	Manual de prácticas del Laboratorio de Estructuras de datos y algoritmos I	Código:	MADO-19
		Versión:	02
		Página	1/11
		Sección ISO	8.3
		Fecha de emisión	25 de enero de 2019
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

Profesor: Marco Antonio Martínez

Asignatura: Estructura de datos y algoritmos 1

Grupo: 17


No de Práctica(s): 9 Introducción a Python (I).

Alumno: Carvajal Axol Brandon Emir

Semestre: 2020-2

Observaciones:

CALIFICACIÓN: _____

	Manual de prácticas del Laboratorio de Estructuras de datos y algoritmos I	Código:	MADO-19
		Versión:	02
		Página	2/11
		Sección ISO	8.3
		Fecha de emisión	25 de enero de 2019
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

Objetivo:

Aplicar las bases del lenguaje de programación Python en el ambiente de Jupyter notebook.

Variables y tipos

- Los nombres de las variables son alfanuméricos (a-z, A-Z, 0-9) y empiezan con una letra en minúscula.
- No se especifica el tipo de valor que una variable contiene, está implícito al momento de asignar un valor.
- No se necesita poner; al final de cada instrucción.
- Mantener las indentaciones al momento de escribir código.

Nombres reservados en Python


and, as, assert, break, class, continue, def, del, elif, else, except, exec, finally, for, from, global, if, import, in, is, lambda, not, or, pass, print, raise, return, try, while, with, yield.

```
Ej1Pract9.py
1 x=10
2 print(x)
3 cadena= "Holamundo"
4 print(cadena)
5 x=y=z=10
6 print(x,y,z)
7 type(x)
8 type(cadena)
```

```
10
Holamundo
10 10 10

Process finished with exit code 0
```

En este ejercicio podemos ver como se inician variables de tipo int y str, al igual como se puede asignar un valor a tres variables .

	Manual de prácticas del Laboratorio de Estructuras de datos y algoritmos I	Código:	MADO-19
		Versión:	02
		Página	3/11
		Sección ISO	8.3
		Fecha de emisión	25 de enero de 2019
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

Las cadenas pueden ser definidas usando comilla simple (') o comilla doble ("). Una característica especial de las cadenas es que son inmutables, esto quiere decir que no se pueden cambiar los caracteres que contiene. El caracter \ sirve para escapar caracteres como \n o \t.

```

1  cadena1='Hola'
2  cadena2="Mundo"
3  print(cadena1)
4  print(cadena2)
5  concat_cadenas=cadena1+cadena2
6  print(concat_cadenas)
7

```

```

C:\Users\rosal\OneDrive\Desktop\python\
Hola
Mundo
HolaMundo

Process finished with exit code 0

```

Podemos observar la concatenación de dos cadenas, formando la oración completa "HolaMundo".

```

1  cadena1='Hola'
2  cadena2="Mundo"
3  print(cadena1)
4  print(cadena2)
5  concat_cadenas=cadena1+cadena2
6  print(concat_cadenas)
7
8  num_cadena="{1}{2}{3}".format(cadena1, cadena2, 3)
9  print(num_cadena)

```

```

Hola
Mundo
HolaMundo
HolaMundo3

Process finished with exit code 0

```

Para concatenar cadenas se usa de la función format() en el ejercicio se puede ver en el resultado "HolaMundo3".

```

1  cadena1='Hola'
2  cadena2="Mundo"
3  print(cadena1)
4  print(cadena2)
5  concat_cadenas=cadena1+cadena2
6  print(concat_cadenas)
7
8  num_cadena="cambiando el orden: {1} {2} {0} #" .format(cadena1, cadena2, 3)
9  print(num_cadena)

```


```

Hola
Mundo
HolaMundo
cambiando el orden: Mundo 3 Hola #

Process finished with exit code 0

```

Por medio de la función format, se puede cambiar el orden en que se imprimen las variables

	Manual de prácticas del Laboratorio de Estructuras de datos y algoritmos I	Código:	MADO-19
		Versión:	02
		Página	4/11
		Sección ISO	8.3
		Fecha de emisión	25 de enero de 2019
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

Operadores

Aritméticos: +, -, *, /


<pre> 1 print(1+5) 2 print(6*3) 3 print(10-4) 4 print(100/50) 5 print(10%2) 6 print(((20*3)+(10+1))/10) 7 print(2**2) </pre>	<pre> 6 18 6 2.0 0 7.1 4 </pre>
--	---------------------------------

Se muestra el código con el uso de los operadores +, -, *, / además del mod en la línea 5.

Comparación: >, <, >=, <=, ==

<pre> 1 print(7<5) 2 print(7>5) 3 print((11*3)+2 == 36-1) 4 print((11*3)+2 == 36) 5 print("curso"!= "CuRso") </pre>	<pre> False True True False True </pre>
---	---

Se muestra el código con el uso de los operadores >, <, >=, <=, == dando como resultado falso o verdadero (false, true).

	Manual de prácticas del Laboratorio de Estructuras de datos y algoritmos I	Código:	MADO-19
		Versión:	02
		Página	5/11
		Sección ISO	8.3
		Fecha de emisión	25 de enero de 2019
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

Listas

- Básicamente son valores que están separados por comas dentro de paréntesis cuadrados.
- Está compuesta por cualquier cantidad y/o tipo de datos, ya sean cadenas, caracteres, números e inclusive otras listas.
-
- Se puede acceder a las listas por medio de índices, estos índices comienzan desde 0 hasta el número de elementos menos 1.
- Las listas son mutables.

Ejercicio de lista simple.

```
1 lista_diasdelmes=(31,28,31,30,31,30,31,31,30,31,30,31)
2
3 print(lista_diasdelmes)
4 print(lista_diasdelmes[0])
5 print(lista_diasdelmes[6])
6 print(lista_diasdelmes[11])
```

```
(31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31)
31
31
31
Process finished with exit code 0
```

En las líneas 4,5,6 hay corchetes que indican lo que se mandara a imprimir en este caso se selecciono el lugar donde el valor del día es 31.

Ejercicio de lista anidada.

```
1 lista_numeros=[['cero',0],['uno',1,'UNO'],['dos',2],['tres',3],['cuatro',4],['X',5]]
2
3 print(lista_numeros)
4
5 print(lista_numeros[0])
6 print(lista_numeros[1])
7
8 print(lista_numeros[2][0])
9 print(lista_numeros[2][1])
10 print(lista_numeros[1][0])
11
12 print(lista_numeros[1][1])
13 print(lista_numeros[1][2])
```


```
[['cero', 0], ['uno', 1, 'UNO'], ['dos', 2], ['tres', 3], ['cuatro', 4], ['X', 5]]
['cero', 0]
['uno', 1, 'UNO']
dos
2
uno
1
UNO
```

En las líneas 8,9,10,12 y 13 hay pares de corchetes que localizan el lugar del dato guardado a imprimir.

```
14 lista_numeros[5][0] = 'cinco'
15 print(lista_numeros[5])
```

```
['cinco', 5]
Process finished with exit code 0
```

En la línea 14 se cambia el valor "X" a "cinco" y se manda a imprimir en la línea 15.

	Manual de prácticas del Laboratorio de Estructuras de datos y algoritmos I	Código:	MADO-19
		Versión:	02
		Página	6/11
		Sección ISO	8.3
		Fecha de emisión	25 de enero de 2019
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

Tuplas

- Son parecidas a las listas, valores separados por una coma.
- Comparadas con las listas, las tuplas no son mutables.
- Se pueden aplicar las mismas operaciones que en las listas y su ventaja es que consumen menos memoria para almacenarse.
- Se crean, ya sea utilizando paréntesis o simplemente separando los valores por comas.

Ejercicio tupla.

```

1 tupla_diasdelmes=(31,28,31,30,31,30,31,31,30,31,30,31)
2
3 print(tupla_diasdelmes)
4 print(tupla_diasdelmes[0])
5 print(tupla_diasdelmes[3])
6 print(tupla_diasdelmes[1])

```

```

(31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31)
31
30
28
Process finished with exit code 0

```

En la línea 3 se imprime la tupla completa y en las líneas 4, 5, 6 se manda a imprimir elementos específicos en posición definida de la tupla.

Ejercicio tupla anidada.

```

1 tupla_numeros= (('cero',0),('uno',1,'UNO'),('dos',2),('tres',3),('cuatro',4),('X',5))
2
3 print(tupla_numeros)
4 print(tupla_numeros[0])
5 print(tupla_numeros[1])
6 print(tupla_numeros[2][0])
7 print(tupla_numeros[2][1])
8 print(tupla_numeros[1][0])
9 print(tupla_numeros[1][1])
10 print(tupla_numeros[1][2])

```

```

(('cero', 0), ('uno', 1, 'UNO'), ('dos', 2), ('tres', 3), ('cuatro', 4), ('X', 5))
('cero', 0)
('uno', 1, 'UNO')
dos
2
uno
1
UNO

```

En la línea 3 se imprime la tupla completa, línea 4 y 5 se manda a imprimir todos los elementos en posición definida de la tupla, de la línea 6 a la 10 se imprimen elementos específicos en las posiciones definidas en los corchetes.

Ejercicio mutabilidad lista y tupla.

```

14 print("valor actual {}".format(lista_diasdelmes[0]))
15 lista_diasdelmes[0]=50
16 print("valor cambiado {}".format(lista_diasdelmes[0]))
17 tupla_diasdelmes[0]=50


```

```

valor actual 31
valor cambiado 50
Traceback (most recent call last):
  File "C:/Users/rosarioxol/Desktop/python/Ej1Pract9.py", line 17, in <module>
    tupla_diasdelmes[0]=50
TypeError: 'tuple' object does not support item assignment
Process finished with exit code 1

```

En la línea 15 cambia de valor la lista y la línea 17 nos manda un error ya que no se pueden cambiar los valores de las tuplas.

	Manual de prácticas del Laboratorio de Estructuras de datos y algoritmos I	Código:	MADO-19
		Versión:	02
		Página	7/11
		Sección ISO	8.3
		Fecha de emisión	25 de enero de 2019
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

Tupla con nombre

Este tipo especial de tuplas permite especificar un nombre para describirla.

```

1  from collections import namedtuple
2
3  planeta = namedtuple('planeta', ['nombre', 'numero'])
4  planeta1 = planeta('mercurio', 1)
5  print(planeta1)
6  planeta2 = planeta('venus', 2)
7
8  print(planeta1.nombre, planeta1.numero)
9  print(planeta2[0], planeta2[1])
10 print('campos de la tupla: {}'.format(planeta1._fields))

```

```

planeta(nombre='mercurio', numero=1)
mercurio 1
venus 2
campos de la tupla: ('nombre', 'numero')

Process finished with exit code 0

```

En la primera línea se importa la librería para ocupar “namedtuple”.

Línea 3 se crea una tupla con nombre que contiene dos argumentos, el primero es el nombre de la tupla y el segundo son los campos.

Línea 4 y 6 se crea planeta 1 y 2 agregando valores a sus campos

Línea 8 y 9 se imprimen los valores de los campos de manera diferente, uno es con nombre de los campos y el otro por orden de los campos.

Línea 10 se imprime nombres de los campos.

Diccionarios

- Un diccionario se crea usando { } y consta de dos partes: llave y valor.
- Las llaves son inmutables, deben de tener un solo tipo de dato, una cadena o número. Una vez que es creado, no se puede cambiar su tipo.
- Mientras que el valor puede ser de cualquier tipo y se puede cambiar con el tiempo.
- Los elementos en un diccionario no están ordenados.

```

1  elementos = {'hidrogeno':1, 'helio':2, 'carbon':6}
2  print(elementos)
3  print(elementos['hidrogeno'])

```

```

{'hidrogeno': 1, 'helio': 2, 'carbon': 6}
1

```

Línea 1 se crea diccionario, en la línea 2 se imprime, en la línea 3 se imprime el elemento hidrogeno, que es el 1.

```

5  elementos['litio']=3
6  elementos['nitrogeno']=8
7  print(elementos)


```

```

{'hidrogeno': 1, 'helio': 2, 'carbon': 6, 'litio': 3, 'nitrogeno': 8}

```

Línea 5 y 6 se agregan nuevos elementos, en la línea 7 se mandan a imprimir (no están ordenados).

	Manual de prácticas del Laboratorio de Estructuras de datos y algoritmos I	Código:	MADO-19
		Versión:	02
		Página	8/11
		Sección ISO	8.3
		Fecha de emisión	25 de enero de 2019
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

```

9      elementos2 = {}
10     elementos2['H'] = {'name': 'Hydrogen', 'number': 1, 'weight': 1.00794}
11     elementos2['He'] = {'name': 'Helium', 'number': 2, 'weight': 4.002602}
12     print(elementos2)

```

```

{'H': {'name': 'Hydrogen', 'number': 1, 'weight': 1.00794}, 'He': {'name': 'Helium', 'number': 2, 'weight': 4.002602}}

```

Línea 9 creamos un nuevo diccionario, línea 10 y 11 son datos que van en el diccionario y en la línea 12 se manda a imprimir el nuevo diccionario (elementos2).

```

14     print(elementos2['H'])
15     print(elementos2['H']['name'])
16     print(elementos2['H']['number'])
17     elementos2['H']['weight'] = 4.30
18     print(elementos2['H']['weight'])

```

```

{'name': 'Hydrogen', 'number': 1, 'weight': 1.00794}
Hydrogen
1
4.3

```

Línea 14 a 16 se imprime datos del hidrógeno guardados en el segundo diccionario, en la línea 17 se cambia el valor de weight a 4.3, se manda a imprimir nuevo valor en la línea 18.

```

20     elementos2['H'].update({'gas noble': True})
21     print(elementos2['H'])

```

```

{'name': 'Hydrogen', 'number': 1, 'weight': 4.3, 'gas noble': True}

```

Línea 20 se agrega elemento a una llave y se imprimen.

```

23     print(elementos2.items())
24     print(elementos2.keys())


```

```

dict_items([('H', {'name': 'Hydrogen', 'number': 1, 'weight': 4.3, 'gas noble': True}),
            ('He', {'name': 'Helium', 'number': 2, 'weight': 4.002602})])
dict_keys(['H', 'He'])

```

Línea 23 muestra todos los elementos del diccionario y la línea 24 muestra todas las llaves del diccionario.

	Manual de prácticas del Laboratorio de Estructuras de datos y algoritmos I	Código:	MADO-19
		Versión:	02
		Página	9/11
		Sección ISO	8.3
		Fecha de emisión	25 de enero de 2019
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

Funciones

- Una función o procedimiento sirve para empaquetar código que sirve para ser reutilizado.
- Se puede usar ese mismo código con diferentes entradas y obtener resultados o comportamiento de acuerdo con esos datos.

```

1 def imprime_nombre(nombre):
2     print("hola "+nombre)
3     imprime_nombre("JJ")

```

hola JJ

Línea 1 se crea la función "imprime_nombre", línea 2 se concatena la cadena con "+", línea 3 se llama la función "imprime_nombre".

```

5 def cuadrado(x):
6     return x**2
7     x=5
8     print("El cuadrado de {} es {}".format(x, cuadrado(x)))

```

El cuadrado de 5 es 25

Línea 5 se define una función que eleva al cuadrado un número, en la línea 8 se usa la función format() para convertir parámetros en cadenas, así imprimiendo el cuadrado de 5.


```

10 def varios(x):
11     return x**2, x**3, x**4
12     val1, val2, val3 = varios(2)
13     print("{} {} {}".format(val1, val2, val3))

```

4 8 16

Línea 10 se define una función que regresa más de un valor, línea 12 se escriben las variables donde se guardarán los valores que regresará la función y el número para obtener los valores es "2", línea 13 se imprimen los valores.

	Manual de prácticas del Laboratorio de Estructuras de datos y algoritmos I	Código:	MADO-19
		Versión:	02
		Página	10/11
		Sección ISO	8.3
		Fecha de emisión	25 de enero de 2019
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

Variables globales

Hay dos tipos de espacio de nombres, el primero es el espacio global y el segundo el espacio local. Las variables que se declaren afuera de las funciones pertenecen al espacio global y no se necesita añadir un modificador para declararlas de esta manera.

Por otro lado, todas las variables que se definen dentro de una función pertenecen al espacio local, estas variables sólo pueden ser reconocidas y usadas dentro de la propia función.

```

1  vg = 'Global'
2  def funcion_v1():
3      print(vg)
4      funcion_v1()
5      print(vg)

```

```

Global
Global

```

Línea 1 se crea la variable “vg”, línea 2 se crea una función que imprime la variable, línea 4 se llama la función y en la línea 5 se imprime la variable global.

```

7  def funcion_v2():
8      vg = "Local"
9      print(vg)
10     funcion_v2()
11     print(vg)

```

```

Local
Global

```

Línea 7 se crea una segunda función, línea 8 se crea una variable con el mismo nombre que la variable global, línea 10 se llama la función e imprime la variable local, línea 11 se imprime variable global.

```

13 def funcion_v3():
14     print(vg)
15     vg="Local"
16     print(vg)
17     funcion_v3()

```

```

Traceback (most recent call last):
  File "C:/Users/rosarioaxol/Desktop/python/variables_globales.py", line 17, in <module>
    funcion_v3()
  File "C:/Users/rosarioaxol/Desktop/python/variables_globales.py", line 14, in funcion_v3
    print(vg)
UnboundLocalError: local variable 'vg' referenced before assignment

```

Línea 13 se crea una función para imprimir variable global, se creó una variable con el mismo nombre que la global, por lo cual era remplazada, se tiene una variable local sin valor, al momento de imprimir manda un error

```

15 def funcion_v4():
16     global vg
17     print(vg)
18     vg="Local"
19     print(vg)
20     funcion_v4()
21     print(vg)


```

```

Global
Local
Local

```

Línea 15 se crea nueva función para resolver el error de la función_v3, para esto se reserva la palabra global, así se imprime la variable con su valor antes de modificarse por la función y después de reasignar valor se imprime nuevamente.

	Manual de prácticas del Laboratorio de Estructuras de datos y algoritmos I	Código:	MADO-19
		Versión:	02
		Página	11/11
		Sección ISO	8.3
		Fecha de emisión	25 de enero de 2019
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

Bibliografía

Tutorial oficial de Python: <https://docs.python.org/3/tutorial/>

Galería de notebooks: <https://wakari.io/gallery>