First Exam 1/03/2023 – (Semana 3 – Elementos Actualizados)

1. **Variable:**
   Es un nombre que hace referencia a cualquier tipo de dato, que contiene el valor de ese dato. Son:
   a) Numéricas (enteros, decimales)
   b) Que almacenan texto (carácter strings)
   c) Indexación (str.[posición])
   d) Que almacenan valores lógicos (booleanos, Verdadero o Falso)
2. **Operadores** Operaciones (+, -, * , /, //, %, **)
   a) De asignación (=) de contenido a una variable
   b) Numéricos (suma, resta, multiplicación, etc.)
   c) Que operan sobre variables de texto (concatenación, repetición) dan otras cadenas de texto.
   d) De comparación (mayor, menor, igual): dan resultado valores lógicos (Verdadero o Falso)
3. **Funciones**
   print() -- imprime valores en pantalla
   help() -- documentacion sobre funciones, metodos, etc.
   len() -- Devuelve la longitud (length) de cadena de caracteres
   type() -- Devuelve variable (int, str, float, bool)
   abs() -- Devuelve valor *absoluto* de un número
   round() -- R*edondea* número
   str() -- convierte una variable (a un string/cadena)
   int() -- convierte una variable a un entero
   float() -- convierte una variable a numero decimal
   bool() -- convierte variable a variable lógica
4. **Métodos:** Varios que actúan sobre cadenas de caracteres
   se escribe var.method()
   1. var.isdigit()
   2. var.toupper()
   3. var.find("caracter")
   4. (strip(), upper(), lower(), replace(), etc)
5. **f-strings:** Formatos de cadenas
6. **Elementos Varios**

Operaciones lógicas (or, and, not) Códigos de escape (\n, \t, \', \", \, \r)

Listas

Son tipos datos para guardar colecciones ordenadas de valores. Se definen con [ ]

Cada elemento tiene índice que refiere a su posición. Las listas se indexan c/núm. enteros, comenzando en 0.

print(frutas[0]) - print(frutas[-1]) -

Desde Excel los valores podemos separarlos usando método `split(',')`: **fila = line.split(',')**

frutas.**append**('cerezas') Para agregar a la lista

frutas.**insert**(3,'sandía') Para agregar en determinad posición

frutas[2] = 'melón' Para cambiar elemento de posición 2 No se puede hacer con una cadena

frutas.**remove**('naranjas') Borrar Naranjas

del frutas[1] Borramos pera en la posición 1 Al borrar no queda un hueco. Los elementos se moverán p/ llenar el vacío. Si hay + de un valor, remove() sólo sacará la primera aparición.

frutas **+** verduras **Concatenación** de 2 listas IDEM CON LOS NUMEROS – NO SUMAN

verduras ***** 3 replica la lista 3 veces IDEM CON LOS NUMEROS – NO MULTIPLICAN

len(frutas) Cuenta la cantidad de elementos len(a + b) Cuenta elementos de listas a y b

s = [10, 1, 7, 3] s.sort() **Ordena** la lista s s.sort(reverse=True) Ordena al reves

t = sorted(s) Ordena s en nueva lista t

in, not in "melon" in frutas - "melon" not in frutas -

frutas.**index**("bananas") Para encontrar rápido un elemento Resultado posición 2 Si hay 2 Devuelve la 1ra

Iteraciones

FOR - PARA - Ejecuta una vez por cada elemento. Range no almacena valores, sino que los va generando.

Es mejor usar **enumerate**() en lugar de range(len(secuencia))

for i in [0,1,2]:
  print(i)
	for i in range(3):
	  print(i) # 1er Rango en 0
		for x in range(-3,3):
		print(x)
			frutas = ['manz', 'peras', 'banana', 'naranja']
			for f in frutas:
			  print(f)

0, 1, 2	0, 1, 2	-3, -2, -1, 0, 1, 2	manz, peras, banana. naranja

Cada iteración del bucle, asigna ordenadamente un nuevo elemento a la variable **f**. El valor previo es sobre escrito, y al final retiene su último valor.

WHILE – MIENTRAS - Ejecuta mientras que cierta condición se cumpla. Cuando la condición no se cumple más finaliza

i = 0
while i < 5:
  print(i)
  i = i + 1
print("listo, sali del ciclo")
	while num_billetes * grosor_billete <= altura_obelisco:
	  print(dia, num_billetes, num_billetes * grosor_billete)
	  dia = dia + 1
	  num_billetes = num_billetes * 2
	print('Cantidad de días', dia)
	print('Cantidad de billetes', num_billetes)
	print('Altura final', num_billetes * grosor_billete)

Imprime del 0 al 4	Imprime cantidad de días, cant. billetes y altura por día

Los comandos indentados debajo del **`while`** se van a ejecutar mientras que **`while`** sea verdadera (**`True`**).

Indentación: Para marcar grupos de comandos que van juntos.

Condicionales: **IF - ELSE - ELIF** Solo se ejecuta si se cumple la condición. Si no se cumple ELSE (si no)

a = 7
b = 5
if a > b:
  print('Gana a')
else:
  print('Gana b')
	a = 3
	b = 5
	if a > b:
	  print('Gana a')
	elif a == b:
	  print('Empate!')
	else:
	  print('Gana b')
		a = 1
		b = 2
		c = 3
		if a < b **or** a < c:
		  print('a no es el más grande')
			if a > b:
			  pass
			else:
			  print('No ganó a')

Gana a	Gana b	a no es el más grande	No ganó a

elif "si no se da la condición if anterior, verifica si se da la siguiente". O si no ELSE

Las condiciones van a estar dadas por un valor booleano. Se pueden combinar varias condiciones usando los operadores booleanos (**and**, **or**, **not**):

break Comando para interrumpir y terminar el bucle, y continua con comandos que siguen.

numeros = [1, 2, 3, 4, 5]
for i in numeros:
  print(i)
  if i == 3:
  break
print('bucle finalizado','ultimo valor de i',i)
	números = [1, -5, 2, -8, 10]
	# imprimo cuadrados de positivos
	for i in números:
	  if i <= 0:
	  continue
	  print(i**2)
		nom = ['Juan', 'Elsa', 'Ingrid', 'Carlos', 'Federico']
		for i, nombre in **enumerate**(nom):
		  print(i, nombre)

1, 2, 3	1, 4, 100	0 Juan, 1 Elsa, 2 Ingrid, etc

Forma general de **enumerate**: enumerate(secuencia [, inicio = 0]), donde valor inicio de contador es opcional.

Secuencias: colección ordenada de elementos. Poseen indexación y tienen longitud. Hay 3 tipos de secuencias: **cadenas**, **listas** y **tuplas**. Pueden concatenarse (unirse) y replicarse. Pueden mostrar max, min, sum

Slicing: (rebanamiento) = s[principio:fin] EL ultimo queda EXCLUIDO

None Null Nil es ausencia de valor

Contenedores: Estructura de datos q' contienen objetos adentro, almacenan.

Cadenas ( ) **Inmutables** No pueden modificarse	Nros enteros – Valor en MEM Y **variable** referencia o puntero a ese valor. Coordenadas en la MEM.	Al hacer a = 11 no cambio el valor en mem sigue el 10 en el mismo lugar y creo nuevo valor en otro lado.	a = 10 b = a a = 11 print('a =', a) print('b =', b) # a=11 b=10
Lista [ ]	**Mutables** que pueden modificarse s/definir Nueva List	Elementos Repetidos e Indexados	separador = ';' resultado = separador.**join**(lista)
Lista	line = 'Manzana,200,390.10' fila = line.split(',') #, indica separador print(len(fila))		a = [1, 2] b = a a.append(3) R = a[1, 2, 3] b=[1, 2, 3]
Conjuntos { } S/Clave Set Add Remove & Intersection I Union Conj - Diferencia Conj-	**Mutables** No repetidos, se eliminan autom	Multiples elementos desorden.	lista_frutas = ['banana', 'naranja', 'naranja'] set_frutas = set(lista_frutas) print(set_frutas) len(lista_frutas) R={'banana', 'naranja'} 3
Tuplas ( )	**Inmutables** Se debe desempaquetar	Index.y Repetidos	lista = ['a', 20, 17.5, 'b'] for e in enumerate(lista): print('tupla:', e)
Tuplas P/registros. Como Fila de BD	1 Obj c/muchas partes. Info de 1 Entidad Tupla dentro de Lista frutas = [('Manzanas', 100, 490.1), ('Peras', 120, 582.5)]		tupla: (0, 'a') tupla: (1, 20)

Diccionarios { } for item in dic_ivan.items(): print(item) Res=('Nom', 'Ivan', 'edad', 28)	Key(clave): value(valor) clave[ ] - más rápido for **val** in dic_ivan.values(): print(val) R = Ivan 28	Almacenan datos en pares NO claves duplicadas for **clave** in dic_ivan.keys(): print(dic_ivan[clave]) R= Ivan 28	dic_ivan = {} dic_ivan['Nom'] = 'Iván' dic_ivan['edad'] = 28 print(dic_ivan) Res={'Nom': 'Ivan', 'edad': 28}

Función **zip**() toma secuencias (listas, tuplas, etc.) y las empaqueta en un iterador que las combina:

```
1 nombres = ['Elsa    ', 'Ingrid', 'Federico']
2 apellidos = ['Pérez', 'Gómez', 'Muller']
3 pares = zip(nombres, apellidos)
4 for nombre, apellido in pares:
5   print('Nombre:\t', nombre,'\tApellido:\t', apellido)
```

```
Nombre:   Elsa          Apellido:       Pérez
Nombre:   Ingrid        Apellido:       Gómez
Nombre:   Federico      Apellido:       Muller
```

 # Creo lista de tuplas

```
lista_tup = list(zip(nombres, apellidos))
print(lista_tup)
```

 # Creo un diccionario

```
dic = dict(zip(nombres, apellidos))
print(dic)
```

ZIP a la Inversa -

```
alumnos = [
   ('Carlos Gómez', 41008418, 5),
   ('Gabriela Torres', 45790918, 8)]
print(list(zip(*alumnos)))
```

Resultado = [('Carlos Gómez', 'Gabriela Torres', (41008418, 45790918), (5, 8)]

Comprensión de listas: La comprensión Lista crea una nueva Lista aplicando una operación

```
lista1 = [x**2 for x in range(0,5)]          #  Hasta la posición 5 Potencia al cuadrado
print(lista1)                                #  0, 1, 2, 3, 4
Resultado = [0, 1, 4, 9, 16]                 # x = 0, 1, 4, 9, 16
```

Creación de Listas

```
a = [1, 2, 3, 4, 5]
b = [2*x for x in a]   # Creo lista c/doble de los elementos de a = [2, 4, 6, 8, 10]
```

Otra forma:

```
a = [1, 2, 3, 4, 5]
b = [ ]
for x in a:
  b.append(2*x)
print(b)                    Resultado = [2, 4, 6, 8, 10]
```

Pase a minuscula

```
nombres = ['Edmundo', 'Juana']
a = [nombre.lower() for nombre in nombres]
print(a)          # ['edmundo', 'juana']
```

Filtros de datos en comprension de Listas

```
a = [1, -5, 4, 2, -2, 10]
b = [2*x for x in a if x > 0]          # uso sólo  positivos
print(b)                               # [2, 8, 4, 20]  Multiplica * 2 la posición
```

Comprensión de diccionarios:

```
numeros = [i for i in range(4)]   # Defino dicc.con cuadrados de números  = [0. 1, 2, 3]
dic2 = {n:n**2 for n in numeros}
print(dic2)                       #  Resultado =  {0: 0, 1: 1, 2: 4, 3: 9}
```

Función id

```
a = 10
print('a =', a, id(a))       #  a = 10 Y el id de a =    139952382233168    Refiere al valor de a
b = a
print('b =', b, id(b))       #  b = 10 Y el id de b =    139952382233168    Refiere al valor de a
a = 11
print('a =', a,  id(a))      #  a = 11 Y el id de b =    139952382233200    Refiere a nuevo valor de a
print('b =', b, id(b))       #  b = 10 Y el id de b =    139952382233168    Refiere al valor de a inicial
```

Ejemplos 001 – Copias profundas

```
lista_a = [1, "a"]
lista_b = [5.5, lista_a]
lista_c = lista_b.copy()
print(lista_c)
lista_a.append('esta cadena la agrego a lista_a')
lista_b.append('esta cadena la agrego a lista_b')
print(lista_c)          #   R  =  [5.5, [1, 'a']]      Y          [5.5, [1, 'a', 'esta cadena la agrego a lista_a']]
```

Otros Ejemplos

```
3
4 a = "hola mundo"
5 print(a[:7])
6
```

```
hola mu
```

```
1 a = 9.87645/5
2 a = round(a, 3)
3 print(a)
```

```
1.975
```

```python
x = 1
a = int(x)  # convertir x a int
b = float(x)    # Convertur x a float
print("%.5f"%b)
```

```
1.00000
```

```python
x = 194.8717
a = int(x)
b = float(x)
print("%.7f"%b)
```

```
194.8717000
```

```python
# intereses    001   - Crisco
cap = 100       # dolares
renta = 10      # %interes anual
a = 7           # años
saldo = (cap *(1 + renta /100) **a)
saldo = round(saldo,4)
print("Dolares Acumulados a los 7 años:     ", saldo )
```

```
Dolares Acumulados a los 7 años:      194.8717
```

```python
# Hipoteca de David

cap = 500000.0
tasa = 0.05     # 5 %  Anual
pago_mes = 2684.11     # Importe a pagar mensual
pagara = 0
mes = 0

while cap > 0:
    cap = cap * (1+tasa/12) - pago_mes
    pagara = pagara + pago_mes
    mes += 1

print("Total a Pagar:                {:,.2f}".format(pagara))
print("\n")
print("Cantidad de Meses a Pagar:   ", mes)

```

```
Total a Pagar:                966,279.60


Cantidad de Meses a Pagar:    360
```

```python
# Ej. 03 Masa corporal - Crisco
nombre = input("Ingrese su Nombre:\t")
edad_str = input('¿Edad?:\t''\t''\t')
edad = int(edad_str)
peso = float(input("Ingrese su Peso[Kg.]:\t"))
altura = float(input("Ingrese su Altura:\t"))
imc = round(peso / (altura **2), 2)
print(f"\nNombre:\t {nombre}\nEdad:\t {edad} \nAltura:\t {altura} \nPeso:\t {peso} \nIndice Masa Corporal:\t {imc}")

```

```
Ingrese su Nombre:      lll
¿Edad?:                 45
Ingrese su Peso[Kg.]:   80
Ingrese su Altura:      1.6

Nombre:  lll
Edad:    45
Altura:  1.6
Peso:    80.0
Indice Masa Corporal:    31.25
```

```python
# SOLUCION 1   - Crisco
num = 5
a = num **4
print(a)
type(a)

```

```
625
int
```

```python
# EJERCICIO 5 NUEVA FRASE - Crisco
frase = input("\nIngrese una frase:\t" )
cant_palabras = len(frase.split())
print("\nCantidad de Palabras:\t", cant_palabras)
cant_caracteres = len(frase)
print("\nCantidad de Caracteres:\t", cant_caracteres)
```

```
Ingrese una frase:      De acuerdo a lo visto y conversado, todo deberia estar OK

Cantidad de Palabras:    11

Cantidad de Caracteres:  57
```

Geringoso

```python
# jeringoso - Ej. 2 - Crisco
cadena = 'Geringoso'
capadepenapa = ""
vocal = ["a","e", "i", "o", "u"]
for c in cadena:
  capadepenapa = capadepenapa + c
  if(c in vocal):
    capadepenapa = capadepenapa + "p" + c
print(capadepenapa)
```

```
Geperipingoposopo
```

MESES Y DIAS

```python
# EJERCICIO  3  SEMANA 3 - Meses/Dias Crisco
dias = [31,28,31,30,31,30,31,31,30,31,30,31]
mes = ["Enero","Febrero","Marzo","Abril","Mayo","Junio","Julio","Agosto","Septiembre","Octubre","Noviembre","Diciembre"]
num_mes = input("Ingrese el Numero de Mes: \t")
indice = int(num_mes)-1
m = mes[indice]
d = dias[indice]
print("\n")
f"Nombre del Mes:    {m}    Cantidad de Dias:    {d}"
```

```
Ingrese el Numero de Mes:       7


'Nombre del Mes:    Julio    Cantidad de Dias:    31'
```

Otro MES Y DIA

```python
# MESES Y DIAS -  EJ. 3 - OTRO
meses = ["Enero","Febrero","Marzo","Abril","Mayo","Junio","Julio","Agosto","Septiembre","Octubre","Noviembre","Diciembre"]
mes_con_28_dias = [2]   #  Mes con 28 dias esta en posicion 2 (El indice empieza en 0 pero el valor en 1)
mes_con_31_dias = [1,3,5,7,8,10,12]
mes_con_30_dias = [4,6,9,11]
meses_num = mes_con_28_dias + mes_con_30_dias + mes_con_31_dias
mes = (int(input("Ingrese mes (su numero): \t")))
mes_menos_1 = mes - 1
if mes in meses_num:
    for m in enumerate(meses):
        indice = m[0]
        valor = m[1]
        if mes_menos_1 == indice and mes in mes_con_28_dias:
            print(f"\nMes: {valor}, contiene 28 dias")
            break
        elif mes_menos_1 == indice and mes in mes_con_31_dias:
            print(f"\nMes: {valor}, contiene 31 dias")
            break
        elif mes_menos_1 == indice and mes in mes_con_30_dias:
            print(f"\nMes: {valor}, contiene 30 dias")
            break
        else:
            continue
else:
    print("Ingrese un numero correcto (1 al 12)")
```

```
Ingrese mes (su numero):        3

Mes: Marzo, contiene 31 dias
```

ENCONTRAR SUBSECUENCIAS DE LISTA QUE SUMEN UN OBJECTIVO DADO

```python
# EJERCICIO 7 OPCIONAL - CRISCO
lista = [1,2,3,4,5,6,7,8,9,10]
print(lista)
objectivo = int(input("\nINGRESE UN NUMERO MENOR A 10 \t"))
conjuntos = [[]]
resultados = []
for n in lista:
  for indice in range(len(conjuntos)):
    nueva = conjuntos[indice] + [n]
    conjuntos.append(nueva)
    if sum(nueva) == objectivo:
      resultados.append(nueva)
print(f"\nNuevas Listas: {resultados}")

```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

INGRESE UN NUMERO MENOR A 10     8

Nuevas Listas: [[1, 3, 4], [1, 2, 5], [3, 5], [2, 6], [1, 7], [8]]
```