

Curso de Python

Clase 01



¡Bienvenidxs!

Contenido

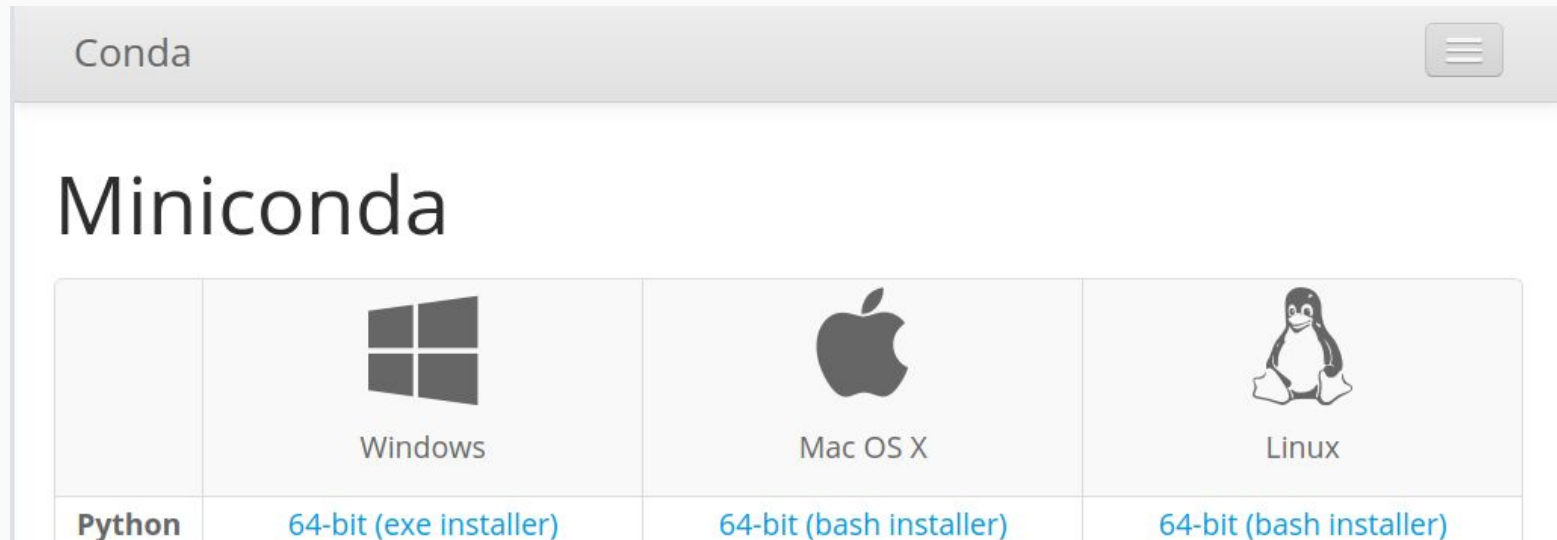
- Instalación
- Variables y tipos de datos
- Operadores lógicos y condicionales
- Cadenas y ciclos de control

Instalación

INSTALACIÓN

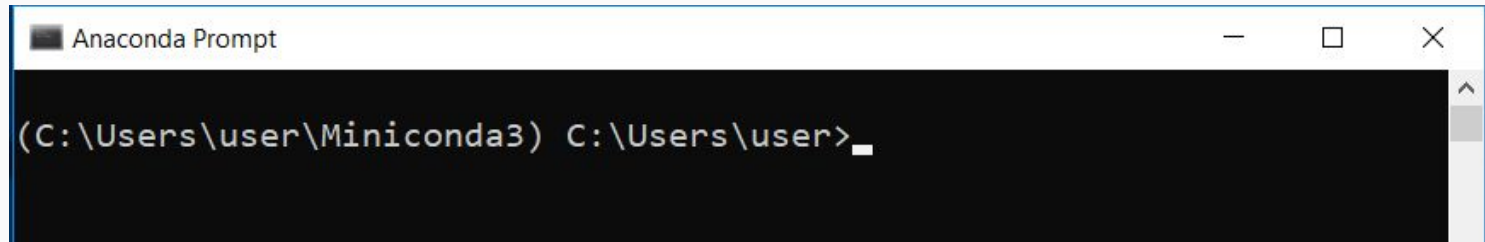
Adicional a la instalación del sitio oficial de python.org o si aún no tienes instalado Python, se sugiere realizar la instalación usando Miniconda

Sitio oficial de Miniconda <https://conda.io/miniconda.html>



INSTALACIÓN PARA WINDOWS

1. Descargar el instalador para la versión de Python 3.7 según la versión de Windows que se tenga
 - a. Para windows de 64 bits descargar [aquí](#)
 - b. Para windows de 32 bits descargar [aquí](#)
2. Dar doble click sobre el archivo descargado
3. Seguir las instrucciones y si no está muy seguro de alguna configuración, dejar las opciones ya establecidas y continuar adelante.
4. Después que la instalación haya terminado ir al menú de **Inicio** y abrir la opción **Anaconda Prompt**, esto abrirá una ventana de comandos como la siguiente:



Probando la instalación

Ejecutar la instrucción `python -V` y presionar ENTER en la ventana de comandos abierta anteriormente, como se muestra a continuación:

```
C:\Users\user> python -V  
Python 3.6.5 :: Anaconda, Inc.  
  
C:\Users\user>
```

El mensaje indica la versión de Python, en este caso la 3.6.5, pero puede variar según la versión descargada, lo importante es que aparezca la versión de Python y no un mensaje de error.

Con esto estamos **listos** para continuar.

INSTALACIÓN PARA MAC OS

1. Descargar el instalador para la versión de Python 3.7 desde [aquí](#)
2. Abrir una ventana de Terminal
3. Ejecutar los siguiente comandos:

```
$ cd Downloads
Downloads $ bash Miniconda3-latest-MacOSX-x86_64.sh
...
```

4. Seguir las instrucciones de la pantalla, si está inseguro de algunas opciones, dejarlas como están y continuar con la instalación.
5. Para que los cambios tomen efecto es necesario cerrar la Terminal y abrirla nuevamente.

Probando la instalación

Ejecutar la instrucción `python -V` y presionar ENTER en la Terminal abierta anteriormente, como se muestra a continuación:

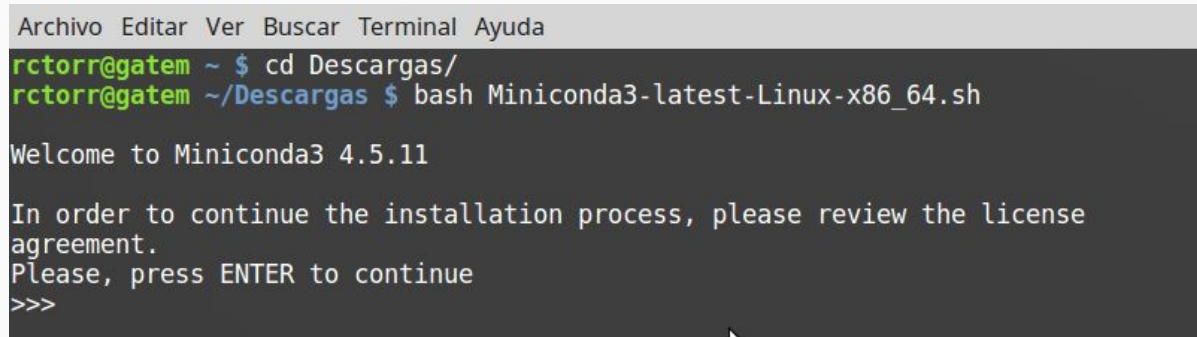
```
$ python -V  
Python 3.6.5 :: Anaconda, Inc.  
  
$
```

El mensaje indica la versión de Python, en este caso la 3.6.5, pero puede variar según la versión descargada, lo importante es que aparezca la versión de Python y no un mensaje de error.

Con esto estamos **listos** para continuar.

INSTALACIÓN PARA LINUX

1. Descargar el instalador para la versión de Python 3.7 tomando en cuenta la arquitectura de Linux instalada
 - a. Para Linux de 64 bits descargar desde [aquí](#)
 - b. Para Linux de 32 bits descargar desde [aquí](#)
2. Abrir una ventana de ventana de Terminal y ejecutar:

A screenshot of a terminal window with a menu bar at the top containing 'Archivo', 'Editar', 'Ver', 'Buscar', 'Terminal', and 'Ayuda'. The terminal shows a user 'rctorr@gatem' in a directory '~' running the command 'cd Descargas/'. The prompt changes to '~/Descargas \$' and the user runs 'bash Miniconda3-latest-Linux-x86_64.sh'. The terminal then displays 'Welcome to Miniconda3 4.5.11', followed by 'In order to continue the installation process, please review the license agreement.' and 'Please, press ENTER to continue'. The prompt is now '>>>'.

```
Archivo  Editar  Ver  Buscar  Terminal  Ayuda
rctorr@gatem ~ $ cd Descargas/
rctorr@gatem ~/Descargas $ bash Miniconda3-latest-Linux-x86_64.sh

Welcome to Miniconda3 4.5.11

In order to continue the installation process, please review the license
agreement.
Please, press ENTER to continue
>>>
```

3. Seguir las instrucciones de la pantalla y si está inseguro de algunas opciones, dejarlas como están y continuar con la instalación.

Probando la instalación

4. Para que los cambios tomen efecto es necesario cerrar la Terminal y abrirla nuevamente.
5. Ejecutar la instrucción `python -V` y presionar ENTER en la Terminal abierta como se muestra a continuación:

```
~ $ python -V
Python 3.7.0
~ $
```

El mensaje indica la versión de Python, en este caso la 3.7.0, pero puede variar según la versión descargada, lo importante es que aparezca la versión de Python y no un mensaje de error y con esto estamos **listos** para continuar.

HOLA MUNDO - PROBANDO LA INSTALACIÓN

Antes de iniciar a realizar programas, se recomienda crear una estructura de carpetas para este curso de la siguiente manera:

```
Curso-de-Python
├── Clase-01
│   └── hola-mundo.py
```

Curso-de-Python
 Carpeta que contendrá todo los archivos del curso

Clase-01
 Carpeta que contiene los archivos sólo para la Clase 1

hola-mundo.py
 Archivo con terminación en .py que contendrá el programa en Python a ejecutar.

HOLA MUNDO - PROBANDO LA INSTALACIÓN

Ahora si vamos a crear nuestro primer programa, en Python comúnmente se llaman scripts.

Nombre del script: `hola-mundo.py`

Descripción:

Este script mostrará el mensaje “Hola Mundo de Python!” en la pantalla después de haberlo ejecutado.

Ejemplo de ejecución:

```
Curso-de-Python/Clase-01 $ python hola-mundo.py
Hola Mundo de Python!
Curso-de-Python/Clase-01 $
```

INSTALANDO IPYTHON

IPython es una mejora para el intérprete de Python al momento de ejecutar comandos en tiempo real y para poder usarlo, lo primero es instalarlo.

La instalación se realiza desde la terminal o consola usando el comando **pip** de a siguiente forma:

```
Curso-de-Python/Clase-01 $ pip install ipython
Collecting ipython
  Using cached
https://files.pythonhosted.org/packages/1b/e2/ffb8c1b574f972cf4183b0aac8f16b57f1e3bbe876b3
1555b107ea3fd009/ipython-7.1.1-py3-none-any.whl
Collecting pygments (from ipython)
...
Successfully installed backcall-0.1.0 decorator-4.3.0 ipython-7.1.1 ipython-genutils-0.2.0
jedi-0.13.1 parso-0.3.1 pexpect-4.6.0 pickleshare-0.7.5 prompt-toolkit-2.0.7
ptyprocess-0.6.0 pygments-2.2.0 six-1.11.0 traitlets-4.3.2 wcwidth-0.1.7

Curso-de-Python/Clase-01 $
```

USANDO IPYTHON

Para usar ipython, sólo basta con escribirlo en la consola y presionar enter

```
Curso-de-Python/Clase-01 $ ipython
Python 3.7.1 (default, Oct 23 2018, 19:19:42)
Type 'copyright', 'credits' or 'license' for more information
IPython 7.1.1 -- An enhanced Interactive Python. Type '?' for help.
```

```
In [1]: import this
```

```
The Zen of Python, by Tim Peters
```

```
Beautiful is better than ugly.
Explicit is better than implicit.
```

```
...
```

```
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!
```

```
In [2]:
```

Variables y tipos de datos

ENTEROS DECIMALES Y BINARIOS

```
In [1]: # Asignando enteros a variables
```

```
In [2]: a = 50
```

```
In [3]: b = 75
```

```
In [4]: # Convirtiendo decimal a binario y binario a decimal
```

```
In [5]: bin(3)
```

```
Out[5]: '0b11'
```

```
In [6]: bin(a)
```

```
Out[6]: '0b110010'
```

```
In [7]: c = bin(b)
```

```
In [8]: c # Contiene el valor de 75 en binario
```

```
Out[8]: '0b1001011'
```

Nombre del script: `dec2bin.py`

Descripción:

Este script convertirá un valor en decimal a binario e imprimirá el resultado

Ejemplo de ejecución:

```
Curso-de-Python/Clase-01 $ python dec2bin.py
Valor en decimal: 53
Valor en binario: 0b110101
Curso-de-Python/Clase-01 $
```



ACTIVIDAD

Modifica el script `dec2bin.py` para que imprima el valor 255 en binario.

Ejemplo de ejecución:

```
Curso-de-Python/Clase-01 $ python dec2bin.py
Valor en decimal: 255
Valor en binario: 0b11111111
Curso-de-Python/Clase-01 $
```



ACTIVIDAD

Crear el script que se pide a continuación

Nombre del script:

`bin2dec.py`

Descripción:

Este script convertirá un valor en binario a decimal e imprimirá el resultado en pantalla.

Sugerencia:

Ejecutar `dir(__builtin__)` en IPython.

Ejemplo de ejecución:

```
Curso-de-Python/Clase-01 $ python bin2dec.py
Valor en binario: 0b110101
Valor en decimal: 53
Curso-de-Python/Clase-01 $
```

OPERACIONES CON ENTEROS Y CADENAS CON FORMATO

```
In [1]: # Operaciones con enteros
```

```
In [2]: 50 * 75
```

```
Out[2]: 3750
```

```
In [3]: 50 + 75
```

```
Out[3]: 125
```

```
In [4]: 150 / 75
```

```
Out[3]: 2.0
```

```
In [5]: 75 - 50
```

```
Out[5]: 25
```

```
In [6]: # Cadenas
```

```
In [7]: "Hola Python"
```

```
Out[7]: 'Hola Python'
```

```
In [8]: # Repetición de cadenas
```

```
In [9]: "Hola" * 3
```

```
Out[9]: 'HolaHolaHola'
```

```
In [10]: # Cadenas con formato
```

```
In [11]: "Hola {}".format("clase")
```

```
Out[11]: 'Hola clase'
```

```
In [12]: "Hola {}".format("Cap!")
```

```
Out[12]: 'Hola Cap!'
```

```
In [12]: "Hola {}".format(5)
```

```
Out[12]: 'Hola 5'
```

TABLA-DEL-3 - VARIABLES Y TIPOS DE DATOS

Nombre del script: tabla-del-3.py

Descripción:

Imprimir la tabla del 3 en la pantalla (salida estándar)

Ejemplo de ejecución:

```
Curso-de-Python/Clase-01 $ python tabla-del-3.py
TABLA DEL 3
-----
3 x 1 = 3
3 x 2 = 6
3 x 3 = 9
...
3 x 8 = 24
3 x 9 = 27
3 x 10 = 30
-----
```

TABLA-DEL-4 - VARIABLES Y TIPOS DE DATOS

Nombre del script: tabla-del-4.py

Descripción:

Imprimir la tabla del 4 en la pantalla (salida estándar) haciendo uso de variables y formatos

Ejemplo de ejecución:

```
Curso-de-Python/Clase-01 $ python tabla-del-4.py
TABLA DEL 4
-----
4 x 1 = 4
4 x 2 = 8
4 x 3 = 12
...
4 x 8 = 32
4 x 9 = 36
4 x 10 = 40
-----
```

ACTIVIDAD

Crear el script que se pide a continuación en **no más de 30 seg.**

Nombre del script:

tabla-del-5.py

Descripción:

Imprimir la tabla del 5 en la pantalla (salida estándar) haciendo uso de variables y formato.



Ejemplo de ejecución:

```
Curso-de-Python/Clase-01 $ python tabla-del-5.py
TABLA DEL 5
-----
5 x 1 = 5
5 x 2 = 10
5 x 3 = 15
...
5 x 8 = 40
5 x 9 = 45
5 x 10 = 50
-----
```


OPERACIONES CON NÚMEROS FLOTANTES E IMPRESIÓN CON FORMATO

```
In [1]: # Operaciones con flotantes
```

```
In [2]: 50 * 2.5
```

```
Out[2]: 125.0
```

```
In [3]: 50 + 2.5
```

```
Out[3]: 52.5
```

```
In [4]: 50 / 2.5
```

```
Out[4]: 20.0
```

```
In [5]: 50 - 2.5
```

```
Out[5]: 47.5
```

```
In [6]: a = 50 - 2.5
```

```
In [7]: a
```

```
Out[7]: 47.5
```

```
In [8]: # Imprimiendo flotantes
```

```
In [9]: print(123.45)
```

```
Out[9]: 123.45
```

```
In [10]: print(a)
```

```
Out[10]: 47.5
```

```
In [11]: # Imprimiendo con formato
```

```
In [12]: print("a= {}".format(a))
```

```
Out[12]: a= 47.5
```

```
In [12]: print("a= {:10.2f}".format(a))
```

```
Out[12]: a=          47.50
```

LISTA-DE-PRODUCTOS - VARIABLES Y TIPOS DE DATOS

Nombre del script: lista-de-productos.py

Descripción:

Imprimir una lista de 5 productos en la salida estándar en forma tabular incluyendo el nombre del producto y su precio.

Ejemplo de ejecución:

```
Curso-de-Python/Clase-01 $ python lista-de-productos.py
-----
PRODUCTO                                | PRECIO
-----
Automóvil                               | 150000.00
Bicicleta                               | 13000.00
Chamarra                                | 3999.99
Laptop Thinkpad (Descuento incluido)    | 21750.00
Gafas de realidad virtual Lenovo con sable laser | 5000.00
-----
```



ACTIVIDAD

Crear el script que se pide a continuación.

Nombre del script:

lista-de-productos-total
.py

Descripción:

Imprimir una lista de 5 productos en la salida estándar en forma tabular incluyendo el nombre del producto, precio, cantidad y subtotal. En la última línea de la tabla se deberá mostrar el total.

Ejemplo de ejecución:

```
Clase-01 $ python lista-de-productos-total.py
-----
PRODUCTO                                     | PRECIO    | CANT | SUBTOTAL
-----
Automóvil                                   | 150000.00 | 1    | 150000.00
Bicicleta                                  | 13000.00  | 2    | 26000.00
Chamarra                                   | 3999.99   | 2    | 7999.98
Laptop Thinkpad (Descuento incluido)       | 21750.00  | 1    | 21750.00
Gafas de realidad virtual Lenovo con sable laser | 5000.00  | 2    | 10000.00
-----
Total | 215749.98
-----
```

Operadores lógicos y condicionales

VALORES Y OPERADORES LÓGICOS

```
In [1]: # Valores lógicos
```

```
In [2]: True
```

```
Out[2]: True
```

```
In [3]: False
```

```
Out[3]: False
```

```
In [4]: # Operadores lógicos
```

```
In [5]: True and False
```

```
Out[5]: False
```

```
In [6]: True or False
```

```
Out[6]: True
```

```
In [7]: not True
```

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

TABLA-VERDAD-AND - OPERADORES LÓGICOS

Nombre del script: tabla-verdad-and.py

Descripción:

Imprimir la tabla de verdad del operador lógico AND en la salida estándar en forma tabular incluyendo operador1, operador2 y resultado.

Ejemplo de ejecución:

```
Curso-de-Python/Clase-01 $ python tabla-verdad-and.py
```

```
          Tabla de verdad de AND
```

```
-----  
Operador 1 | Operador 2 | Resultado  
-----
```

```
True       | True       | True  
True       | False      | False  
False      | True       | False  
False      | False      | False  
-----
```

ACTIVIDAD

Crear el script que se pide a continuación en **no más de 3 mins.**

Nombre del script:

tabla-verdad-or.py

Descripción:

Imprimir la tabla de verdad del operador lógico OR en la salida estándar en forma tabular incluyendo operador1, operador1 y resultado.



Ejemplo de ejecución:

```
Clase-01 $ python tabla-verdad-or.py
```

```
                Tabla de verdad de OR
-----
Operador 1 | Operador 2 | Resultado
-----
True       | True       | True
True       | False      | True
False      | True       | True
False      | False      | False
-----
```

CENTRANDO CADENAS Y FORMATOS VARIABLES

```
In [1]: # Centrando cadenas
```

```
In [2]: "Hola".center(60)
```

```
Out[2]: '                Hola                '
```

```
In [3]: "-" * 10
```

```
Out[3]: '-----'
```

```
In [4]: ("-" * 10).center(30)
```

```
Out[4]: '          -----          '
```

```
In [5]: # Formatos variables
```

```
In [6]: print("| {:{} } | {:{} } |".format("Nombre", 10, "Apellido", 15))
```

```
| Nombre      | Apellido      |
```

```
In [7]: print("| {:{} } | {:{} } |".format("Nombre", 20, "Apellido", 10))
```

```
| Nombre                | Apellido  |
```


ACTIVIDAD

Crear el script que se pide a continuación.

Nombre del script:

tabla-verdad-not.py

Descripción:

Imprimir la tabla de verdad del operador lógico NOT en la salida estándar en forma tabular incluyendo operador1 y resultado. La tabla tiene estar centrada en la ventana y se debe poder ajustar el ancho de las columnas y por consecuencia el ancho de la tabla en menos de 5 seg.



Ejemplo de ejecución:

```
Clase-01 $ python tabla-verdad-not.py
```

```
                Tabla de verdad de NOT
-----
Operador 1      | Resultado
-----
True            | False
False           | True
-----
```

```
Clase-01 $ python tabla-verdad-not.py
```

```
                Tabla de verdad de NOT
-----
Operador 1 | Resultado
-----
True       | False
False      | True
-----
```

ESTRUCTURA DE CONTROL IF Y VALORES LÓGICOS

```
In [1]: # Estructura if
```

```
In [2]: if True:
...:     print("Imprime esto si es verdadero")
...:
Imprime esto si es verdadero
```

```
In [3]: # Estructura if - else
```

```
In [4]: if False:
...:     print("Imprime esto si es verdadero")
...: else:
...:     print("Imprime esto si es falso")
...:
Imprime esto si es falso
```

```
In [5]:
```

LEE-ENTEROS - ESTRUCTURA DE CONTROL IF Y VALORES LÓGICOS

Nombre del script: lee-enteros.py

Descripción:

Lee un número entero e imprime su valor multiplicado por 100, si el valor leído no es un número entero se deberá mostrar un mensaje de error.

Ejemplo de ejecución:

```
Curso-de-Python/Clase-01 $ python lee-enteros.py
```

```
Escribe un número entero: 51
```

```
Resultado: 5100
```

```
Curso-de-Python/Clase-01 $ python lee-enteros.py
```

```
Escribe un número entero: 51.0
```

```
Error: 51.0 no es un entero
```

CADENAS NUMÉRICAS Y OPERADORES CONDICIONALES

```
In [1]: # Para conocer si una cadena es numérica
```

```
In [2]: "123".isdecimal()
```

```
Out[2]: True
```

```
In [3]: # Operadores condicionales
```

```
In [4]: 123 > 123 # Mayor que
```

```
Out[4]: False
```

```
In [4]: 123 >= 123 # Mayor o igual que
```

```
Out[4]: True
```

```
In [4]: 123 == 123 # Igual a
```

```
Out[4]: True
```

```
In [4]: 123 != 123 # Diferente a
```

```
Out[4]: False
```

LEE-POSITIVOS - OPERADORES CONDICIONALES

Nombre del script: lee-enteros-positivos.py

Descripción:

Lee un número entero positivo e imprime su valor multiplicado por 1000, si el valor leído no es un número entero positivo se deberá mostrar un mensaje de error.

Ejemplo de ejecución:

```
Curso-de-Python/Clase-01 $ python lee-enteros-positivos.py
```

```
Escribe un número entero positivo: -51  
Error: -51 no es un entero positivo
```

```
Curso-de-Python/Clase-01 $ python lee-enteros-positivos.py
```

```
Escribe un número entero: uno  
Error: uno no es un entero positivo
```

ACTIVIDAD

Crear el script que se pide a continuación.

Nombre del script:

lee-edad.py

Descripción:

Leer una valor de edad desde la entrada estándar (teclado) e imprime un mensaje indicando si es mayor de edad o no.

Considerar que una persona es mayor de edad si tiene 18 o más años cumplidos.

Si el valor leído no es una edad, entonces se imprime un mensaje de error.



Ejemplo de ejecución:

```
Clase-01 $ python lee-edad.py
```

```
Escribe tu edad: 15
```

```
Eres menor de edad, puedes tomar malteadas!
```

```
Clase-01 $ python lee-edad.py
```

```
Escribe tu edad: 18
```

```
Eres mayor de edad, ya puedes tomar gaseosas!
```

```
Clase-01 $ python lee-edad.py
```

```
Escribe tu edad: quince
```

```
Error: quince no es valor adecuado para una edad
```

ACTIVIDAD

Crear el script que se pide a continuación.

Nombre del script:

lee-email.py

Descripción:

Leer una email desde la entrada estándar (teclado) e imprime un mensaje indicando si es o no una dirección de correo válida.



Ejemplo de ejecución:

```
Clase-01 $ python lee-email.py

Escribe tu e-mail: user@demonio.com
La dirección user@demonio.com es una e-mail válida

Clase-01 $ python lee-email.py

Escribe tu e-mail: user-demonio.com
La dirección user-demonio.com no es una e-mail válida

Clase-01 $ python lee-email.py

Escribe tu e-mail: user@demonio
La dirección user@demonio no es una e-mail válida
```

Cadenas y ciclos de control

TRIÁNGULOS - CADENAS

Nombre del script: `triangulos.py`

Descripción:

Imprimir en la salida estándar un triángulo cuya base es de longitud `n` y centrado a lo ancho de la terminal.

Ejemplo de ejecución:

```
Curso-de-Python/Clase-01 $ python triangulos.py
```

```
  #  
 ###  
#####  
#####  
#####  
#####
```

```
Curso-de-Python/Clase-01 $
```

CICLO FOR Y RANGE

```
In [1]: # Ciclo for repite 3 veces
```

```
In [2]: for i in range(3):  
...:     print(i)  
...:
```

```
0  
1  
2
```

```
In [3]: # Repite 3 veces iniciando  
# en 1
```

```
In [4]: for i in range(1, 4):  
...:     print(i)  
...:
```

```
1  
2  
3
```

```
In [5]: # Repite 3 veces iniciando  
# en 1 y sólo impares
```

```
In [6]: for i in range(1, 6, 2):  
...:     print(i)  
...:
```

```
1  
3  
5
```

TRIÁNGULOS - CICLO FOR

Nombre del script: triangulos-ciclos.py

Descripción:

Imprimir en la salida estándar un triángulo cuya base es de longitud n y centrado a lo ancho de la terminal usando ciclos y además preguntando al inicio el valor de n que tiene que ser un valor entero positivo e impar mayor o igual a 3.

Ejemplo de ejecución:

```
Curso-de-Python/Clase-01 $ python triangulos-ciclos.py
Longitud de la base del triángulo n = 7
```

```
      #
     ###
    #####
   #####
```

```
Curso-de-Python/Clase-01 $ python triangulos-ciclos.py
```

```
Longitud de la base del triángulo n = 8
La longitud tiene que ser un valor impar
```

PRODUCTOS Y TOTALES - CICLO FOR

Nombre del script: lista-de-productos-total-ciclos.py

Descripción:

Modifica el script lista-de-productos-total.py para que el total se calcule usando ciclos

Ejemplo de ejecución:

```
Curso-de-Python/Clase-01 $ python lista-de-productos-total-ciclos.py
```

| PRODUCTO | PRECIO | CANT | SUBTOTAL |
|--|-----------|------|-----------|
| Automóvil | 150000.00 | 1 | 150000.00 |
| Bicicleta | 13000.00 | 2 | 26000.00 |
| Chamarra | 3999.99 | 2 | 7999.98 |
| Laptop Thinkpad (Descuento incluido) | 21750.00 | 1 | 21750.00 |
| Gafas de realidad virtual Lenovo con sable laser | 5000.00 | 2 | 10000.00 |
| Total | | | 215749.98 |

ACTIVIDAD

Crear el script que se pide a continuación.

Nombre del script:

tabla-del-n.py

Descripción:

Modificar el script tabla-del-5.py para leer N desde el teclado e imprimir la tabla del N en la salida estándar usando ciclos.

N tiene que ser un entero positivo mayor o igual a 2, si el valor leído es incorrecto se muestra un mensaje de error y se termina el script.



Ejemplo de ejecución:

```
Clase-01 $ python tabla-del-n.py
```

```
Escribe el número de la tabla a imprimir n= 11
```

```
TABLA DEL 11
```

```
-----
```

```
11 x 1 = 11
```

```
11 x 2 = 22
```

```
11 x 3 = 33
```

```
...
```

```
11 x 8 = 88
```

```
11 x 9 = 99
```

```
11 x 10 = 110
```

```
-----
```

```
Clase-01 $
```

CICLO WHILE

```
In [1]: # El ciclo while se ejecuta mientras la condición sea verdadera
```

```
In [2]: while True:  
    ...:     print("Infinito ...")  
    ...:
```

```
Infinito ...
```

```
Infinito ...
```

```
(Presiona Control + C)
```

```
In [3]: # Lee hasta que sea un decimal
```

```
In [4]: num = ""
```

```
In [5]: while not num.isdecimal():  
    ...:     num = input("Escribe un número: ")  
    ...:
```

```
Escribe un número: a
```

```
Escribe un número: b
```

```
Escribe un número: 12
```

```
In [6]:
```

LEYENDO EDAD - CICLO - WHILE

Nombre del script: lee-edad-ciclos.py

Descripción:

Modifica el script lee-edad.py para que cuando se proporcione una edad incorrecta, se vuelva a leer otro valor, continuando así hasta que se proporcione un valor adecuado de edad, entonces se imprime.

Ejemplo de ejecución:

```
Curso-de-Python/Clase-01 $ python lee-edad-ciclos.py
```

```
Escribe tu edad: 0
```

```
Error: 0 no es un valor adecuado para una edad
```

```
Escribe tu edad: cinco
```

```
Error: cinco no es un valor adecuado para una edad
```

```
Escribe tu edad: 15
```

```
La edad es: 15
```



ACTIVIDAD FINAL

Nombre del script a crear:

nota-de-venta.py

Descripción:

Modificar el script

lista-de-productos-total-ciclos.py para preguntar al usuario si quiere o no el iva desglosado, entonces imprimir la lista de productos y el total con o sin iva desglosado según corresponda.

El usuario puede responder s, S, Si o SI para el caso afirmativo, n, N, No o NO para el caso contrario y cualquier otra respuesta se considera incorrecta por lo que se deberá de volver a preguntar.

Ejemplo de ejecución:

```
Clase-01 $ python nota-de-venta.py
```

```
Desea iva desglosado (Si/No) ? 16
```

```
Error: 16 no es una respuesta válida, intenta de nuevo.
```

```
Desea iva desglosado (Si/No) ? s
```

```
-----  
PRODUCTO | PRECIO | CANT | SUBTOTAL  
-----  
Automóvil | 150000.00 | 1 | 150000.00  
Bicicleta | 13000.00 | 2 | 26000.00  
Chamarra | 3999.99 | 2 | 7999.98  
Laptop Thinkpad (Descuento incluido) | 21750.00 | 1 | 21750.00  
Gafas de realidad virtual Lenovo con sable | 5000.00 | 2 | 10000.00  
-----  
Subtotal | 215749.98  
-----  
IVA | 34520.00  
Total | 250269.98  
-----
```