1 punto)

QUESTION 2

1/1 punto (no calificado)

Consider the following dictionary:

```
D = {'a':0,'b':1,'c':2}
```

What is the output of the following `D['b']`?

- ○ 0
- ◉ 1
- ○ 2

✔

Guardar

Enviar Ha realizado 1 de 2 intentos

✔ Correcto (1/1 punto)

Question 1

1/1 punto (calificado)

What is the syntax used to obtain the first element of the tuple:

A = ('a','b','c')

- ○ A[1]
- ◉ A[0]
- ○ A[:]

✔

Guardar

Enviar Ha realizado 1 de 2 intentos

✔ Correcto (1/1 punto)

Question 2

1/1 punto (calificado)

After applying the following method, L.append(['a','b']), the following list will only be one element longer.

- (●) True
- ( ) False

✔

Enviar Ha realizado 1 de 1 intento

✔ Correcto (1/1 punto)

Question 3

1/1 punto (calificado)

How many duplicate elements can you have in a set?

- ( ) 1
- (●) 0. You can only have one unique element in a set.
- ( ) 100
- ( ) Depends on the number of elements in your set.

✔

Guardar

Enviar Ha realizado 1 de 2 intentos

✔ Correcto (1/1 punto)

Question 4

1/1 punto (calificado)

Consider the following Python dictionary:

```
Dict={"A":1,"B":"2","C":[3,3,3],"D":(4,4,4),'E':5,'F':6}
```

What is the result of the following operation: `Dict["D"]` ?

- ○ 4
- ○ 3
- ○ [3,3,3]
- ◉ (4, 4, 4)
- ○ Error

✔

Guardar

Enviar Ha realizado 1 de 2 intentos

✔ Correcto (1/1 punto)

Question 5

1/1 punto (calificado)

What is an important difference between lists and tuples?

- ○ Lists can't contain a string.
- ○ Tuples can only have integers.
- ○ Lists and tuples are the same.
- ◉ Lists are mutable, tuples are not.
- ○ There are no zeros in lists.

✔

Guardar

Enviar Ha realizado 1 de 2 intentos

✔ Correcto (1/1 punto)

MODULO 3

----------Contenido----------

Conditions and branching

Los operadores comparativos, aplican comparaciones y dan como resultado un boleano, ósea verdadero o falso

El operador de igualdad o equality operator, compara si dos valores son iguales

Valor=6, valor==7, da como resultado false

Valor > 5, da como resultado true, porque valor que vale 6, s[I es mayor que 5

< menor que, > mayor que, >= y <=, mayor igual y menor iguaal, diferente a se escribe !=

- equal: **==**
- not equal: **!=**
- greater than: **>**
- less than: **<**
- greater than or equal to: **>=**
- less than or equal to: **<=**

El branching nos permite poner diferentes condiciones para diferentes entradas

El statement if, nos permite aplicar una condicion y tener una accion como resultado. Ojo con los espacio o las indentation en las condiciones

```
Edad = 20

If (edad>18):

    Print('Eres mayor de 18 anos')

Print('No eres menor de 18 anos')
```

Despues de la condicion se ponen dos puntos :, todo lo que tenga una indentacion despues de la condicion ocurre, lo que esta fuera de la indentacion o espaciado no esta sujeto a la condicion, en nuestro ejemplo el segundo print no esta sujeto a ninguna condicion

Lo mismo ocurre con el else, Tambien lleva los :, y va sin indentacion, lo que pongan despues indentado dentro del else ocurrira si no ocurre la primera condicion del if. Pero si ocurre la condicion del if, lo que esta en el else ya no ocurre

```
Edad = 20

If (edad>18):

    Print('Eres mayor de 18 anos')

Else:

        Print('Eres menor de 18 anos')

Print('Adios')
```

El statement elif, es la abreviacion de else if, el elseif, es una opcion de un if adicional dentro del primer if, pero no es un if anidado

```
Edad = 20

If (edad>18):

    Print('Eres mayor de 18 anos')

If (edad==17):

            Print('Ya te falta un ano')

Else:

        Print('Eres menor de 17 anos')

Print('Adios')
```

Los operadores logicos toman valores boleanos y producen otros valores boleanos

not(verdadero), dara como resultado falso, es decir invierte

El or u o en espanol, toma dos valores los compara y aplica la siguiente regla

A	B	A or B
False	False	False
False	True	True
True	False	Ture
True	True	True

```
Fecha_de_Nacimiento=1990
If (Fecha_de_Nacimiento<2000) or (Fecha_de_Nacimiento>1989)
  Print('naciste antes del ano 2000)
Else:
   Print('No naciste en esa fecha)
```

La condicion de arriba dice que si fecha de Nacimiento es menor a los 2000 o es mayor a 1989, indique que nacio antes del a;o 2000, no tiene sentido el ejemplo.

El operador and o y en espano, es igual al or pero estas son sus reglas

A	B	A & B (AND)
False	False	False
False	True	False
True	False	False
True	True	True

----------Jupyter lab / Google Colabs----------

```
#Utilizar operadores de comparacion
valor=10
valor!=20
#Utilizar branching
```

valor2='Perro'

if(valor2=='Perro'):

  print('Tienes un perro')

else:

  print('No tienes un perro')

#Usar branching con operadores logicos

valor3=30

valor>1 and valor3<40

In [1]:
```
#Utilizar operadores de comparacion
valor=10
valor!=20
```

Out[1]: True

In [2]:
```
#Utilizar branching
valor2='Perro'
if(valor2=='Perro'):
    print('Tienes un perro')
else:
    print('No tienes un perro')
```

```
Tienes un perro
```

In [4]:
```
#Usar branching con operadores logicos
valor3=30
valor>1 and valor3<40
```

Out[4]: True

#Usar los operadores de comparacion

variable = 20

variable == 10

variable > 5

variable != 15

"Carlos" == 'Ramon'

'A'> 'B'

#Toma los valores ascii de las letras para comparar, es casesensitive

#Usar el if, else y elif statement con and y or

fechanacimiento = 1950

sexo= 'Hombre'

```
if (fechanacimiento>1988) and (fechanacimiento<1990):
   print("Naciste en el ano 1989")
elif (fechanacimiento>1978) or (fechanacimiento>1988):
   print("Eres mayor de edad al menos")
else:
   print('Ni naciste en el 1989 ni eres mayor de edad')
print('Gracias por participar')
not(True)
```

In [3]:
```
#Usar los operadores de comparacion
variable = 20
variable == 10
variable > 5
variable != 15
"Carlos" == 'Ramon'
'A'> 'B'
#Toma los valores ascii de las letras para comparar, es casesensitive
```

Out[3]: False

In [8]:
```
#Usar el if, else y elif statement con and y or
fechanacimiento = 1950
sexo= 'Hombre'
if (fechanacimiento>1988) and (fechanacimiento<1990):
    print("Naciste en el ano 1989")
elif (fechanacimiento>1978) or (fechanacimiento>1988):
    print("Eres mayor de edad al menos")
else:
    print('Ni naciste en el 1989 ni eres mayor de edad')
print('Gracias por participar')
not(True)
```

```
Ni naciste en el 1989 ni eres mayor de edad
Gracias por participar
```

Out[8]: False

----------Quiz----------

QUESTION 1

1/1 punto (no calificado)

Select the values of `i` that produces a True for the following:

```
i!=0
```

- [x] 1
- [ ] 0
- [x] -1
- [x] 100000000

✔

Enviar Ha realizado 2 de 2 intentos

✔ Correcto (1/1 punto)

QUESTION 2

1/1 punto (no calificado)

What is the output of the following:

```
x='a'
if(x!='a'):
    print("This is not a.")
else:
    print("This is a.")
```

- ○ "This is not a."
- ● "This is a." ✔

Guardar

Enviar Ha realizado 1 de 2 intentos

✔ Correcto (1/1 punto)

----------Contenido----------

Loops for and while, ciclos for y while, para y mientras

La función range, nos permite crear secuencias en formato lista, hasta alcanzar el valor indicado, por ejemplo:

Ranges(3), da como resultado[1,2,3]

Range(5,10), da como resultado [5,6,7,8,9]

Los loops ejecutan una acción de forma repetida indefinidamente

Suponiendo que tenemos una lista de 5 elementos y queremos encontrar una palabra ahí

```
Mi_lista=['carro','casa','avion','perro','abeja']
For i in range(0,6):
    Mi_lista[i]='perro'
```

En ele ejemplo anterior creamos una variable i, que es incremental y se va a ir comparando con cada valor

Para interactuar con listas y tuples, no se necesita usar índices incrementales, como la variable i.

En este caso, va recorriendo la lista incrementalmente de forma automática

```
colores=’verde’,’rojo’,’azul’
For color in colores
    color
```

En el ejemplo anterior nuestra variable se llama color, y va recorriendo la lista que se llama colores

Podemos usar también el loop for, con índices

```
colores=’verde’,’rojo’,’azul’
for i,color in enumerate(colores)
    color
    i
```

En el ejemplo anterior, la variable i, representa el índice de cada color y color representa el elemento que le corresponde a cada indice. Ademas, usamos la función enumerate(), con la lista como argumento, para referirnos al tamaño de la lista que se recorrerá

Los ciclos while son similares a los ciclos for, pero en vez de ejecutar una operación un determinado nuermo de veces, e; coc;p whiole solo se ejecutar[a hasta que una condición se cumpla

```
Colores=[’verde’,’negro’,’rojo’,’azul’,’amarillo’,’verde’,’verde’]
Soloverdes=[]
Contador=0
While(colores[Contador]==’verde’):
   Soloverdes.append(colores[Contador])
    Contador=Contador+1
```

En el ejemplo primero creamos una lista con elementos y una lista vacia donde solo queremos que se copien los elementos que sean color verde, pornemos un contador en cero y decirmo que mientras el color ubicado en el índice 0 de la lista colores sea igual a verde, añadir ese elemento a la lista solo verdes, después aumentamos el contador, por lo que el índice pasa al siguiente elemento, hasta que la condición no se cumpla y el contador deje de aumentar se termina el ciclo while

----------Jupyter lab / Google Colabs----------

#Crear una lista y usar un loop para imprimir todos los elemento

A=[3,4,5]

```
for i in A:
    print(i)


Lista=[1,2,3,4,5,6]
for contador in Lista:
    print('elemento',contador)
    print(Lista[contador])
#imprimir una secuencia
variable1=10
variable2=1
while (variable2<variable1):
    print(variable2)
    variable2=variable2+1
```

In [1]:

```
#Crear una lista y usar un loop para imprimir todos los elemento
A=[3,4,5]
for i in A:
    print(i)

Lista=[1,2,3,4,5,6]
for contador in Lista:
    print('elemento',contador)
    print(Lista[contador])
```

```
elemento 1
2
elemento 2
3
elemento 3
4
elemento 4
5
elemento 5
6
elemento 6

---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
Cell In [1], line 6
      4 for contador in Lista:
      5     print('elemento',contador)
----> 6     print(Lista[contador])

IndexError: list index out of range
```

In [4]:

```
#imprimir una secuencia
variable1=10
variable2=1
while (variable2<variable1):
    print(variable2)
    variable2=variable2+1
```

```
1
2
3
4
5
6
7
8
```

#Usar range

range(5)

range(1,8)

#Usar el loop for

```
fechas = [1999,2000,2001]
N = len(fechas)

for i in range(N):
    print(fechas[i])

for i in range(0, 8):
    print(i)

for ano in fechas:
    print(ano)
#Usar el loop for con listas
colores = ['rojo', 'amarillo', 'verde', 'morado', 'azul']

for i in range(0, 5):
    print("El color actual ", i, 'es',  colores[i])
    colores[i] = 'blanco'
    print("El color siguente", i, 'es',  colores[i])

colores = ['rojo', 'amarillo', 'verde', 'morado', 'azul']

for contador, color in enumerate(colores):
    print(contador, color)
#Usar el loop while

fechas = [1982, 1980, 1973, 2000]

i = 0
ano = fechas[0]

while(ano != 1973):
    print('el ano es',ano)
```

```
    i = i + 1
    ano = fechas[i]



print("El ciclo recorrio", i ,"veces para encontrar la fecha")
#Ejercicio no l e encuentro el sentido
Evaluaciones = [10, 9.5, 10, 8, 7.5, 5, 10, 10]
i = 0
Evaluacion = Evaluaciones[0]
#Mientras que el contador i sea menor a la longitud de la lista evaluaciones
#Y la evaluacion sea mayor a 6
while(i < len(Evaluaciones) and Evaluacion >= 6):
    print('la calificacion es',Evaluacion)
    i = i + 1
    Evaluacion = Evaluaciones[i]
    i = i + 1
#Otro ejercicio
Colores = ['verde', 'verde', 'azul', 'verde', 'rojo ', 'naranja']
Solo_Verdes = []
i = 0
while(i < len(Colores) and Colores[i] == 'verde'):
    print('Color revisado', Colores[i])
    Solo_Verdes.append(Colores[i])
    i = i + 1
    print(i)
print (Solo_Verdes)
```

In [2]:
```
#Usar range
range(5)
range(1,8)
```

Out[2]:
```
range(1, 8)
```

In [6]:
```
#Usar el loop for
fechas = [1999,2000,2001]
N = len(fechas)

for i in range(N):
    print(fechas[i])

for i in range(0, 8):
    print(i)

for ano in fechas:
    print(ano)
```

```
1999
2000
2001
0
1
2
3
4
5
6
7
1999
2000
2001
```

In [9]:
```
#Usar el loop for con listas
```

In [9]:

```python
#Usar el loop for con listas
colores = ['rojo', 'amarillo', 'verde', 'morado', 'azul']

for i in range(0, 5):
    print("El color actual ", i, 'es',  colores[i])
    colores[i] = 'blanco'
    print("El color siguente", i, 'es',  colores[i])

colores = ['rojo', 'amarillo', 'verde', 'morado', 'azul']

for contador, color in enumerate(colores):
    print(contador, color)
```

```
El color actual  0 es rojo
El color siguente 0 es blanco
El color actual  1 es amarillo
El color siguente 1 es blanco
El color actual  2 es verde
El color siguente 2 es blanco
El color actual  3 es morado
El color siguente 3 es blanco
El color actual  4 es azul
El color siguente 4 es blanco
0 rojo
1 amarillo
2 verde
3 morado
4 azul
```

In [10]:

```
#Usar el loop while

fechas = [1982, 1980, 1973, 2000]

i = 0
ano = fechas[0]

while(ano != 1973):
    print('el ano es',ano)
    i = i + 1
    ano = fechas[i]


print("El ciclo recorrio", i ,"veces para encontrar la fecha")
```

```
el ano es 1982
el ano es 1980
El ciclo recorrio 2 veces para encontrar la fecha
```

In [11]:

```
#Ejercicio no l e encuentro el sentido
Evaluaciones = [10, 9.5, 10, 8, 7.5, 5, 10, 10]
i = 0
Evaluacion = Evaluaciones[0]
#Mientras que el contador i sea menor a la longitud de la lista evaluaciones
#Y la evaluacion sea mayor a 6
while(i < len(Evaluaciones) and Evaluacion >= 6):
    print('la calificacion es',Evaluacion)
    i = i + 1
    Evaluacion = Evaluaciones[i]
    i = i + 1
```

```
la calificacion es 10
la calificacion es 9.5
la calificacion es 8
```

In [15]:

```
#Otro ejercicio
Colores = ['verde', 'verde', 'azul', 'verde', 'rojo ', 'naranja']
Solo_Verdes = []
i = 0
while(i < len(Colores) and Colores[i] == 'verde'):
    print('Color revisado', Colores[i])
    Solo_Verdes.append(Colores[i])
    i = i + 1
    print(i)
print (Solo_Verdes)
```

```
Color revisado verde
1
Color revisado verde
2
['verde', 'verde']
```

----------Quiz----------

QUESTION 1

1/1 punto (no calificado)

What is the output of the following lines of code:

```
A=[3,4,5]

for a in A:

    print(a)
```

- ○ 1, 2, 3
- ○ 0, 1, 2
- ◉ 3, 4, 5

✔

Guardar

Enviar Ha realizado 1 de 2 intentos

✔ Correcto (1/1 punto)

QUESTION 2

1/1 punto (no calificado)

What is the output of the following lines of code:

```
x=3

y=1

while(y!=x):

    print(y)

    y=y+1
```

- (●) 1, 2
- ( ) 0, 1
- ( ) 0

✔

Guardar

Enviar Ha realizado 1 de 2 intentos

✔ Correcto (1/1 punto)

----------**Contenido**----------

Funciones

Las funciones toman una entrada y producen una salida. Las funciones nos permiten hacer mas corto el código, al ser llamadas para ejecutar una acción, las puedes usar un número ilimitado de veces.

Las Funciones de Python o built in functions, no se muestran para saber como funcionan internamente, solamente se llaman.Alguno s ejemplos

Len, que toma como argumentos, una lista o un diccionario o un set, y devuelve como valor la longitud de esa secuencia o colección.

Longitud=len(listaamedirse)

Sum, suma todos los valores de una lista o set, y devuelve la suma

Suma=sum(listaasumarse)

Sorted, devuelve una lista o tuple ordenado

Ordenado=sorted(desordenado)

Pero por ejemplo si usamos el método sort, no creamos un nuevo elemento, sino que organizamos el mismo

Desordenado.sort(), cambia y ahora sus valores internos ya están ordenados, sin necesidad de crear un nuevo elemento que se llame ordenados

Funciones creadas por el programador

La estructura es:

```
def nombre_de_funcion(parámetros a ingresarseo llamadao argumento)
    código indentado y explicación o documentation en triple comillas de apertura y cierre
    """ Aqui va lo que hace mi código
    """
    variables creadas
    return variablecreada
```

ejemploi

```
def aumentador(numero_a_incrementar)
    aumentado = numero_a_incrementar + 1
    return aumentado
```

Para llamarla

Aumentador(10), da como resultado 11. O también Variable = aumentador(5).

Las funciones no almacenan elementos, cada vez que es llamada con un nuevo valor, esta se reinicia

Las funciones pueden tener múltiples parámetros de ingreso o argumentos, van separados por comas y pueden ser dioversos tipos de datos

Las funciones pueden hacer cosas sin devolver un valor o un return statement, en estos casos Python regresa un especial no objeto, es decir la función no devuelve nada. Un ejemplo de esta función puede ser

```
def imprimehola()
  print('hola')
```

cuando llamamos imprimehola(), solo se imprime hola y ya

Las funciones no pueden tener el cuerpo vacio, en esos casos debemos de poner la palabra pass, en el cuerpo de la función, y lo que nos va a devolver una función vacia, es la palabra none

Una función vacia puede ser

```
def No_Hace_Nada()
    pass
    return None
```

función para imprimir los valores de una cadena

```
micandena=[1,2,3,4,5,6]
def impresor(cadena)
    for i,c in enumerat(cadena):
     print('el valor',i,"es",c)
impresor(micadena)
```

Los parámetros variados o variadic, nos permite meter una variable con varios elementos, estas van con asterisco al inicio

La variable nombres es un tuple

```
nombres=('pedro','oscar','juan')
def imprimenombres(*nombres)
for nombre in nombres
print(nombre)
```

El scope de parámetros es una parte del programa que nos permite tener variables accesibles, si la variable se escribe fuera de la función, son variables de acceso global, o global scope, se pueden usar en todo el programa

Las funciones tienen su propio scope, donde sus variables solo pueden ser usadas dentro de si mismas estas son variables locales, sin embargo, las funciones además de sus argumentos de ingreso, pueden usar variables globales para hacer cálculos

Puede haber variables locales y globales con el mismo nombre sin problemas, la global siempre existirá y estará vigente, mientras que la local, solo aparecerá cuando sea llamada la función

```
def agenda()
   fecha=2000
   return(fecha)
fecha=2020
print(agenda())
print(fecha)
```

el resultado de lo anterior imprimirá primero 2000 y después 2020

Para declarar una variable global dentro del scope local de una función ponemos

Global nombredevariable

Nombredevariable= 5500, aunque se termine la funci[on, esa variable conservara a nivel global para todo el programa el ultimo valor asignado dentro de la funcion

----------Jupyter lab / Google Colabs----------

```
#Definir una funcion y llamarla
def funcion(a,b):
    return a+b


a=4
b=2


if a+b == funcion(a,b):
    print("Correct.")
else:
    print("Incorrect.")
#definir y llamar una fucion que sume una cadena
def sumacadena(c):
    return sum(c)


c=[1,2,3,4,5]


if sum(c)==sumacadena(c):
    print("Correct.")
else:
    print("Incorrect.")
```

In [4]:
```
#Definir una funcion y llamarla
def funcion(a,b):
    return a+b

a=4
b=2

if a+b == funcion(a,b):
    print("Correct.")
else:
    print("Incorrect.")
```

```
Correct.
```

In [6]:
```
#definir y llamar una fucion que sume una cadena
def sumacadena(c):
    return sum(c)

c=[1,2,3,4,5]

if sum(c)==sumacadena(c):
    print("Correct.")
else:
    print("Incorrect.")
```

```
Correct.
```

```
#Define una funcion con comentarios
def mifuncion(valor1,valor2):
    """
    Esta función hace una operacion aritmetica
    """
    suma=valor1*valor2
    return(suma)
a=1
b=2
print ("la suma de",a, "+",b, "es",mifuncion(a,b))


mifuncion("carlos",3)
#Obtener información sobre la funcion
help(mifuncion)
#Usar una funcion vacia
def soloimprime():
```

```
    print("esta funcion solo imprime")


soloimprime()
#Usar funciones predefinidas
Cadena=[10,3,5,9,12,654,68]
#funcion print
print("Hola")
#Funcion sum
sum(Cadena)
#funcion len
len(Cadena)
#HAcer una funcion con else if y lopps
def mifuncion(edad, nombre):
    if edad>18:
        return"Eres mayor de edad"
    else:
        return "No eres mayor de edad"
print("Carlos",mifuncion(20,"carlos"))
def imprimelista(lista):
    for elemento in lista:
        print(elemento)

imprimelista(['1', 1, 'the man', "abc"])
#Usar variables globales

artista ="Carlos"
def impresor(artista):
    global variableinterna
    variableinterna="Maluma"
    print(artista,"es un artista")

impresor(artista)
```

```
impresor(variableinterna)
#Usar funciones con argumentops indefinidas
def printDictionary(**args):
    for key in args:
        print(key + " : " + args[key])

printDictionary(Country='Canada',Province='Ontario',City='Toronto')
def anadeelementos(list):
    list.append("Three")
    list.append("Four")

myList = ["One","Two"]

anadeelementos(myList)

myList
```

In [4]:
```python
#Define una funcion con comentarios
def mifuncion(valor1,valor2):
    """
    Esta función hace una operacion aritmetica
    """
    suma=valor1*valor2
    return(suma)
a=1
b=2
print ("la suma de",a, "+",b, "es",mifuncion(a,b))

mifuncion("carlos",3)
```

```
la suma de 1 + 2 es 2
```

Out[4]:
```
'carloscarloscarlos'
```

In [ ]:
```python
#Obtener información sobre la funcion
help(mifuncion)
```

In [6]:
```python
#Usar una funcion vacia
def soloimprime():
    print("esta funcion solo imprime")

soloimprime()
```

```
esta funcion solo imprime
```

In [7]:
```python
#Usar funciones predefinidas
Cadena=[10,3,5,9,12,654,68]
#funcion print
print("Hola")
#Funcion sum
sum(Cadena)
#funcion len
len(Cadena)
```

```
Hola
```

Out[7]:
```
7
```

In [11]:

```python
#HAcer una funcion con else if y lopps
def mifuncion(edad, nombre):
    if edad>18:
        return"Eres mayor de edad"
    else:
        return "No eres mayor de edad"
print("Carlos",mifuncion(20,"carlos"))
```

```
Carlos Eres mayor de edad
```

In [14]:

```python
def imprimelista(lista):
    for elemento in lista:
        print(elemento)

imprimelista(['1', 1, 'the man', "abc"])
```

```
1
1
the man
abc
```

In [16]:

```python
#Usar variables globales

artista ="Carlos"
def impresor(artista):
    global variableinterna
    variableinterna="Maluma"
    print(artista,"es un artista")

impresor(artista)
impresor(variableinterna)
```

```
Carlos es un artista
Maluma es un artista
```

In [17]:
```
#Usar funciones con argumentops indefinidas
def printDictionary(**args):
    for key in args:
        print(key + " : " + args[key])

printDictionary(Country='Canada',Province='Ontario',City='Toronto')

```

```
Country : Canada
Province : Ontario
City : Toronto
```

In [18]:
```
def anadeelementos(list):
    list.append("Three")
    list.append("Four")

myList = ["One","Two"]

anadeelementos(myList)

myList
```

Out[18]:
```
['One', 'Two', 'Three', 'Four']
```

In [ ]:
```

```

----------Quiz----------

QUESTION 1

1/1 punto (no calificado)

What is the value of c after the following block of code is run ?

```
a=1

def add(b):

    return a+b

c=add(10)
```

- ○ 0
- ○ 1
- ◉ 11
- ○ 111

✔

Guardar

Enviar Ha realizado 1 de 2 intentos

✔ Correcto (1/1 punto)

QUESTION 2

1/1 punto (no calificado)

What is the value of `c` after the following block of code is run with proper numerical input?

```
def f(*x):
    return sum(x)
```

- (●) Return the total of a variable amount of parameters.
- ( ) Return the total of a list.
- ( ) The function is not valid.

✔

Enviar Ha realizado 2 de 2 intentos

✔ Correcto (1/1 punto)

----------**Contenido**----------

Objetos y clases

Cada objeto tiene un tipo, una representación interna de datos o un blueprint

Un set de procedimientos para interactuar con ese objeto, estos son los métodos

Un objeto es una instancia de un tipo particular

Por ejemplo cuando creamos un entero, estamos creando una instancia del tipo entero

Objeto1 con valor 1, o objeto1:1 es una instancia del tipo entero

Para conocer el tipo de un objeto usamos el comando type()

Las clases o métodos de tipo (type methods), son funciones que cada instancia de esa clase o tipo provee

Método sort() ordena, método reverse(), ordena alreves

Para crear nuestra propia clase o tipo

Las clases tiene atributos de datos o data attributes y tienes métodos o methods

Creamos unas clases

Class circulo

Data Attributes: Radio, color

Class Rectángulo

Data Attributes, color, base, altura

----------usamos la función _init_, es una función constructora

----------los parámetros van entre paréntesis, el parámetro self representa la instancia recienc creada, pensemos en self como una caja que contiene todos los atributos del objeto

```
class circulo (object):
    def _init(self,radio,color):
            self.radio=radio;
            self.color=color;
class rectangulo (object)
    def _init(self,color, base, altura):
            self.color=color;
            self.base=base;
          self.altura=altura;
```

Ahora creamos un objeto de cada clase

Lo que esta después del igual se llama object contructor

Nombredeobjeto = Nombredelaclase(parámetros)

CirculoRojo= circulo(10,”rojo”)

Para obtener los valores solo ponemos

Circulorojo.radio, esto nos devolverá 10

Para cambiar el valor de un atributo usamos el signo =

CorculoRojo.color=”azul”

Los métodos interactúan con los atributos de los objetos, modificándolos

Dentro de la clase añadimos los métodos como si fueran funciones

```
class circulo (object):
    def _init_(self,radio,color):
            self.radio=radio;
```

```
        self.color=color;
    def añadir_radio(self,r)
        self.radio=self.radio+r
        returno(self.radio)
```

ahora la podemos usar como cualquier otro método

micirculo=circulo(2,”amarillo”)

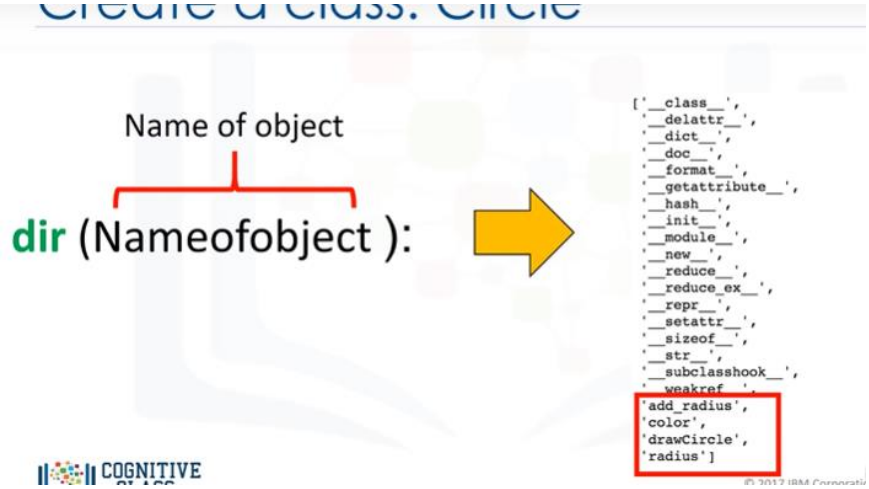
micirculo.añadir_radio(3), me devolverá como resultado un circulo con el radio de 5

Los parámetros después del init, los que no sean self, pueden llevar valores por default, por ejemplo

```
    def _init_(self,radio=3,color=”amarillo”):
```

la función dir, no da la lista de los data attributes de un objeto

dir(nombredelobjeto):



----------Jupyter lab / Google Colabs----------

```
#Ejercicio del carro y sus duenos
#Creamos el objeto
class vehiculo(object):
    def _init_(self,marca,modelo,color):
        self.marca=marca;
        self.modelo=modelo;
        self.color=color;
        self.numero_duenos=0
#Ahora hacemos el metodo par aimrpimir los atributos
    def imprimir_informacion(self):
        print("Carro marca",self.marca)
```

```
        print("Carro modelo",self.modelo)
        print("Carro color",self.color)
        print("Cantidad de duenos",self.numero_duenos)
#Definimos el metodo vender, para aumentar el numero de vendedoreas
    def venta(self):
        self.numero_duenos=numero_duenos+1
#Creamos un nuevo carro e imprimios su atributo
dir(vehiculo)

marca="BMW"
modelo="Serie1"
color="Blanco"

micarro = vehiculo(marca,modelo,color)

micarro.imprimir_informacion()
#Aumentamos sus duenos

for i in range(3):
    micarro.venta()
micarro.imprimir_informacion()
```

In [ ]:

```python
#Ejercicio del carro y sus duenos
#Creamos el objeto
class vehiculo(object):
    def _init_(self,marca,modelo,color):
        self.marca=marca;
        self.modelo=modelo;
        self.color=color;
        self.numero_duenos=0
#Ahora hacemos el metodo par aimrpimir los atributos
    def imprimir_informacion(self):
        print("Carro marca",self.marca)
        print("Carro modelo",self.modelo)
        print("Carro color",self.color)
        print("Cantidad de duenos",self.numero_duenos)
#Definimos el metodo vender, para aumentar el numero de vendedoreas
    def venta(self):
        self.numero_duenos=numero_duenos+1
```

In [ ]:

```python
#Creamos un nuevo carro e imprimios su atributo
dir(vehiculo)

marca="BMW"
modelo="Serie1"
color="Blanco"

micarro = vehiculo(marca,modelo,color)

micarro.imprimir_informacion()
```

In [ ]:

```python
#Aumentamos sus duenos

for i in range(3):
    micarro.venta()
micarro.imprimir_informacion()
```

#Importamos libreria para dibujar objetos

import matplotlib.pyplot as plt

%matplotlib inline

Creamos la clase circulo

class Circulo(object):

  # Constructor

  def __init__(self, radio=3, color='blue'):

    self.radio = radio

```
        self.color = color

    # Creamos el metodo para aumentar radio
    def aumentar_radio(self, r):
        self.radio = self.radio + r
        return(self.radio)

    # Creamos metodo de impresion, los comandos no los conozco bien
    def dibujarCirculo(self):
        plt.gca().add_patch(plt.Circle((0, 0), radio=self.radio, fc=self.color))
        plt.axis('scaled')
        plt.show()
#cCreamos circulos
CirculoRojo=Circulo(10,'Rojo')
#Imprimirmos sus atributos
print('creaste un circulo de radio', CirculoRojo.radio)
print('creaste un circulo de color', CirculoRojo.color)
#Cambiamos el tamano del radio e imprimios el circulo
CirculoRojo.radio=5
CirculoRojo.dibujarCirculo()
CirculoRojo.aumentar_radio(3)
CirculoRojo.dibujarCirculo()
#Ejercicio del texto analizado
class TextoAnalizado(object):

    def __init__ (self, text):
        # Quita signos de puntuacion
        QuitarComas = text.replace('.','').replace('!','').replace('?','').replace(',','')

        # Convierte el texto en minuscula
        TextoenMinuscula = QuitarComas.lower()
```

```
        self.TextoSinPuntos = TextoenMinuscula


    def ContadordePalabras(self):
        # Divide el texto en palabras separadas por espacio
        ListaPalabras = self.QuitarComas.split(' ')


        # Creamos el diccionario de palabras
        DiccionarioPalabras = {}
        for word in set(ListaPalabras): # usamos la funcion set para elimnar palabras duplicadas
            DiccionarioPalabras[word] = ListaPalabras.count(word)


        return DiccionarioPalabras


    def RecorreLista(self,word):
        # get frequency map
        Palabras = self.ContadordePalabras()


        if word in Palabras:
            return Palabras[word]
        else:
            return 0
#Lo probamos
import sys


PalabrasMuestra = {'eirmod': 1,'sed': 1, 'amet': 2, 'diam': 5, 'consetetur': 1, 'labore': 1, 'tempor': 1, 'dolor': 1, 'magna': 2,
'et': 3, 'nonumy': 1, 'ipsum': 1, 'lorem': 2}


def MensajeEmergente(Resultado):
    if Resultado:
        return 'El texto esta aprobado'
    else :
        return 'el texto esta erroneo'
```

```
print("El resultado es: ")
try:
    TextoPrueba = analysedText("Lorem ipsum dolor! diam amet, consetetur Lorem magna. sed diam nonumy eirmod tempor. diam et labore? et diam magna. et diam amet.")
    print(MensajeEmergente(TextoPrueba.TextoSinPuntos == "lorem ipsum dolor diam amet consetetur lorem magna sed diam nonumy eirmod tempor diam et labore et diam magna et diam amet"))
except:
    print("Se detecto un error, revisa el texto " )
print("freqAll: ")
try:
    MapraPalabras = TextoPrueba.ContadordePalabras()
    print(MensajeEmergente(MapaPalabras==PalabrasMuestra))
except:
    print("Se detecto un error, revisa el texto" )
print("freqOf: ")
try:
    Resultado = True
    for word in PalabrasMuestra:
        if TextoPrueba.RecorreLista(word) != PalabrasMuestra[word]:
            Resultado = False
            break
    print(MensajeEmergente(Resultado))

except:
    print("Se detecto un error, revisa el texto " )
#Ejercicio del texto analizado
class analysedText(object):

    def __init__ (self, text):
        # remove punctuation
        formattedText = text.replace('.','').replace('!','').replace('?','').replace(',','')
```

```
        # make text lowercase
        formattedText = formattedText.lower()

        self.fmtText = formattedText

    def freqAll(self):
        # split text into words
        wordList = self.fmtText.split(' ')

        # Create dictionary
        freqMap = {}
        for word in set(wordList): # use set to remove duplicates in list
            freqMap[word] = wordList.count(word)

        return freqMap

    def freqOf(self,word):
        # get frequency map
        freqDict = self.freqAll()

        if word in freqDict:
            return freqDict[word]
        else:
            return 0
#Lo probamos
import sys

sampleMap = {'eirmod': 1,'sed': 1, 'amet': 2, 'diam': 5, 'consetetur': 1, 'labore': 1, 'tempor': 1, 'dolor': 1, 'magna': 2, 'et': 3,
'nonumy': 1, 'ipsum': 1, 'lorem': 2}

def testMsg(passed):
    if passed:
```

```
    return 'Test Passed'
  else :
    return 'Test Failed'


print("Constructor: ")
try:
  samplePassage = analysedText("Lorem ipsum dolor! diam amet, consetetur Lorem magna. sed diam nonumy eirmod tempor. diam et labore? et diam magna. et diam amet.")
  print(testMsg(samplePassage.fmtText == "lorem ipsum dolor diam amet consetetur lorem magna sed diam nonumy eirmod tempor diam et labore et diam magna et diam amet"))
except:
  print("Error detected. Recheck your function " )
print("freqAll: ")
try:
  wordMap = samplePassage.freqAll()
  print(testMsg(wordMap==sampleMap))
except:
  print("Error detected. Recheck your function " )
print("freqOf: ")
try:
  passed = True
  for word in sampleMap:
    if samplePassage.freqOf(word) != sampleMap[word]:
      passed = False
      break
  print(testMsg(passed))

except:
  print("Error detected. Recheck your function  " )
```

In [ ]:
```
#Importamos libreria para dibujar objetos

import matplotlib.pyplot as plt
%matplotlib inline
```

In [ ]:
```
# Creamos la clase circulo

class Circulo(object):

    # Constructor
    def __init__(self, radio=3, color='blue'):
        self.radio = radio
        self.color = color

    # Creamos el metodo para aumentar radio
    def aumentar_radio(self, r):
        self.radio = self.radio + r
        return(self.radio)

    # Creamos metodo de impresion, los comandos no los conozco bien
    def dibujarCirculo(self):
        plt.gca().add_patch(plt.Circle((0, 0), radio=self.radio, fc=self.color))
        plt.axis('scaled')
        plt.show()
```

In [ ]:
```
#cCreamos circulos
CirculoRojo=Circulo(10,'Rojo')
#Imprimirmos sus atributos
print('creaste un circulo de radio', CirculoRojo.radio)
print('creaste un circulo de color', CirculoRojo.color)
```

In [ ]:
```
#Cambiamos el tamano del radio e imprimios el circulo
CirculoRojo.radio=5
CirculoRojo.dibujarCirculo()
CirculoRojo.aumentar_radio(3)
CirculoRojo.dibujarCirculo()
```

In [ ]:
```
#Ejercicio del texto analizado
class TextoAnalizado(object):

    def __init__ (self, text):
        # Quita signos de puntuacion
        QuitarComas = text.replace('.','').replace('!','').replace('?','').replace(',','')

        # Convierte el texto en minuscula
        TextoenMinuscula = QuitarComas.lower()

        self.TextoSinPuntos = TextoenMinuscula

    def ContadordePalabras(self):
        # Divide el texto en palabras separadas por espacio
        ListaPalabras = self.QuitarComas.split(' ')

        # Creamos el diccionario de palabras
        DiccionarioPalabras = {}
        for word in set(ListaPalabras): # usamos la funcion set para elimnar palabras duplicadas
            DiccionarioPalabras[word] = ListaPalabras.count(word)

        return DiccionarioPalabras

    def RecorreLista(self,word):
        # get frequency map
        Palabras = self.ContadordePalabras()

        if word in Palabras:
            return Palabras[word]
        else:
            return 0
```

In [ ]:
```
#Lo probamos
import sys

PalabrasMuestra = {'eirmod': 1,'sed': 1, 'amet': 2, 'diam': 5, 'consetetur': 1, 'labore': 1, 'tempor': 1, 'dolor': 1, 'magna': 2

def MensajeEmergente(Resultado):
    if Resultado:
       return 'El texto esta aprobado'
    else :
       return 'el texto esta erroneo'

print("El resultado es: ")
try:
    TextoPrueba = analysedText("Lorem ipsum dolor! diam amet, consetetur Lorem magna. sed diam nonumy eirmod tempor. diam et lab
    print(MensajeEmergente(TextoPrueba.TextoSinPuntos == "lorem ipsum dolor diam amet consetetur lorem magna sed diam nonumy eir
except:
    print("Se detecto un error, revisa el texto " )
print("freqAll: ")
try:
    MapraPalabras = TextoPrueba.ContadordePalabras()
    print(MensajeEmergente(MapaPalabras==PalabrasMuestra))
except:
    print("Se detecto un error, revisa el texto" )
print("freqOf: ")
try:
    Resultado = True
    for word in PalabrasMuestra:
        if TextoPrueba.RecorreLista(word) != PalabrasMuestra[word]:
            Resultado = False
            break
    print(MensajeEmergente(Resultado))

except:
    print("Se detecto un error, revisa el texto " )
```

In [ ]:
```
#Ejercicio del texto analizado
class analysedText(object):

    def __init__ (self, text):
        # remove punctuation
        formattedText = text.replace('.','').replace('!','').replace('?','').replace(',','')

        # make text lowercase
        formattedText = formattedText.lower()

        self.fmtText = formattedText

    def freqAll(self):
        # split text into words
        wordList = self.fmtText.split(' ')

        # Create dictionary
        freqMap = {}
        for word in set(wordList): # use set to remove duplicates in list
            freqMap[word] = wordList.count(word)

        return freqMap

    def freqOf(self,word):
        # get frequency map
        freqDict = self.freqAll()

        if word in freqDict:
            return freqDict[word]
        else:
            return 0
```

```
: #Lo probamos
import sys

sampleMap = {'eirmod': 1,'sed': 1, 'amet': 2, 'diam': 5, 'consetetur': 1, 'labore': 1, 'tempor': 1, 'dolor': 1, 'magna': 2, 'et'

def testMsg(passed):
    if passed:
       return 'Test Passed'
    else :
       return 'Test Failed'

print("Constructor: ")
try:
    samplePassage = analysedText("Lorem ipsum dolor! diam amet, consetetur Lorem magna. sed diam nonumy eirmod tempor. diam et l
    print(testMsg(samplePassage.fmtText == "lorem ipsum dolor diam amet consetetur lorem magna sed diam nonumy eirmod tempor dia
except:
    print("Error detected. Recheck your function " )
print("freqAll: ")
try:
    wordMap = samplePassage.freqAll()
    print(testMsg(wordMap==sampleMap))
except:
    print("Error detected. Recheck your function " )
print("freqOf: ")
try:
    passed = True
    for word in sampleMap:
        if samplePassage.freqOf(word) != sampleMap[word]:
            passed = False
            break
    print(testMsg(passed))

except:
    print("Error detected. Recheck your function  " )
```

----------Quiz----------

QUESTION 1

1/1 punto (no calificado)

Using the Class Car in the lab, create a Car object with the following attributes:

make="Honda"

model="Accord"

color="blue"

- [x] Car(make="Honda",model="Accord",color="blue")
- [x] Car("Honda","Accord","blue")
- [x] Car(model="Accord",make="Honda",color="blue")
- [ ] Car("Accord","Honda","blue")

Note: Make sure you select all of the correct options—there may be more than one!

Enviar Ha realizado 2 de 2 intentos

Correcto (1/1 punto)

QUESTION 2

1/1 punto (no calificado)

From the lab, how would you change the data attribute `owner_number` ?

- (●) Utilising the method `sell()` .
- ( ) Utilising the method `car_info()` .

✔

Enviar Ha realizado 2 de 2 intentos

✔ Correcto (1/1 punto)

Question 1

1/1 punto (calificado)

What is the output of the following lines of code:

```
x=1

if(x!=1):

        print('Hello')

else:

        print('Hi')

print('Mike')
```

- (●) Hi Mike
- ( ) Mike
- ( ) Hello Mike
- ( ) The Mike

✔

Guardar

Enviar Ha realizado 1 de 2 intentos

✔ Correcto (1/1 punto)

Question 2

1/1 punto (calificado)

What is the output of the following few lines of code?

```
A = ['1','2','3']

for a in A:

    print(2*a)
```

- ○ 2 4 6
- ○ '2' '4' '6'
- ◉ '11' '22' '33'
- ○ A B C

✔

Enviar Ha realizado 2 de 2 intentos

✔ Correcto (1/1 punto)

Question 3

1/1 punto (calificado)

Consider the function Delta, when will the function return a value of 1

```
def Delta(x):
    if x==0:
        y=1;
    else:
        y=0;
    return(y)
```

- ○ When the input is anything but 0.
- ○ When the input is 1.
- ○ Never.
- ◉ When the input is 0. ✔

Enviar Ha realizado 2 de 2 intentos

✔ Correcto (1/1 punto)

Question 4

0/1 punto (calificado)

What is the correct way to sort the list 'B' using a method? The result should not return a new list, just change the list 'B'.

- (●) B.sort()
- ( ) sort(B)
- ( ) sorted(B)
- ( ) B.sorted()

Enviar Ha realizado 2 de 2 intentos

✖ Incorrecto (0/1 punto)

Question 5

1/1 punto (calificado)

What are the keys of the following dictionary: `{'a':1,'b':2}` ?

- ( ) 1,2
- ( ) ;;:
- (●) a,b

✔

Guardar

Enviar Ha realizado 1 de 2 intentos

✔ Correcto (1/1 punto)

Modulo 4

----------Contenido----------

----------Jupyter lab / Google Colabs----------

----------Quiz----------