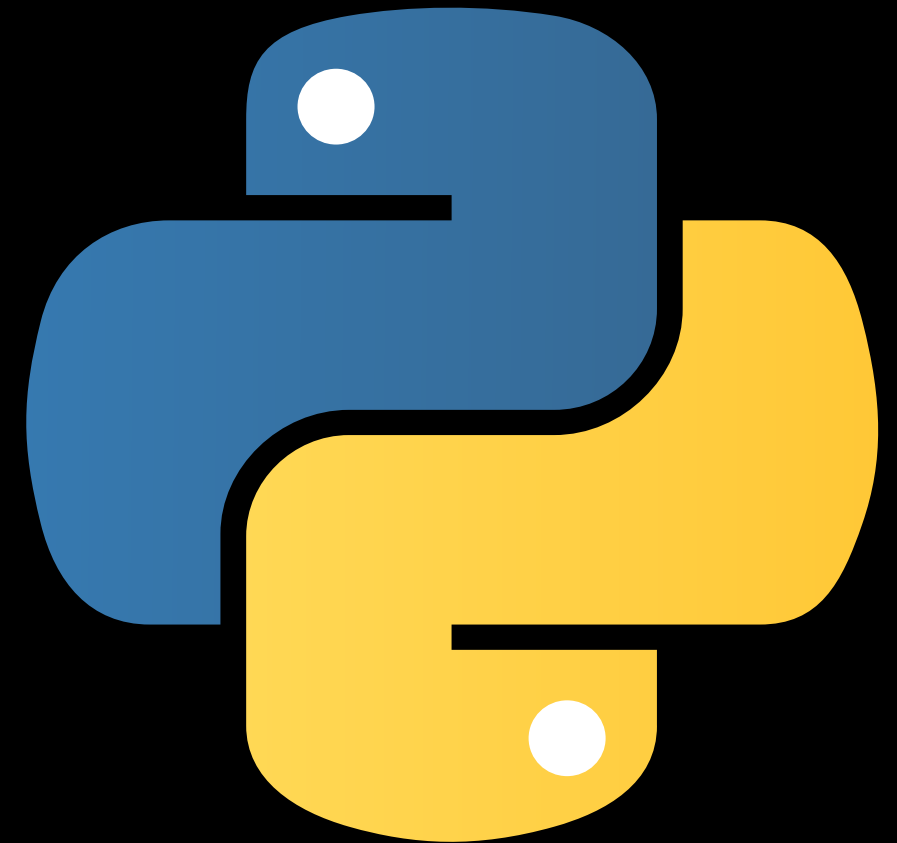


LABORATORIO III

Python



C.S.O COMPUTACIÓN

Prof: Lozano Juan Pablo



@EETVillaguay



@eetlvillaguay



/eetlvillaguay

CONTENIDOS

- INTRODUCCIÓN AL LENGUAJE
REQUISITOS. INSTALACIÓN SOFTWARE.
- FUNDAMENTOS Y SINTAXIS BASICA
- POO CON PYTHON
- ALGORITMOS, LISTAS Y TRAMOS
- BBDD
- TRABAJO CON GRÁFICOS Y CONTENIDOS
- PROCESOS Y TAREAS
- PROGRAMACIÓN DE RED. SOCKETS

CARACTERÍSTICAS

- CREADO POR GUIDO VAN ROSSUM EN LOS '90.
- LENGUAJE DE ALTO NIVEL. GRAMÁTICA SENCILLA, CLARA Y MUY LEGIBLE.
- TIPADO DINÁMICO Y FUERTE.
- ORIENTADO A OBJETOS.
- SOBRECARGA DE INSTRUCCIONES. HERENCIA MÚLTIPLE. ENCAPSULAMIENTO. INTERFACES. POLIFORMISMO.**
- OPEN SOURCE.
- FÁCIL DE APRENDER.
- LIBRERÍA ESTÁNDAR MUY AMPLIA.
- INTERPRETADO.
- VERSÁTIL.
- APLICACIONES WEB, DE ESCRITORIO, DE SERVIDOR**

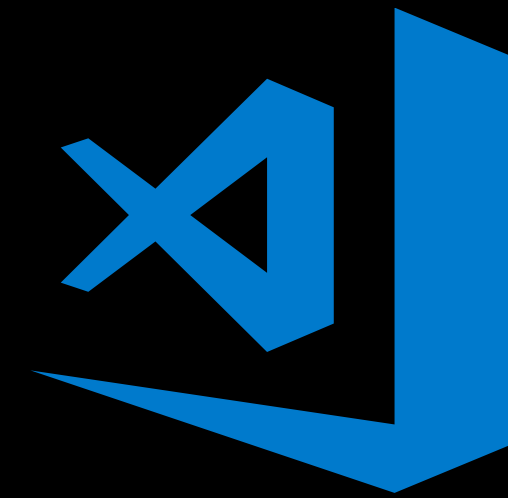
INSTALACIÓN DEL SOFTWARE

- INGRESAMOS A LA WEB PYTHON.ORG. DESCARGAMOS E INSTALAMOS.
AL INSTALAR, MARCAMOS LA CASILLA ADD PYTHON TO PATH

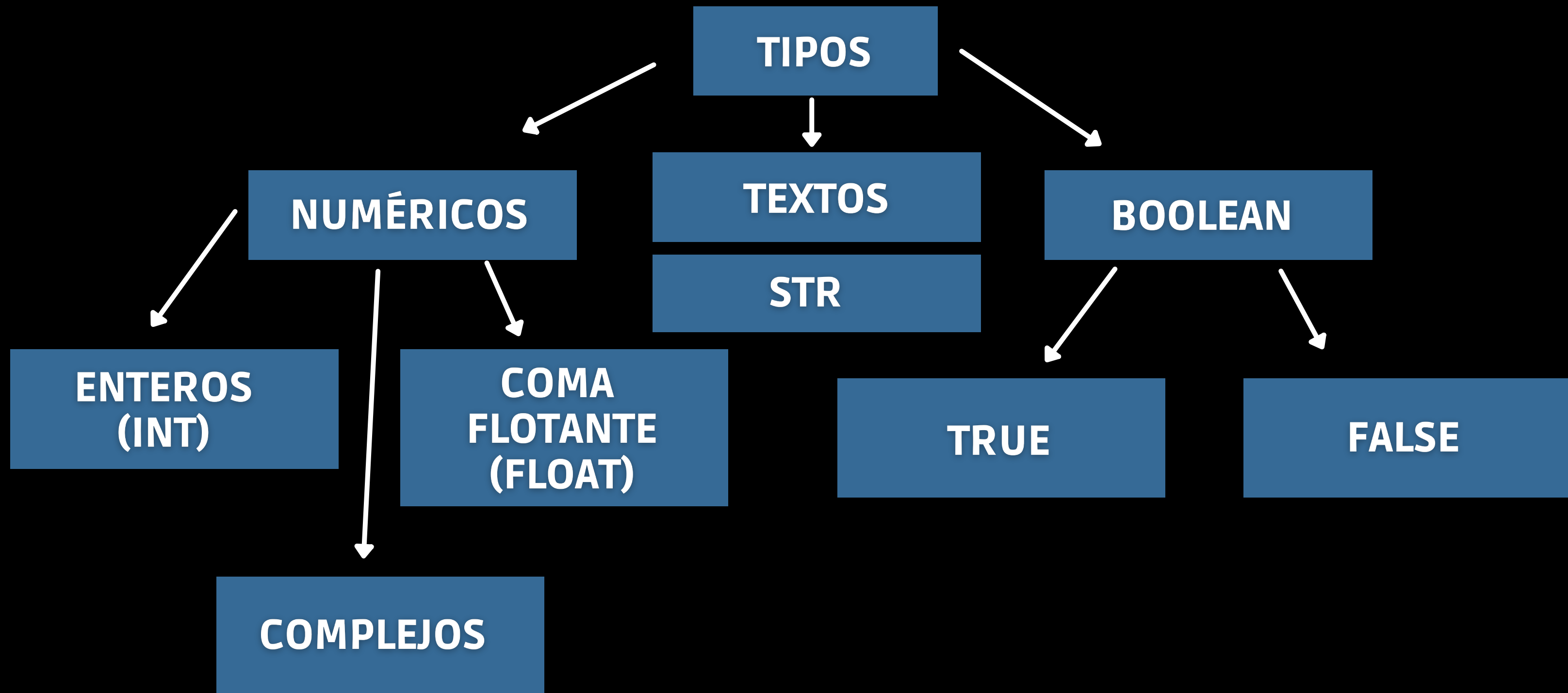


EDITORES DE CÓDIGO

- PODEMOS OPTAR POR CUALQUIER EDITOR DE CÓDIGO QUE DESEEN, SI USAN SUBLIME DEBERÁN DESCARGAR ALGUNAS EXTENSIONES PARA UTILIZARLO CON PYTHON.
- DE LA MISMA FORMA, SI UTILIZAN VSCODE, TIENEN QUE DESCARGAR EL PAQUETE DE PYTHON PARA ESTE SOFTWARE



Visual Studio Code





- ARITMÉTICOS: (+), (-), (*), (/), (MÓDULO %), (EXPONENTE **), (DIVISION ENTERA //)
- COMPARACIÓN: (IGUAL QUE ==), (DISTINTO !=), (>), (<), (>=), (<=)
- LÓGICOS: (AND), (OR), (NOT)
- ASIGNACIÓN: (IGUAL =), (INCREMENTO +=), (DECREMENTO -=), (*=), (/=), (%=), (**=), (//=)
- ESPECIALES: (IS), (IS NOT), (IN), (NOT IN)



PALABRAS RESERVADAS EN PYTHON

FALSE CLASS FINALLY IS RETURN

NONE CONTINUE FOR LAMBDA TRY

TRUE DEF FROM NONLOCAL WHILE

AND DEL GLOBAL NOT WITH

AS ELIF IF OR YIELD

ASSERT ELSE IMPORT PASS

BREAK EXCEPT IN RAISE

- VARIABLE: ESPACIO DE MEMORIA DONDE SE GUARDARÁ UN VALOR QUE PUEDE CAMBIAR DURANTE LA EJECUCIÓN DE UN PROGRAMA

LAS VARIABLES EN PYTHON SE CREAN CUANDO SE DEFINEN POR PRIMERA VEZ, ES DECIR, CUANDO SE LES ASIGNA UN VALOR POR PRIMERA VEZ. PARA ASIGNAR UN VALOR A UNA VARIABLE SE UTILIZA EL OPERADOR DE IGUALDAD (=).

- TYPE: DEVUELVE EL TIPO DE DATO DE LA VARIABLE
SINTAXIS: TYPE (NOMBRE VARIABLE)

- PRINT: IMPRIME UN MENSAJE EN PANTALLA
SINTAXIS: PRINT (" ")

- EJEMPLO: PRINT ("SOMOS 7MO COMPUTACIÓN")



LA ESTRUCTURA DE CONTROL IF ... PERMITE QUE UN PROGRAMA EJECUTE UNAS INSTRUCCIONES CUANDO SE CUMPLAN UNA CONDICIÓN. EN INGLÉS "IF" SIGNIFICA "SI" (CONDICIÓN).

SINTAXIS: IF CONDICIÓN:

LA EJECUCIÓN DE ESTA SENTENCIA ES LA SIGUIENTE:

LA CONDICIÓN SE EVALÚA SIEMPRE. SI EL RESULTADO ES TRUE SE EJECUTA EL BLOQUE DE SENTENCIAS. SI EL RESULTADO ES FALSE NO SE EJECUTA EL BLOQUE DE SENTENCIAS.

LA PRIMERA LÍNEA CONTIENE LA CONDICIÓN A EVALUAR Y ES UNA EXPRESIÓN LÓGICA. ESTA LÍNEA DEBE TERMINAR SIEMPRE POR DOS PUNTOS (:).



A CONTINUACIÓN VIENE EL BLOQUE DE ÓRDENES QUE SE EJECUTAN CUANDO LA CONDICIÓN SE CUMPLE (ES DECIR, CUANDO LA CONDICIÓN ES VERDADERA).

ES IMPORTANTE SEÑALAR QUE ESTE BLOQUE DEBE IR SANGRADO, PUESTO QUE PYTHON UTILIZA EL SANGRADO PARA RECONOCER LAS LÍNEAS QUE FORMAN UN BLOQUE DE INSTRUCCIONES. EL SANGRADO QUE SE SUELE UTILIZAR EN PYTHON ES DE CUATRO ESPACIOS, PERO SE PUEDEN UTILIZAR MÁS O MENOS ESPACIOS.

AL ESCRIBIR DOS PUNTOS (:) AL FINAL DE UNA LÍNEA, EL EDITOR SANGRARÁ AUTOMÁTICAMENTE LAS LÍNEAS SIGUIENTES. PARA TERMINAR UN BLOQUE, BASTA CON VOLVER AL PRINCIPIO DE LA LÍNEA.

ESTE SANGRADO ES LO QUE LLAMAMOS IDENTACIÓN

INSTRUCCIÓN ELIF



SE UTILIZA PARA EVALUAR MUCHAS CONDICIONES Y TODAS DEBEN IR ACOMPAÑADAS DE UN ELSE.

EJ:

```
PRINT ("VERIFICACIÓN DE ACCESO")
```

```
EDAD_USUARIO=INT(INPUT("INTRODUCIR EDAD:"))
```

```
IF EDAD_USUARIO <18:  
    PRINT ("NO PUEDE PASAR")  
IF EDAD_USUARIO >100:  
    PRINT ("EDAD INCORRECTA")  
ELSE:  
    PRINT ("PUEDE PASAR")
```

```
PRINT ("PROGRAMA TERMINADO")
```

```
IF EDAD_USUARIO <18:  
    PRINT ("NO PUEDE PASAR")  
ELIF EDAD_USUARIO >100:  
    PRINT ("EDAD INCORRECTA")  
ELSE:  
    PRINT("PUEDE PASAR")
```



INSTRUCCIÓN ELIF

ELIF ENLAZA CONDICIONES. SIRVE PARA ENLAZAR VARIOS “ELSE IF”, SIN TENER QUE AUMENTAR LAS IDENTACIONES EN CADA NUEVA COMPARACIÓN.

EN PYTHON NO EXISTE UNA ORDEN SWITCH O CASE COMO EN OTROS LENGUAJES, SINO QUE SE DEBEN REALIZAR ENLAZANDO VARIOS CASOS CON ELIF.

LA ESTRUCTURA ES SIEMPRE IF, LUEGO ELIF Y FINALMENTE ELSE (QUE SE EJECUTA EN CASO QUE NO SE HAYA CUMPLIDO NINGUNA DE LAS ANTERIORES IF/ELIF)

- TP 01: CREAR UNA VARIABLE NUMÉRICA, Y SI ESTÁ ENTRE 0 Y 10, MOSTRAR UN MENSAJE INDICÁNDOLO.
- TP 02: AÑADIR AL EJERCICIO ANTERIOR, QUE, SI ESTÁ ENTRE 11 Y 20, MUESTRE OTRO MENSAJE DIFERENTE Y SI ESTÁ ENTRE 21 Y 30, OTRO MENSAJE.
- TP 03: REPETIR EL EJERCICIO ANTERIOR, PERO AÑADIR QUE CUANDO LA VARIABLE SEA 0, MUESTRE EL MENSAJE CORRESPONDIENTE.

UN BUCLE WHILE PERMITE REPETIR LA EJECUCIÓN DE UN GRUPO DE INSTRUCCIONES MIENTRAS SE CUMPLA UNA CONDICIÓN (ES DECIR, MIENTRAS LA CONDICIÓN TENGA EL VALOR TRUE).

LA SINTAXIS DEL BUCLE WHILE ES LA SIGUIENTE:

**WHILE CONDICION:
CUERPO DEL BUCLE**

**LA EJECUCIÓN DE ESTA ESTRUCTURA DE CONTROL WHILE ES LA SIGUIENTE:
SE EVALÚA LA CONDICIÓN: SI EL RESULTADO ES TRUE SE EJECUTA EL CUERPO DEL BUCLE. UNA VEZ EJECUTADO EL CUERPO DEL BUCLE, SE REPITE EL PROCESO (SE EVALÚA DE NUEVO LA CONDICIÓN Y, SI ES CIERTA, SE EJECUTA DE NUEVO EL CUERPO DEL BUCLE) UNA Y OTRA VEZ MIENTRAS LA CONDICIÓN SEA CIERTA.
SI EL RESULTADO ES FALSE, EL CUERPO DEL BUCLE NO SE EJECUTA Y CONTINÚA LA EJECUCIÓN DEL RESTO DEL PROGRAMA.**

BUCLE WHILE

LA VARIABLE O LAS VARIABLES QUE APAREZCAN EN LA CONDICIÓN SE SUELEN LLAMAR VARIABLES DE CONTROL. LAS VARIABLES DE CONTROL DEBEN DEFINIRSE ANTES DEL BUCLE WHILE Y MODIFICARSE EN EL BUCLE WHILE.

EJEMPLO:

EL SIGUIENTE PROGRAMA ESCRIBE LOS NÚMEROS DEL 1 AL 3:

```
I = 1
WHILE I <= 3:
    PRINT(I)
    I += 1
PRINT("PROGRAMA TERMINADO")
```

```
1
2
3
PROGRAMA TERMINADO
```

ENTRADA POR TECLADO - INPUT ()

EN INFORMÁTICA, LA "ENTRADA" DE UN PROGRAMA SON LOS DATOS QUE LLEGAN AL PROGRAMA DESDE EL EXTERIOR. ACTUALMENTE, EL ORIGEN MÁS HABITUAL ES EL TECLADO.

LA FUNCIÓN INPUT() PERMITE OBTENER TEXTO ESCRITO POR TECLADO. AL LLEGAR A LA FUNCIÓN, EL PROGRAMA SE DETIENE ESPERANDO QUE SE ESCRIBA ALGO Y SE PULSE LA TECLA ENTER, COMO MUESTRA EL SIGUIENTE EJEMPLO:

EJEMPLO:

```
PRINT("¿CÓMO TE LLAMAS?")  
NOMBRE = INPUT()  
PRINT(F"HOLA, {NOMBRE}")
```

EN EL EJEMPLO, EL USUARIO ESCRIBE SU RESPUESTA EN UNA LÍNEA DISTINTA A LA PREGUNTA PORQUE PYTHON AÑADE UN SALTO DE LÍNEA AL FINAL DE CADA PRINT()

CONVERSIÓN DE TIPOS CON INPUT

DE FORMA PREDETERMINADA LA FUNCIÓN CONVIERTE LA ENTRADA EN CADENA, AUNQUE ESCRIBAMOS UN NÚMERO. PARA TRABAJAR CON ENTEROS O DECIMALES DEBEMOS AGREGAR EL TIPO DE DATO ANTES DEL INPUT:

```
CANTIDAD = INT (INPUT ("INGRESE UN NUMERO:"))  
CANTIDAD = FLOAT (INPUT ("INGRESE CANTIDAD:"))
```



PERMITEN CONCATENAR CADENAS DE TEXTO CON VARIABLES, DE FÁCIL LECTURA, INTEGRANDO VARIABLES Y EXPRESIONES DENTRO DE UNA CADENA CON SINTAXIS REDUCIDA.

EJEMPLO:

MARCA= NISSAN

PRECIO= 2.000.000

IMPUESTOS=PRECIO*21/100

PRINT(F"AUTO{MARCA}:{PRECIO+IMPUESTOS}PESOS")

UNA CADENA "F" SE SUELE PRESENTAR COMO UN LITERAL ENTRECOMILLADO AL QUE LE PRECEDE EL CARÁCTER "F" O "F": F'CADENA'

EL RESULTADO DE EVALUAR UNA CADENA "F" SE PUEDE ASIGNAR A UNA VARIABLE:

CADENAF = F'CADENA'

DENTRO DE UNA CADENA "F" SE PUEDEN INSERTAR VARIABLES Y EXPRESIONES ESCRIBIÉNDOLAS ENTRE LLAVES {}:



- TP 04: **MOSTRAR UN CICLO O BUCLE CON LOS NÚMEROS QUE ESTÁN ENTRE 1 Y 100.**
- TP 05: **ESCRIBIR UN PROGRAMA QUE PIDA DOS NÚMEROS ENTEROS. EL PROGRAMA PEDIRÁ DE NUEVO EL SEGUNDO NÚMERO MIENTRAS NO SEA MAYOR QUE EL PRIMERO. FINALMENTE, EL PROGRAMA TERMINARÁ ESCRIBIENDO LOS DOS NÚMEROS.**
- TP 06: **ESCRIBIR UN PROGRAMA QUE PIDA NÚMEROS DECIMALES MIENTRAS EL USUARIO ESCRIBA NÚMEROS MAYORES QUE EL PRIMERO.**
- TP 07: **ESCRIBIR UN PROGRAMA QUE PIDA NÚMEROS ENTEROS MIENTRAS SEAN CADA VEZ MÁS GRANDES.**

- TP 08: **ESCRIBA UN PROGRAMA QUE PREGUNTE UNA Y OTRA VEZ SI DESEA CONTINUAR CON EL PROGRAMA, SIEMPRE QUE SE CONTESTE EXACTAMENTE SÍ (EN MINÚSCULAS Y CON TILDE).**
- TP 09: **ESCRIBA UN PROGRAMA QUE PREGUNTE UNA Y OTRA VEZ SI DESEA TERMINAR EL PROGRAMA, SALVO SI SE CONTESTA EXACTAMENTE SI (EN MAYÚSCULAS Y SIN TILDE).**
- TP 10: **ESCRIBA UN PROGRAMA QUE PREGUNTE UNA Y OTRA VEZ SI DESEA TERMINAR EL PROGRAMA, SIEMPRE QUE SE CONTESTE EXACTAMENTE N (EN MAYÚSCULAS).**
- TP 11: **ESCRIBA UN PROGRAMA QUE PREGUNTE UNA Y OTRA VEZ SI DESEA CONTINUAR CON EL PROGRAMA, SIEMPRE QUE SE CONTESTE S O S (EN MAYÚSCULAS O EN MINÚSCULAS).**
- TP 12: **ESCRIBA UN PROGRAMA QUE SOLICITE UNA CONTRASEÑA (EL TEXTO DE LA CONTRASEÑA NO ES IMPORTANTE) Y LA VUELVA A SOLICITAR HASTA QUE LAS DOS CONTRASEÑAS COINCIDAN.**

- TP 13: ESCRIBIR UN PROGRAMA EN PYTHON QUE PIDA PRIMERO DOS NÚMEROS ENTEROS (MÍNIMO Y MÁXIMO) Y QUE DESPUÉS PIDA NÚMEROS ENTEROS SITUADOS ENTRE ELLOS. EL PROGRAMA TERMINARÁ CUANDO SE ESCRIBA UN NÚMERO QUE NO ESTÉ COMPRENDIDO ENTRE LOS DOS VALORES INICIALES. EL PROGRAMA TERMINA ESCRIBIENDO LA CANTIDAD DE NÚMEROS ESCRITOS.
- TP 14: ESCRIBA UN PROGRAMA QUE PIDA DOS NÚMEROS ENTEROS Y QUE ESCRIBA SI EL MAYOR ES MÚLTIPLO DEL MENOR.
- TP 15: ESCRIBA UN PROGRAMA QUE PIDA UN VALOR AL USUARIO Y REALICE LA DESCOMPOSICIÓN EL MISMO EN FACTORES PRIMOS.
- TP 16: ESCRIBA UN PROGRAMA QUE PIDA LOS COEFICIENTES DE UNA ECUACIÓN DE PRIMER GRADO ($A X + B = 0$) Y ESCRIBA LA SOLUCIÓN (UNA ECUACIÓN DE PRIMER GRADO PUEDE NO TENER SOLUCIÓN, TENER UNA SOLUCIÓN ÚNICA O QUE TODOS LOS NÚMEROS SEAN SOLUCIÓN). SE RECUERDA QUE LA FÓRMULA DE LAS SOLUCIONES ES $X = - B / A$

FUNCIONES



CONJUNTO DE LÍNEAS DE CÓDIGO AGRUPADAS (BLOQUE DE CÓDIGO) QUE FUNCIONAN COMO UNA UNIDAD REALIZANDO UNA TAREA ESPECÍFICA.

- LAS FUNCIONES EN PYTHON PUEDEN DEVOLVER VALORES
- LAS FUNCIONES EN PYTHON PUEDEN TENER PARÁMETROS/ARGUMENTOS
- A LAS FUNCIONES TAMBIÉN SE LAS DENOMINA "MÉTODOS", CUANDO SE ENCUENTRAN DEFINIDAS DENTRO DE UNA CLASE

UTILIDAD

REUTILIZACIÓN DE CÓDIGO (CUANDO SEA NECESARIO, SI ES NECESARIO)

SINTAXIS

DEF NOMBRE_FUNCION () :

- INSTRUCCIONES DE LA FUNCIÓN
- RETURN (OPCIONAL)

DEF NOMBRE_FUNCION (PARÁMETROS) :

- INSTRUCCIONES DE LA FUNCIÓN
- RETURN (OPCIONAL)

EJECUCIÓN

NOMBRE_FUNCION ()

NOMBRE_FUNCION (PARÁMETROS)



FUNCIONES

EJEMPLO: IMPRIMO 3 TEXTOS EN PANTALLA

```
PRINT ("HOLA, SOMOS SEPTIMO COMPUTACION")
```

```
PRINT ("ESTAMOS EN LABORATORIO III")
```

```
PRINT ("TRABAJANDO CON LENGUAJE PYTHON")
```

AHORA, CREAMOS UNA **FUNCIÓN** LLAMADA **MENSAJE** Y LOS COPIAMOS DENTRO DEL CUERPO DE LA MISMA CON LA RESPECTIVA IDENTACIÓN:

```
DEF MENSAJE():  
    PRINT ("HOLA, SOMOS SEPTIMO COMPUTACIÓN")  
    PRINT ("ESTAMOS EN LABORATORIO III")  
    PRINT ("TRABAJANDO CON LENGUAJE PYTHON")
```

```
MENSAJE ()
```

FUNCIONES **(PASO DE PARÁMETROS)**

```
DEF SUMA ()  
    NUM1 = 5  
    NUM2 = 7  
    PRINT (NUM1+NUM2)
```

SUMA ()

SUMA ()

SUMA ()

EN ESTE EJEMPLO, CREAMOS UNA FUNCIÓN LLAMADA SUMA CON DOS VARIABLES YA DEFINIDAS. LUEGO, EJECUTAMOS LA FUNCIÓN 3 VECES, LLAMANDOLA DESDE SU NOMBRE. ACÁ, LOS PARÁMETROS ESTÁN PREDEFINIDOS.

```
DEF SUMA (NUM1,NUM2)  
    PRINT (NUM1+NUM2)
```

SUMA (5,7)

SUMA (2,3)

SUMA (35,358)

EN ESTE OTRO EJEMPLO, CREAMOS UNA FUNCIÓN LLAMADA SUMA PERO LAS VARIABLES ESTÁN DENTRO DEL SECTOR DE PARÁMETROS DE LA FUNCIÓN. NOSOTROS SOMOS QUIENES LE PASAMOS LOS PARÁMETROS UNA VEZ EJECUTADA LA FUNCIÓN.

RETURN

CUANDO SE EVALÚA UNA EXPRESIÓN CON UNA LLAMADA DE FUNCIÓN, LA LLAMADA DE FUNCIÓN SE RESUELVE DE MANERA TEMPORAL CON UN RETURN (VALOR DEVUELTO)

EJEMPLO

```
DEF SUMA (NUM1,NUM2) :  
    RESULTADO = NUM1+NUM2  
    RETURN RESULTADO
```

```
PRINT (SUMA (5,7) )
```

```
PRINT (SUMA (2,3) )
```

```
PRINT (SUMA (35,358) )
```

```
DEF SUMA (NUM1,NUM2) :  
    RESULTADO = NUM1+NUM2  
    RETURN RESULTADO
```

```
ALMACENA_RESULTADO = SUMA (5,8)  
PRINT (ALMACENA_RESULTADO)
```

-TP 17: CREAR UNA FUNCIÓN QUE SE LLAME AREA_RECTANGULO (BASE, ALTURA) Y QUE NOS DEVUELVA EL ÁREA DEL RECTANGULO INGRESADO A PARTIR DE UNA BASE Y UNA ALTURA. EL ÁREA DEL RECTÁNGULO A CALCULAR TENDRÍA 25 DE BASE Y 15 DE ALTURA.

**-TP 18: PROGRAMAR UNA FUNCIÓN LLAMADA EVALUAR(A, B) QUE A PARTIR DE DOS VALORES PREDETERMINADOS CUMPLA LO SIGUIENTE:
SI EL PRIMER VALOR ES MÁS GRANDE QUE EL SEGUNDO, DEVOLVER 1.
SI EL PRIMER VALOR ES MÁS CHICO QUE EL SEGUNDO, DEVOLVER -1.
SI AMBOS VALORES SON IGUALES, DEBE DEVOLVER UN 0.
LA RELACIÓN A COMPROBAR ES ENTRE LOS NÚMEROS: '20 Y 30', '30 Y 20' Y '20 Y 20'.**

-TP 19:: CREAR UNA FUNCIÓN LLAMADA PUNTOMEDIO(A,B) QUE TENIENDO EN CUENTA DOS VALORES IDENTIFIQUE EL PUNTO MEDIO ENTRE LOS MISMOS. REALIZAR LA COMPROBACIÓN ENTRE LOS VALORES 2 Y 48.

**-TP 20: PROGRAMAR UNA FUNCIÓN LLAMADA INFORMAR(VALOR, BAJO, ALTO) QUE RECIBA TRES PARÁMETROS. EL PRIMER PARÁMETRO ES EL NÚMERO A INFORMAR, EL SEGUNDO ES EL LÍMITE MÁS BAJO Y EL TERCERO EL LÍMITE MÁS ALTO. LA FUNCIÓN DEBERÁ TRABAJAR DE LA SIGUIENTE MANERA: DEVOLVER EL LÍMITE MÁS BAJO SI VALOR ES MENOR A ÉSTE
DEVOLVER EL LÍMITE MÁS ALTO SI VALOR ES MÁS GRANDE QUE ÉSTE.
DEVOLVER EL NÚMERO TAL CUAL ES SI NO SE SUPERA NINGÚN LÍMITE.
REALIZAR EL RESULTADO DE INFORMAR 25 ENTRE LOS LÍMITES 1 Y 40.21**

-TP 21: PROGRAMAR UNA FUNCIÓN LLAMADA MÁXIMO, QUE AL INGRESAR DOS NÚMEROS (PEDIR INGRESO POR EL USUARIO) DEVUELVA EL MAYOR DE ELLOS.

-TP 22: PROGRAMAR UNA FUNCIÓN LLAMADA MÍNIMO, QUE AL INGRESAR DOS NÚMEROS (PEDIR INGRESO POR EL USUARIO) DEVUELVA EL MENOR DE ELLOS.

- TP 23: PROGRAMAR UNA FUNCIÓN QUE TOME UN CARÁCTER INGRESADO POR EL USUARIO Y DEVUELVA 'TRUE' SI ES UNA VOCAL, Y SINO 'FALSE'.
- TP 24: PROGRAMAR UNA FUNCIÓN LLAMADA CALCULO, QUE TOME DOS VALORES INGRESADOS POR EL USUARIO, SI ALGUNO ES 0 (CERO) INFORMAR: ERROR Y TERMINAR EL PROCESO; SI EL PRIMERO ES POSITIVO Y EL SEGUNDO NEGATIVO, SUMARLOS; SI EL PRIMERO ES NEGATIVO Y EL SEGUNDO POSITIVO, RESTARLOS; SI AMBOS SON POSITIVOS, DIVIDIRLOS; SI AMBOS SON NEGATIVOS, MULTIPLICARLOS. EN TODOS LOS CASOS REALIZAR LA OPERACIÓN CORRESPONDIENTE DEJANDO EL RESULTADO EN UNA VARIABLE Y MOSTRAR EL RESULTADO.

LISTAS

ESTRUCTURA DE DATOS QUE NOS PERMITE ALMACENAR UNA GRAN CANTIDAD DE VALORES (EQUIVALENTE A LOS ARRAYS EN OTROS LENGUAJES).

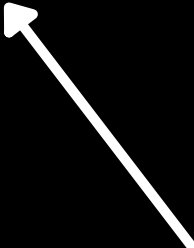
EN PYTHON LAS LISTAS PUEDEN GUARDAR DIFERENTES TIPOS DE VALORES.

SE PUEDEN EXPANDIR DINÁMICAMENTE AÑADIENDO NUEVOS ELEMENTOS.

SINTAXIS:

MILISTA = [ELEM1, ELEM2, ELEM3...]
 (0) (1) (2)

INDICE O POSICIÓN



VER ELEMENTOS DE UNA LISTA

MILISTA = ["MARÍA", "JUAN", "PABLO"]

PRINT (MILISTA [:])	IMPRIME TODOS LOS ELEMENTOS
PRINT (MILISTA [2])	IMPRIME EL ELEMENTO EN LA POS.2
PRINT (MILISTA [-3])	IMPRIME EL ELEMENTO EN LA POS.-3 (EMPIEZA DESDE EL FINAL, PERO EL ÚLTIMO NÚMERO ES -1)
PRINT (MILISTA [0:3])	IMPRIME 0, 1 Y 2 (ACCEDER A UNA PORCIÓN)
PRINT (MILISTA [2:])	DESDE ÍNDICE 2 HASTA EL FINAL.

AGREGAR ELEMENTOS

```
MILISTA = ["MARÍA", "JUAN", "PABLO"]
```

```
MILISTA.APPEND ("SANDRA")
```

AGREGA ELEMENTO AL FINAL

```
PRINT (MILISTA [ : ])
```

```
MILISTA.INSERT (2, "SANDRA")
```

AGREGA ELEMENTO DONDE LO
INDIQUE EL ÍNDICE

```
MILISTA.EXTEND (["SANDRA", "ANA", "LUCÍA"])
```

AGREGA MÚLTIPLES ELEMENTOS

CONSULTAR ELEMENTOS

PRINT (MILISTA.INDEX("ANTONIO"))

DEVUELVE EL ÍNDICE DEL ELEMENTO

PRINT ("PEPE" IN MILISTA)

**MUESTRA SI EXISTE UN ELEMENTO,
TRUE O FALSE**

MILISTA = ["MARÍA", 5 , TRUE , 78.35]

**LAS LISTAS EN PYTHON PUEDEN
GUARDAR DISTINTOS TIPOS DE DATOS.**

ELIMINAR ELEMENTOS

MILISTA.REMOVE ("ANA")

ELIMINA EL ÚLTIMO ELEMENTO SELECCIONADO

MILISTA.POP ()

ELIMINA EL ÚLTIMO ELEMENTO DE LA LISTA

UNIR LISTAS

```
MILISTA = ["MARÍA", 5, TRUE, 78.35]
```

```
MILISTA2 = ["SANDRA", "LUCÍA"]
```

```
MILISTA3 = MILISTA+MILISTA2
```

```
PRINT (MILISTA3[ : ])
```

REPETIR LISTAS

```
MILISTA = ["MARÍA", 5, TRUE, 78.35] * 3
```

```
PRINT (MILISTA [:])
```

EN ESTE CASO ,REPITE LA LISTA 3 VECES.

- TP 25

1-CREAR UNA LISTA CON LOS NOMBRES DE TODOS LOS INTEGRANTES DE 7° COMPUTACIÓN.

2-CREAR OTRA LISTA CON LOS PROFESORES DE 7°COMPUTACIÓN.

3-CREAR OTRA LISTA CON LAS ASIGNATURAS DE 7°COMPUTACIÓN.

4-CONCATENAR LA LISTA DE ALUMNOS Y PROFESORES.

5-GUARDAR ESTE ARCHIVO COMO EJERCICIO 25, SIGUIENDO EL ORDEN DE LOS PRÁCTICOS ANTERIORES.

- TP 26:

UTILIZAR COMO BASE EL EJERCICIO 25 PARA:

1- A LA LISTA DE ALUMNOS AGREGARLE 4 ALUMNOS MÁS AL FINAL DE LA MISMA

2- A LA LISTA DE PROFESORES AGREGARLE UNO MÁS EN LA POSICION 3.

3- A LA LISTA DE ASIGNATURAS AGREGARLE UNA MÁS AL FINAL DE LA MISMA.

4- REPETIR 3 VECES LA LISTA DE ALUMNOS.

5- GUARDAR ESTOS NUEVOS CAMBIOS COMO EJERCICIO 26.

TUPLAS



SON LISTAS INMUTABLES, ES DECIR, NO SE PUEDEN MODIFICAR DESPUES DE SU CREACIÓN.

NO PERMITEN AÑADIR, ELIMINAR, MOVER ELEMENTOS, ETC (NO APPEND, EXTEND).

SI PERMITEN EXTRAER PORCIONES, PERO EL RESULTADO DE LA ACCIÓN ES UNA TUPLA NUEVA.

NO PERMITEN BÚSQUEDAS (NO INDEX)

PERMITEN COMPROBAR SI UN ELEMENTO SE ENCUENTRA EN LA TUPLA.



TUPLAS



¿QUE UTILIDAD O VENTAJA TIENEN RESPECTO DE LAS LISTAS?

SON MÁS RÁPIDAS

MENOS ESPACIO (MAYOR OPTIMIZACIÓN)

FORMATEAN STRINGS

PUEDEN UTILIZARSE COMO CLAVES EN LOS DICCIONARIOS (LAS LISTAS NO)

LAS TUPLAS UTILIZARÁN PARÉNTESIS EN TODOS LOS CASOS, EXCEPTO CUANDO TRABAJAN CON DICCIONARIOS (UTILIZARAN CORCHETES) Y PARA EMPAQUETAR Y DESEMPAQUETAR TUPLAS (NO UTILIZAN NI PARENTESIS NI CORCHETES).



TUPLAS

SINTAXIS:

NOMBRETUPLA=(ELEM1, ELEM2, ELEM3...) SE PUEDEN OMITIR LOS ()

EJEMPLOS:

NOMBRETUPLA=("MARÍA",13,1,1995) IMPRIME ELEMENTO EN POSICIÓN 2

PRINT (NOMBRETUPLA [2])

TUPLAS

CONVERTIR TUPLA EN LISTA:

```
NOMBRELISTA=LIST(NOMBRETUPLA)
```

```
PRINT(NOMBRELISTA)
```

CONVERTIR LISTA EN TUPLA:

```
NOMBRELISTA=["MARÍA",13,1,1995]
```

```
NOMBRETUPLA=TUPLE(NOMBRELISTA)
```

```
PRINT(NOMBRETUPLA)
```

TUPLAS



COMPROBAR SI HAY ELEMENTOS EN UNA TUPLA:

```
PRINT ("JUAN" IN NOMBRETUPLA)
```

AL SER BOOLEAN ENTREGARA UN RESULTADO (TRUE O FALSE) SEGUN CORRESPONDA.

CONTAR ELEMENTOS EN UNA TUPLA:

```
PRINT(NOMBRETUPLA.COUNT(13))
```

INDICA LA CANTIDAD DE VECES QUE SE ENCUENTRA EL ELEMENTO 13 EN LA TUPLA



TUPLAS

AVERIGUAR LA LONGITUD DE UNA TUPLA

```
PRINT (LEN (NOMBRETUPLA))
```

CREAR TUPLAS UNITARIAS (UN UNICO ELEMENTO):

```
NOMBRETUPLA=("JUAN")
```

```
PRINT (LEN(NOMBRETUPLA))
```

TUPLAS

EMPAQUETADO DE TUPLA - TUPLA SIN ()

NOMBRETUPLA="JUAN",13,1,1995
 NOMBRE,DIA,MES,ANIO


ASIGNAMOS VARIABLES A CADA ELEMENTO

DESEMPAQUETADO DE TUPLA

NOMBRE,DIA,MES,ANIO=NOMBRETUPLA
PRINT(NOMBRE)
PRINT(DIA), ETC

- TP 27:

1-ESCRIBIR LAS LÍNEAS DE CÓDIGO PARA UN PROGRAMA DE PYTHON QUE PERMITA INGRESAR 10 NOMBRES DE PERSONAS EN UNA TUPLA E IMPRIMA EN PANTALLA LAS POSICIONES 4, 6 Y 9.

2-ESCRIBIR LAS LÍNEAS DE CÓDIGO PARA UN PROGRAMA DE PYTHON QUE PERMITA COMPROBAR SI EXISTE EL ELEMENTO 'MARTÍN' EN LA TUPLA DEL PUNTO 1.

3-ESCRIBIR LAS LÍNEAS DE CÓDIGO PARA UN PROGRAMA DE PYTHON QUE PERMITA CONTAR ALGUN ELEMENTO (A ELECCIÓN) DEL EJERCICIO 1.

4-ESCRIBIR LAS LÍNEAS DE CÓDIGO PARA UN PROGRAMA DE PYTHON QUE PERMITA AVERIGUAR LA LONGITUD DE LA TUPLA DEL PUNTO 1.

ESTRUCTURA DE DATOS QUE NOS PERMITE ALMACENAR VALORES DE DIFERENTE TIPO (ENTEROS, CADENAS DE TEXTO, DECIMALES) E INCLUSO LISTAS Y OTROS DICCIONARIOS.

LA PRINCIPAL CARACTERÍSTICA DE LOS DICCIONARIOS ES QUE LOS DATOS SE ALMACENAN ASOCIADOS A UNA CLAVE DE TAL FORMA QUE SE CREA UNA ASOCIACIÓN DE TIPO CLAVE:VALOR PARA CADA ELEMENTO ALMACENADO.

LOS ELEMENTOS ALMACENADOS NO ESTÁN ORDENADOS. EL ORDEN ES INDIFERENTE A LA HORA DE ALMACENAR INFORMACIÓN EN UN DICCIONARIO.

DICCIONARIOS

SINTAXIS:

NOMBREDICCIONARIO = {" ": " "}

EJEMPLO

NOMBREDICCIONARIO={"ALEMANIA":"BERLIN","FRANCIA":"PARÍS","ESPAÑA":"MADRID"}

PRINT(NOMBREDICCIONARIO["ALEMANIA"])

CLAVE : VALOR

DEVUELVE EL VALOR ASOCIADO A LA CLAVE

DICCIONARIOS

AGREGAR ELEMENTOS:

```
NOMBREDICCIONARIO ["ITALIA"] = "ROMA"  
PRINT (MIDICCIONARIO)
```

ELIMINAR ELEMENTOS:

```
DEL NOMBREDICCIONARIO ["ALEMANIA"]
```

GUARDAR DIFERENTES TIPOS DE DATOS:

```
NOMBREDICCIONARIO = {"ARGENTINA": "BUENOS AIRES", 10: "MESSI"}  
PRINT (NOMBREDICCIONARIO)
```

DICCIONARIOS

UTILIZAR TUPLA PARA ASIGNAR CLAVES AL DICCIONARIO

```
NOMBRETUPLA=["ESPAÑA","FRANCIA","REINO UNIDO","ALEMANIA"]
```

```
NOMBREDICCIONARIO={NOMBRETUPLA[0]:"MADRID",NOMBRETUPLA[1]:"PARIS",  
NOMBRETUPLA[2]: "LONDRES",NOMBRETUPLA[3]:"BERLIN"}
```

```
PRINT (MIDICCIONARIO)
```

```
PRINT (MIDICCIONARIO[FRANCIA])
```

SE CREA LA TUPLA, SE ASIGNA UN VALOR DE LA TUPLA A CADA VALOR DEL DICCIONARIO.

DICCIONARIOS



ALMACENAR UNA TUPLA DESDE UN DICCIONARIO

```
MIDICCIONARIO={23:"MICHAEL","NOMBRE":"JORDAN","EQUIPO":"CHICAGO","ANILLOS":[1991,  
1992,1995,1996,1997,1998]}
```

 ASIGNAMOS UNA TUPLA A UNA CLAVE (ANILLOS)

```
PRINT(MIDICCIONARIO)
```

```
PRINT(MIDICCIONARIO["ANILLOS"])
```

```
MIDICCIONARIO={23:"JUAN","NOMBRE":"MICHAEL","EQUIPO":"CHICAGO","ANILLOS":  
[1991,1992,1995,1996,1997,1998]}
```

```
PRINT(MIDICCIONARIO)
```



DICCIONARIOS



GUARDAR UN DICCIONARIO DENTRO DE OTRO

```
MIDICCIONARIO={23:"MICHAEL","NOMBRE":"JORDAN","EQUIPO":"CHICAGO","ANILLOS":  
{"TEMPORADAS":[1991,1992,1995,1996,1997,1998]}}
```

PRINT(MIDICCIONARIO)

SUBDICCIONARIO

TUPLA

PRINT(MIDICCIONARIO.KEYS())

DEVUELVE LAS CLAVES

PRINT(MIDICCIONARIO.VALUES())

DEVUELVE LOS VALORES

PRINT(LEN(MIDICCIONARIO))

DEVUELVE LA LONGITUD



- TP 28:

- 1- ESCRIBIR LAS LÍNEAS DE CÓDIGO PARA UN PROGRAMA DE PYTHON QUE PERMITA CREAR UN DICCIONARIO DONDE LA CLAVE SEA EL NOMBRE DEL USUARIO Y EL VALOR SEA EL TELEFONO (5 ELEMENTOS CON SU CLAVE: VALOR CORRESPONDIENTES). NO SE PUEDEN REPETIR NOMBRES.**
- 2- ESCRIBIR LAS LÍNEAS DE CÓDIGO PARA UN PROGRAMA DE PYTHON QUE PERMITA AGREGAR 5 ELEMENTOS MÁS A ESE DICCIONARIO.**
- 3- ESCRIBIR LAS LÍNEAS DE CÓDIGO PARA UN PROGRAMA DE PYTHON QUE PERMITA ELIMINAR LOS 3 ÚLTIMOS ELEMENTOS AGREGADOS AL DICCIONARIO EN EL EJERCICIO ANTERIOR.**
- 4- ESCRIBIR LAS LÍNEAS DE CÓDIGO PARA UN PROGRAMA DE PYTHON QUE PERMITA AGREGAR UN SUBDICCIONARIO AL DICCIONARIO CREADO ANTERIORMENTE.**
- 5- GUARDAR ESTE ARCHIVO COMO EJERCICIO 18, SIGUIENDO EL ORDEN DE LOS PRÁCTICOS ANTERIORES.**

- TP 29:

1- ESCRIBIR LAS LÍNEAS DE CÓDIGO PARA UN PROGRAMA DE PYTHON QUE PERMITA CREAR UNA LISTA DE NUMEROS DEL 1 AL 30.

2-CREAR UNA SEGUNDA LISTA SOLO CON LOS NUMEROS PARES DEL PUNTO 1.

3-CREAR UNA TERCER LISTA SOLO CON LOS NUMEROS IMPARES DEL PUNTO 1.

4-CREAR UNA CUARTA LISTA PARA GUARDAR LAS LISTAS DE LOS PUNTOS 2 Y 3 CONCATENADAS.

5- CONSULTAR SI SE ENCUENTRA EL NUMERO 40 EN CADA UNA DE LAS LISTAS.

6- MOSTRAR LA LONGITUD DE CADA UNA DE LAS LISTAS.

7- ELIMINAR LOS ULTIMOS ELEMENTOS DE CADA UNA DE LAS LISTAS.

8- ELIMINAR EL NUMERO 10 DE CADA UNA DE LAS LISTAS.

9-MOSTRAR EN PANTALLA TODAS LAS LISTAS RESULTANTES.

- TP 30:

REALIZAR UN EMPAQUETADO Y DESEMPAQUETADO DE TUPLA UTILIZANDO COMO VARIABLES UN EJEMPLO DISTINTO AL VISTO EN CLASES.

TP 31:

1- CONVERTIR A TUPLAS LAS LISTAS DEL EJERCICIO 29.

2-CREAR UNA TUPLA ADICIONAL CON LOS NUMEROS DEL 30 AL 50.

3-COMPROBAR SI LOS ELEMENTOS 10, 20, 30 Y 40 SE ENCUENTRAN EN LAS TUPLAS.

4-CONTAR CUANTAS VECES ESTÁ EL ELEMENTO 1 EN LAS TUPLAS.

-TP 32:

1- CREAR UN DICCIONARIO CON LAS CLAVES:VALORES CORRESPONDIENTES A LOS PRIMEROS 6 MESES DEL AÑO Y EL NÚMERO QUE LE CORRESPONDE A CADA MES.

2-AGREGAR AL DICCIONARIO, LOS SIGUIENTES 6 MESES DEL AÑO.

3-UTILIZAR UNA TUPLA PARA AGREGAR ALGUN DATO ADICIONAL AL DICCIONARIO (FECHAS DE CUMPLEAÑOS, ALGUNA EFEMERIDE, ETC).

-TP 33:

ESCRIBIR UN PROGRAMA QUE GUARDE EN UN DICCIONARIO LAS SIGUIENTES MONEDAS DE PAÍSES {'PESO ARGENTINO': 'S', 'DOLAR': '\$', 'EURO': 'E'}, Y QUE EL USUARIO PUEDA CONSULTAR POR UNA MONEDA Y EL PROGRAMA MUESTRE SU SÍMBOLO O UN MENSAJE DE AVISO SI LA MONEDA NO ESTÁ EN EL DICCIONARIO.

-TP 34:

ESCRIBIR UN PROGRAMA EN PYTHON QUE REALICE LA GESTIÓN DE LOS COMPROBANTES QUE NO HAN SIDO COBRADOS DE UNA EMPRESA.

ESTOS COMPROBANTES SE VAN A GUARDAR EN UN DICCIONARIO EN EL QUE LA CLAVE DE CADA COMPROBANTE SERÁ EL NRO DEL MISMO Y EL VALOR EL IMPORTE DE LA FACTURA.

EL PROGRAMA DEBERÁ INTERACTUAR CON EL USUARIO SOLICITÁNDOLE SI DESEA AGREGAR UN NUEVO COMPROBANTE, PAGAR UNO QUE YA EXISTE O FINALIZAR.

SI QUIERE AÑADIR UN NUEVO COMPROBANTE SE CONSULTARÁ POR EL NÚMERO DEL MISMO Y SU VALOR SE AGREGARÁ AL DICCIONARIO.

SI SE QUIERE ABONAR UN COMPROBANTE SE CONSULTARÁ EL NÚMERO DEL MISMO Y SE ELIMINARÁ DEL DICCIONARIO.

FINALIZADA CADA OPERACIÓN EL PROGRAMA DEBERÁ INFORMAR EN PANTALLA LA CANTIDAD COBRADA HASTA EL MOMENTO Y LA CANTIDAD PENDIENTE DE COBRO.

BUCLE FOR



UN BUCLE ES UNA ESTRUCTURA DE CONTROL QUE REPITE UN BLOQUE DE INSTRUCCIONES.
UN BUCLE **FOR** ES UN BUCLE QUE REPITE EL BLOQUE DE INSTRUCCIONES UN NÚMERO
PREDETERMINADO DE VECES. EL BLOQUE DE INSTRUCCIONES QUE SE REPITE SE SUELE LLAMAR
CUERPO DEL BUCLE Y CADA REPETICIÓN SE SUELE LLAMAR ITERACIÓN.
LA SINTAXIS DE UN BUCLE FOR ES LA SIGUIENTE:

FOR VARIABLE IN ELEMENTO ITERABLE (LISTA, CADENA, RANGE, ETC.) :
CUERPO DEL BUCLE



BUCLE FOR

NO ES NECESARIO DEFINIR LA VARIABLE DE CONTROL ANTES DEL BUCLE, AUNQUE SE PUEDE UTILIZAR COMO VARIABLE DE CONTROL UNA VARIABLE YA DEFINIDA EN EL PROGRAMA.

EL CUERPO DEL BUCLE SE EJECUTA TANTAS VECES COMO ELEMENTOS TENGA EL ELEMENTO RECORRIBLE (ELEMENTOS DE UNA LISTA O DE UN RANGE (), CARACTERES DE UNA CADENA, ETC.).

EJEMPLO:

```
PRINT("COMIENZO")  
FOR I IN [0, 1, 2]:  
    PRINT("HOLA ", END="")  
PRINT()  
PRINT("FINAL")
```

```
COMIENZO  
HOLA HOLA HOLA  
FINAL
```

BUCLE FOR

SE EJECUTA LA PRIMERA INSTRUCCIÓN DEL PROGRAMA. EN ESTE CASO, IMPRIME EL PÁRRAFO DE COMIENZO. A CONTINUACIÓN, SE EJECUTA EL BUCLE. LA VARIABLE DE CONTROL TOMA EL PRIMER VALOR DE LA LISTA. EN ESTE CASO, LA VARIABLE DE CONTROL ES I Y TOMA EL VALOR 0. LUEGO, SE EJECUTAN LAS INSTRUCCIONES DEL BLOQUE: EN ESTE CASO EL BLOQUE CONSTA DE UNA SOLA INSTRUCCIÓN QUE IMPRIME EL TEXTO "HOLA"; ENTONCES LA VARIABLE DE CONTROL TOMA EL SEGUNDO VALOR DE LA LISTA, EN ESTE CASO, LA VARIABLE DE CONTROL ES I Y TOMA EL VALOR 1. DESPUÉS DE ESTO, SE EJECUTAN POR SEGUNDA VEZ LAS INSTRUCCIONES DEL BLOQUE, EN ESTE CASO EL BLOQUE CONSTA DE UNA SOLA INSTRUCCIÓN QUE IMPRIME EL TEXTO "HOLA". LO MISMO PARA LA TERCERA VEZ. UNA VEZ TERMINADO EL BUCLE, SE EJECUTA LA INSTRUCCIÓN QUE SIGUE AL BUCLE. EN ESTE CASO, IMPRIME UN SALTO DE LÍNEA, PARA QUE EL SIGUIENTE PRINT() IMPRIMA EN LA LÍNEA SIGUIENTE.

BUCLE FOR



EN ESTE EJEMPLO SE HA UTILIZADO UNA LISTA PARA FACILITAR LA COMPRENSIÓN DEL FUNCIONAMIENTO DEL BUCLE PERO, SI ES POSIBLE HACERLO, SE RECOMIENDA UTILIZAR TIPOS RANGE (), ENTRE OTROS MOTIVOS PORQUE DURANTE LA EJECUCIÓN DEL PROGRAMA OCUPAN MENOS MEMORIA.

```
PRINT("COMIENZO")  
FOR I IN RANGE(3):  
    PRINT("HOLA ", END="")  
PRINT()  
PRINT("FINAL")
```

```
COMIENZO  
HOLA HOLA HOLA  
FINAL
```

OTRA DE LAS VENTAJAS DE UTILIZAR TIPOS RANGE() ES QUE EL ARGUMENTO DEL TIPO RANGE() CONTROLA EL NÚMERO DE VECES QUE SE EJECUTA EL BUCLE. EN EL EJEMPLO ANTERIOR BASTA CAMBIAR EL ARGUMENTO PARA QUE EL PROGRAMA SALUDE MUCHAS MÁS VECES.



EL TIPO RANGE

EN PYTHON 3, RANGE ES UN TIPO DE DATOS. EL TIPO RANGE ES UNA LISTA INMUTABLE DE NÚMEROS ENTEROS EN SUCESIÓN ARITMÉTICA. INMUTABLE SIGNIFICA QUE, A DIFERENCIA DE LAS LISTAS, LOS RANGE NO SE PUEDEN MODIFICAR. UNA SUCESIÓN ARITMÉTICA ES UNA SUCESIÓN EN LA QUE LA DIFERENCIA ENTRE DOS TÉRMINOS CONSECUTIVOS ES SIEMPRE LA MISMA. UN RANGE SE CREA LLAMANDO AL TIPO DE DATOS CON UNO, DOS O TRES ARGUMENTOS NUMÉRICOS, COMO SI FUERA UNA FUNCIÓN.

EL TIPO RANGE() CON UN ÚNICO ARGUMENTO SE ESCRIBE RANGE(N) Y CREA UNA LISTA INMUTABLE DE N NÚMEROS ENTEROS CONSECUTIVOS QUE EMPIEZA EN 0 Y TERMINA EN N - 1. PARA VER LOS VALORES DEL RANGE(), ES NECESARIO CONVERTIRLO A LISTA MEDIANTE LA FUNCIÓN LIST().

EL TIPO RANGE

```
X = RANGE(10)
```

```
X
```

```
RANGE(0, 10)
```

```
LIST(X)
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

```
RANGE(7)
```

```
RANGE(0, 7)
```

```
LIST(RANGE(7))
```

```
[0, 1, 2, 3, 4, 5, 6]
```



EL TIPO RANGE

SI N NO ES POSITIVO, SE CREA UN RANGE VACÍO.

```
LIST(RANGE(-2))
```

```
[]
```

```
LIST(RANGE(0))
```

```
[]
```

EL TIPO RANGE CON DOS ARGUMENTOS SE ESCRIBE RANGE(M, N) Y CREA UNA LISTA INMUTABLE DE ENTEROS CONSECUTIVOS QUE EMPIEZA EN M Y ACABA EN N - 1.

```
LIST(RANGE(5, 10))
```

```
[5, 6, 7, 8, 9]
```

```
LIST(RANGE(-5, 1))
```

```
[-5, -4, -3, -2, -1, 0]
```

EL TIPO RANGE

SI N ES MENOR O IGUAL QUE M, SE CREA UN RANGE VACÍO.

```
LIST(RANGE(5, 1))
```

```
[]
```

```
LIST(RANGE(3, 3))
```

```
[]
```

EL TIPO RANGE CON TRES ARGUMENTOS SE ESCRIBE RANGE(M, N, P) Y CREA UNA LISTA INMUTABLE DE ENTEROS QUE EMPIEZA EN M Y ACABA JUSTO ANTES DE SUPERAR O IGUALAR A N, AUMENTANDO LOS VALORES DE P EN P. SI P ES NEGATIVO, LOS VALORES VAN DISMINUYENDO DE P EN P.

```
LIST(RANGE(5, 21, 3))
```

```
[5, 8, 11, 14, 17, 20]
```

```
LIST(RANGE(10, 0, -2))
```

```
[10, 8, 6, 4, 2]
```

EL TIPO RANGE

EL VALOR DE P NO PUEDE SER CERO:

```
RANGE(4,18,0)
```

```
TRACEBACK (MOST RECENT CALL LAST) :/ SPAN>
```

```
FILE "<PYSHELL#0>", LINE 1, IN <MODULE>
```

```
RANGE(4,18,0)
```

```
VALUEERROR: RANGE() ARG 3 MUST NOT BE ZERO
```

SI P ES POSITIVO Y N MENOR O IGUAL QUE M, O SI P ES NEGATIVO Y N MAYOR O IGUAL QUE M, SE CREA UN RANGE VACÍO.

```
LIST(RANGE(25, 20, 2))
```

```
[]
```

```
LIST(RANGE(20, 25, -2))
```

```
[]
```

EL TIPO RANGE

EN LOS RANGE(M, N, P), SE PUEDEN ESCRIBIR P RANGE DISTINTOS QUE GENERAN EL MISMO RESULTADO. POR EJEMPLO:

LIST(RANGE(10, 20, 3))

[10, 13, 16, 19]

LIST(RANGE(10, 21, 3))

[10, 13, 16, 19]

LIST(RANGE(10, 22, 3))

[10, 13, 16, 19]

EL TIPO RANGE

EN RESUMEN, LOS TRES ARGUMENTOS DEL TIPO RANGE(M, N, P) SON:

-M: EL VALOR INICIAL

-N: EL VALOR FINAL (QUE NO SE ALCANZA NUNCA)

-P: EL PASO (LA CANTIDAD QUE SE AVANZA CADA VEZ).

SI SE ESCRIBEN SÓLO DOS ARGUMENTOS, PYTHON LE ASIGNA A P EL VALOR 1. ES DECIR RANGE(M, N) ES LO MISMO QUE RANGE(M, N, 1)

SI SE ESCRIBE SÓLO UN ARGUMENTO, PYTHON, LE ASIGNA A M EL VALOR 0 Y A P EL VALOR 1. ES DECIR RANGE(N) ES LO MISMO QUE RANGE(0, N, 1)

EL TIPO RANGE() SÓLO ADMITE ARGUMENTOS ENTEROS. SI SE UTILIZAN ARGUMENTOS DECIMALES, SE PRODUCE UN ERROR.

-TP 35:

ESCRIBIR UN PROGRAMA EN PYTHON QUE COLOQUE EN UNA LISTA LOS NÚMEROS DEL 1 AL 100.

-TP 36:

ESCRIBIR UN PROGRAMA EN PYTHON QUE CREE UNA TUPLA CON LOS MESES DEL AÑO, PIDA NÚMEROS AL USUARIO, SI EL NUMERO ESTA ENTRE 1 Y LA LONGITUD MÁXIMA DE LA TUPLA, MUESTRE EL CONTENIDO DE ESA POSICIÓN SINO MUESTRA UN MENSAJE DE ERROR. EL PROGRAMA TERMINA CUANDO EL USUARIO INTRODUCE UN CERO.

-TP 37:

ESCRIBIR UN PROGRAMA EN PYTHON QUE PIDA UN NUMERO POR TECLADO Y GUARDE EN UNA LISTA SU TABLA DE MULTIPLICAR HASTA EL 10. POR EJEMPLO, SI PIDE EL 5 LA LISTA TENDRÁ:
5,10,15,20,25,30,35,40,45,50

-TP 38:

ESCRIBIR UN PROGRAMA EN PYTHON EN EL QUE DEBEMOS CREAR UNA TUPLA CON NÚMEROS, PEDIR AL USUARIO UN NUMERO POR TECLADO E INDICAR CUANTAS VECES SE REPITE.

-TP 39:

ESCRIBIR UN PROGRAMA QUE SOLICITE AL USUARIO EL INGRESO DE UN STRING Y QUE INFORME POR PANTALLA LA CANTIDAD DE VECES QUE ESE STRING CONTIENE CADA VOCAL.

-TP 40:

ESCRIBIR UN PROGRAMA QUE GUARDE EN UNA LISTA LOS SIGUIENTES PRECIOS, 90, 63, 44, 21, 70, 91 Y MUESTRE POR PANTALLA EL MENOR Y EL MAYOR DE LOS PRECIOS.

-TP 41:

ESCRIBIR UN PROGRAMA EN PYTHON QUE SOLICITE UN NÚMERO ENTERO POSITIVO Y MUESTRE POR PANTALLA LA CUENTA ATRÁS DESDE ESE NÚMERO HASTA CERO SEPARADOS POR COMAS

-TP 42:

ESCRIBIR UN PROGRAMA QUE PIDA AL USUARIO UNA PALABRA Y LUEGO MUESTRE POR PANTALLA UNA A UNA LAS LETRAS DE LA PALABRA INTRODUCIDA EMPEZANDO POR LA ÚLTIMA.

- TP 43

ESCRIBIR UN PROGRAMA EN PYTHON QUE NOS PERMITA GUARDAR LOS NOMBRES DE LOS ESTUDIANTES Y LAS CALIFICACIONES QUE HAN OBTENIDO. CADA ALUMNO PUEDE TENER VARIASE UTILIZAR UN DICCIONARIO PARA GUARDAR ESTA INFORMACIÓN CUYAS CLAVES SERÁN LOS NOMBRES DE LOS ESTUDIANTES Y LOS VALORES SERÁN LISTAS CON LAS CALIFICACIONES DE CADA ALUMNO.

EL PROGRAMA SOLICITARÁ LA CANTIDAD DE ESTUDIANTES QUE VAMOS A INGRESAR, PEDIRÁ SU NOMBRE E IRÁ PIDIENDO SUS CALIFICACIONES HASTA QUE INTRODUCAMOS UN NÚMERO NEGATIVO. AL FINALIZAR EL PROGRAMA NOS MOSTRARÁ LA LISTA DE ALUMNOS Y LA NOTA PROMEDIO OBTENIDA POR CADA UNO DE ELLOS.

ACALARACIÓN: SI SE INTRODUCE EL NOMBRE DE UN ESTUDIANTE QUE YA EXISTE EL PROGRAMA NOS DARÁ UN ERROR.

- TP 44

ESCRIBIR UN PROGRAMA EN PYTHON QUE PERMITA CREAR UNA LISTA DE PALABRAS Y QUE, A CONTINUACIÓN DE TRES OPCIONES:

- * CONTAR: ME PIDE UNA CADENA, Y ME DICE CUANTAS VECES APARECE EN LA LISTA
- * MODIFICAR: ME PIDE UNA CADENA, Y OTRA CADENA A MODIFICAR, Y MODIFICA TODAS LAS APARICIONES DE LA PRIMERA POR LA SEGUNDA EN LA LISTA.
- * ELIMINAR: ME PIDE UNA CADENA, Y LA ELIMINA DE LA LISTA.
- * MOSTRAR: MUESTRA LA LISTA DE CADENAS
- * TERMINAR

- TP 45

ESCRIBIR UN PROGRAMA EN PYTHON QUE DECLARE TRES LISTAS 'LISTA1', 'LISTA2' Y 'LISTA3' DE CINCO ENTEROS CADA UNA, PIDA VALORES PARA 'LISTA1' Y 'LISTA2' Y CALCULE $LISTA3 = LISTA1 + LISTA2$.

- TP 46

ESCRIBIR UN PROGRAMA EN PYTHON EN EL CUAL VAMOS A UTILIZAR UN DICCIONARIO PARA GUARDAR LOS PRECIOS DE LOS DISTINTOS UTTILES DE UNA LIBRERIA. EL PROGRAMA PEDIRÁ EL NOMBRE DEL PRODUCTO Y LA CANTIDAD QUE SE HA VENDIDO Y NOS MOSTRará EL PRECIO FINAL A PARTIR DE LOS DATOS GUARDADOS EN EL DICCIONARIO. SI EL PRODUCTO NO EXISTE NOS DARÁ UN ERROR. TRAS CADA CONSULTA EL PROGRAMA NOS PREGUNTARÁ SI QUEREMOS HACER OTRA CONSULTA

-TP 47:

ESCRIBIR UN PROGRAMA QUE REALICE UNA AGENDA TELEFONICA. EN LA AGENDA SE PODRÁN GUARDAR NOMBRES Y NÚMEROS DE TELÉFONO. EL PROGRAMA NOS DARÁ EL SIGUIENTE MENÚ:

- **AÑADIR/MODIFICAR:** NOS PIDE UN NOMBRE. SI EL NOMBRE SE ENCUENTRA EN LA AGENDA, DEBE MOSTRAR EL TELÉFONO Y, OPCIONALMENTE, PERMITIR MODIFICARLO SI NO ES CORRECTO. SI EL NOMBRE NO SE ENCUENTRA, DEBE PERMITIR INGRESAR EL TELÉFONO CORRESPONDIENTE.
- **BUSCAR:** NOS PIDE UNA CADENA DE CARACTERES, Y NOS MUESTRAS TODOS LOS CONTACTOS CUYOS NOMBRES COMIENCEN POR DICHA CADENA.
- **BORRAR:** NOS PIDE UN NOMBRE Y SI EXISTE NOS PREGUNTARÁ SI QUEREMOS BORRARLO DE LA AGENDA.
- **LISTAR:** NOS MUESTRA TODOS LOS CONTACTOS DE LA AGENDA.

IMPLEMENTAR EL PROGRAMA CON UN DICCIONARIO.