_1type/_0keywords.py

```
"""_____________________________Keywords_____________________________
None                continue            global              pass
True                def                 if                  raise
and                 del                 import              return
as                  elif                in                  try
assert              else                is                  while
async               except              lambda              with
await               finally             nonlocal            yield
break               for                 not                       """

Booleans      = False, True, None
LógicsBools   = "and, or, not"
Pertenece     = "in, is"                # presente en una secuencia, si dos objetos son el mismo objeto.
Interruptus   = "yield"                 # devuelve un valor sin finalizar la ejecución de la función.
Gestion_recur = "with"
eliminar      = "del"                   # elimina variables, objetos listas, dicionarios,sets, atributos.

flujo         = "if, elif, else"        # mientras es True

Bucles        = "for, while"            # for itera un nº especifico de veces, while mientras una condición sea verdadera.
Flujo         = "break, continue"       # controlar el flujo dentro de los bucles. break sale de un bucle antes de que se complete,continue omite la ejecución del resto del cuerpo del bucle y continuar con la siguiente iteración.

Funcion_Valor = "def, return "          # define una función, y return devolve un valor de una función.
Class         = "class, pass"           # definir una nueva clase, y pass marcador de posición cuando no se requiere ninguna acción.
F_anonym      = "lambda "               # crea funciones anónimas en Python.
F_asyncrona   = "async, await"          # define funcion asyncona, await indica que el bucle puede seguir mientras espera

Excepciones   = "try, except, finally"  # excepciones. try se utiliza para definir un bloque de código en el que pueden ocurrir excepciones, except se utiliza para manejar excepciones específicas que pueden ocurrir en el bloque try, y finally se utiliza para definir un bloque de código que siempre se ejecuta, independientemente de si se produce una excepción o no.
Manualexcept  = "raise "                # Esta palabra clave se utiliza para generar una excepción manualmente en Python.

Import        = "import, from, as"      # módulos y paquetes. import se utiliza para importar un módulo o paquete, from se utiliza para importar partes específicas de un módulo o paquete, y as se utiliza para renombrar los módulos o partes importadas.
Accesoexterno = "global, nonlocal"      # acceder a variables fuera del ámbito actual en Python. global se utiliza dentro de una función para indicar que una variable se debe buscar en el ámbito global, mientras que nonlocal se utiliza dentro de una función anidada para indicar que una variable se debe buscar en el ámbito de la función externa más cercana.

Asegurate     = "assert"
```

```
with open('archivo.txt', 'r') as archivo:
    contenido = archivo.read()
                                        # with asegura que el archivo se cierre correctamente después de que se lea, incluso si ocurre una excepción durante la lectura.
None                                    # Representa la ausencia de un valor

"""Funciones Integradas/ BUild-in Functions"""
"""Conversión de tipos"""
abs             # Devuelve el valor absoluto de un número.
bin             # Convierte un número en una cadena binaria.
bool            # Convierte un valor en un booleano.
bytearray       # Devuelve un objeto bytearray.
bytes           # Devuelve un objeto bytes.
chr             # Convierte un número entero a su carácter Unicode correspondiente.
complex         # Crea un número complejo.
dict            # Crea un diccionario.
float           # Convierte un número en un flotante.
frozenset       # Devuelve un objeto frozenset.
hex             # Convierte un número en una cadena hexadecimal.
int             # Convierte un valor a un número entero.
list            # Crea una lista.
memoryview      # Devuelve una vista de memoria.
object          # Crea un objeto.
oct             # Convierte un número en una cadena octal.
ord             # Convierte un carácter en su valor Unicode.
range           # Crea una secuencia de números.
set             # Crea un conjunto.
slice           # Crea un objeto slice.
str             # Convierte un valor en una cadena de texto.
tuple           # Crea una tupla.
"""Manejo Atributos/clases"""
getattr         # Obtiene el valor de un atributo de un objeto.
hasattr         # Verifica si un objeto tiene un atributo específico.
setattr         # Establece un atributo en un objeto.
delattr         # Elimina un atributo de un objeto.
isinstance      # Verifica si un objeto es una instancia de una clase.
issubclass      # Verifica si una clase es una subclase de otra.
property        # Devuelve un objeto propiedad.
classmethod     # Define un método de clase.
staticmethod    # Define un método estático.
super           # Devuelve un objeto proxy para delegar llamadas a métodos en una clase base.
type            # Devuelve el tipo de un objeto o crea un nuevo tipo de objeto.
"""Entrada/salida"""
input           # Permite la entrada de datos del usuario.
open            # Abre un archivo y lo retorna como un objeto de archivo.
print           # Imprime un mensaje en la salida estándar.
"""Funciones matemáticas"""
abs             # Devuelve el valor absoluto de un número.
divmod          # Devuelve el cociente y el resto de la división.
max             # Devuelve el valor máximo en una secuencia.
min             # Devuelve el valor mínimo en una secuencia.
pow             # Calcula la potencia de un número.
round           # Redondea un número flotante.
sum             # Suma los elementos de una secuencia.
```

```
"""Funciones iteracion"""
all             # Devuelve True si todos los elementos de una secuencia son verdaderos.
any             # Devuelve True si algún elemento de una secuencia es verdadero.
enumerate       # Retorna un objeto enumerador.
filter          # Filtra elementos de una secuencia.
iter            # Retorna un iterador para el objeto dado.
map             # Aplica una función a todos los elementos en una secuencia.
next            # Recupera el próximo ítem del iterador.
range           # Crea una secuencia de números.
reversed        # Retorna un iterador que accede a la secuencia en orden inverso.
zip             # Combina varios iterables, retornando un iterador de tuplas.
"""Manipulacion/Evaluacion código"""
compile         # Compila el código fuente en un objeto que puede ser ejecutado.
eval            # Evalúa una cadena como una expresión Python.
exec            # Ejecuta un bloque dinámico de código Python.
__import__      # Función avanzada para importar módulos.
"""Cadenas"""
ascii           # Retorna una representación en cadena con caracteres ASCII.
format          # Formatea una cadena de acuerdo con el formato especificado.
repr            # Retorna una representación de cadena de un objeto.
"""Miscelaneo"""
breakpoint      # Introduce un punto de interrupción en el depurador.
callable        # Verifica si un objeto es llamable.
dir             # Retorna una lista de atributos válidos para el objeto.
globals         # Retorna un diccionario del espacio de nombres global actual.
hash            # Retorna el valor hash de un objeto.
help            # Invoca el sistema de ayuda de Python.
id              # Retorna el identificador único de un objeto.
len             # Retorna la longitud de un objeto.
locals          # Retorna un diccionario del espacio de nombres local actual.
sorted          # Ordena los elementos de una secuencia.
vars            # Retorna el diccionario `__dict__` de un objeto.

#Conversores de tipo
type(42)
x = int("10")                   #entero 10.
x = float("3.14")               #flotante 3.14.
x = str(42)                     #"42".
x = bool(0)                     #False.   bool(1)=True
x = list((1, 2, 3))             #[1, 2, 3].
x = tuple([1, 2, 3])            #(1, 2, 3).
x = dict([(1, 'uno'), (2, 'dos')]) #{1: 'uno', 2: 'dos'}.
x = set([1, 2, 3])              #{1, 2, 3}.
x = complex(2j +2)              #2j+2?
x = chr(65)                     #"A" código Unicode 65.
x = ord('A')                    #código Unicode 65
x = bin(10)                     #entero 10 a binaria '0b1010'.
x = hex(16)                     #entero 16 a la cadena hexadecimal '0x10'.
x = oct(8)                      #entero 8 a la cadena octal '0o10'.
lista=[1,2,7,5,4,3]
x = ascii(lista)                #American Standard Code for Information Interchange
identify = id("Alicia")                                     #140395687955056  nº único
valor_hash = hash("python")                                 # Devuelve el valor hash de un
objeto.
```

```
codigo = compile('print("Hola, mundo!")', 'test', 'exec') #Compila una expresión Python en un objeto código o AST.

atributos = dir(list)                    #atributos contendrá una lista de atributos y métodos de la clase list.
valor = getattr(lista, "append")         #valor contendrá el método append si está presente en lista. Obtiene el valor de un atributo de un objeto.
setattr(lista, "nombre", "Mi Lista")     #El objeto lista tendrá un nuevo atributo nombre con el valor "Mi Lista". Establece el valor de un atributo de un objeto.
delattr(lista, "nombre")                 #Si lista tiene un atributo llamado nombre, será eliminado. Elimina un atributo de un objeto.
resultado = eval("2 + 3") #Evalúa una expresión Python en una cadena.
exec('print("Hola, mundo!")') # Ejecuta código Python dinámicamente.
#archivo = open("archivo.txt", "r") #Abre un archivo en modo de lectura, escritura o ambos.

reverso = reversed([1, 2, 3])                    #Devuelve un iterador que recorre un iterable en orden inverso.
enumerado = enumerate(["a", "b", "c"])           #Enumera los elementos de un iterable junto con sus índices.
numeros = range(1, 5)                            #Genera una secuencia de números.
combinacion = zip([1, 2, 3], ['a', 'b', 'c'])    #Combina dos o más iterables en tuplas.
ordenado = sorted([3, 1, 2])                     #Ordena los elementos de un iterable.
filtrado = filter(lambda x: x > 0, [-1, 0, 1, 2])  #Filtra los elementos de un iterable según una función.
mapeado = map(lambda x: x * 2, [1, 2, 3])        #Aplica una función a cada elemento de un iterable.
ayuda=help(int)

todos = all([True, False, True])                 #True si todos los elementos de un iterable son verdaderos.
alguno = any([True, False, True])                #True si algún elemento de un iterable es verdadero.
es_callable = callable(print)                    #True ya que print es una función.
tiene_atributo = hasattr(lista, "append")        #True si lista tiene el atributo append, de lo contrario, False.
es_instancia = isinstance(lista, list)           #True si lista es una instancia de la clase list, de lo contrario, False. Verifica si un objeto es una instancia de una clase.
```