

Instituto Tecnológico de Celaya

---

**Curso Básico de Python**

---

**Ihtoa.org**

**Objetivo:**

El objetivo del curso básico de python es dar a conocer un lenguaje multiplataforma y multiparadigma de programación moderno y poderoso, proporcionando un nivel de aprendizaje medio el cual le permita desarrollar aplicaciones de buen desempeño.

## 1. Tipos de Datos

```
# Objetivo:
#   El usuario comprendera a usar los diferentes tipos de datos
#   y estructuras de Python

#Enteros
numero = 10
print numero
print numero + 10

#flotantes
noentero = 12.6
print noentero

#decimales muy largos
bigdecimal = 0.1238123891283182038190238012830912803812038129
print "%.47f" % bigdecimal

#booleanos
bool = True
print bool
bool = False
print bool

#Determinando el tipo de dato que es
cadena = "esto es una cadena"
numero = 101
booleano = True

print type(cadena).__name__
print type(numero).__name__
print type(booleano).__name__
```

## 2. Uso de Cadenas

```
# Objetivo:
# El usuario aprendera el uso del tipo nativo conocido como cadenas

#Para imprimir una cadena se usa la siguiente sentencia
print "Hola mundo"

#La lectura de datos se hace usando la función raw_input("cadena a
mostrar")
variable = raw_input("Dame un dato: ")

#Si quisiéramos imprimir el dato leído anteriormente podemos usar varias
formas
#La primera forma es separando las cadenas con una coma
print "el valor que me diste fue: ", variable

#La segunda forma es concatenando cadenas
#Nota: la concatenación de cadenas se hace mediante el operador de suma al
estilo C
print "el valor que me diste fue: " + variable

#La tercera forma es usando un formato parecido al printf de C
print "el valor que me diste fue: %s " % (variable)
#Nota: para imprimir e incluir dentro del texto en este formato depende del
tipo de dato
print "Este es un dato entero: %d" % (10)
print "Este es un dato cadena: %s" % ("python")
print "Este es un dato flotante: %f" % (3.1416)
print "Este es un dato flotante limitando decimales: %0.2f" % (9.123123123)
#Nota: como bien nos dimos cuenta, la conversión de tipos no es automática,
pues al leer
# con raw_input el valor regresado es una cadena
#ejemplo
numero = raw_input("Dame un numero: ")
print numero,10 #concatena
#print numero + 10 #Error, forma incorrecta

#conversión a entero y suma
numero = int(numero)
print "Suma: %d + 10 = %d" % (numero,numero+10)

#De nuevo pedimos una cadena
cadena = raw_input("Cual es tu nombre? ")
#Imprimimos la cadena completa
print "Palabra completa:",cadena
#Todo es un objeto en python, por lo tanto, podemos tratarla como una
estructura
#de datos, por ejemplo, como una tupla de datos
print "Primer letra de tu nombre:",cadena[0]
print "Segunda letra de tu nombre:",cadena[1]
print "Las primeras dos letras:",cadena[:2]
print "De la letra 2 a la 4:",cadena[2:4]
print "Las últimas 2 letras:",cadena[-2:]
print "El último carácter:",cadena[-1]
print "Longitud de la cadena:",len(cadena)
```

## Reto 1

```
#Primer reto del curso basico de python
#Realizar un programa que lea una cadena de n digitos, donde n > 10
#el script mostrara una lista con los requisitos correspondientes:
#1. Tamaño de la cadena leida del teclado
#2. Ultimos 3 digitos y seguidamente los primeros 3 digitos
#3. El digito central
#4. La cadena invertida
#5. Tipo de dato que leimos
```

```
cadena = raw_input("Dame una cadena de longitud n, donde n>10: ")
print cadena
print "1.",cadena.__len__()
print "2.",cadena[-3:],cadena[:3]
print "3. %s" % cadena[cadena.__len__()/2]
print "4.", cadena[::-1]
print "5.", type(cadena).__name__
```

### 3. Estructuras

```
# Objetivo:
# El usuario comprendera a usar los diferentes tipos de datos
# y estructuras de Python

#Estructuras de Datos en Python
#Diccionario:
#La estructura de datos del tipo diccionario es una relacion de uno a uno
#entre claves y valores.

#Sintaxis
diccionario = {"clave":"valor"}

#ejemplo
diccionario = {"manzana":"roja",
               "platano":"amarillo"}

print diccionario
print diccionario["manzana"]
print diccionario["platano"]

#Error, el valor no aplica
print diccionario["roja"]

#Imprimimos las claves
print diccionario.keys()

#Imprimirmos los valores
print diccionario.values()

#alterar un dato
diccionario["manzana"]="verde"
print diccionario["manzana"]

#Añadir elementos al diccionario
diccionario["sandia"] = "roja"
print diccionario

#Eliminar elementos del diccionario
del diccionario["platano"]
print diccionario

#El diccionario de python es similar a la estructura hashtable de java

#Listas:
#Una lista es similar a un arreglo de java, pero me atreveria a decir mas
poderoso,
#es mas bien, algo mas parecido al ArrayList de java

#Sintaxis
lista = ["java","C","C++","VB","python","ruby"]

print lista

#Tratandolo como arreglo de C
print "Elemento 0",lista[0]
print "Elemento 2",lista[2]
```

```

#Longitud de una lista
print "Tamaño lista",lista.__len__()

#Añadir elementos al final de la lista
lista.append("pascal")
print lista

#Añadir elemento en cierta posicion
lista.insert(2,"perl")
print lista

#Unir listas
lista.extend(["basic","asp"])
print lista

#Rangos
print lista[0:5]
print lista[-4:]
print lista[0:5:2]

#Busquedas
print lista.index("python")
print "VB" in lista
print "assamble" in lista

#Eliminar elementos
print lista
lista.remove("asp")
print lista.pop()
print lista

#Tuplas
#Un tupla es una lista inmutable. Una vez creada no puede
#cambiar de ninguna manera

tupla = ("windows","linux","mac")
print tupla

#No se puede agregar, eliminar, o buscar, pero si podemos usar in para
comprobar
#si existe un elemento en la tupla

print "windows" in tupla

```

## Reto 2

```
#!/usr/bin/env python
# -*- coding: UTF-8 -*-
from time import *

#Segundo Reto A del curso Básico de Python
#Realizar un programa que lea una cadena (sin espacios)
#y determine si esta es palindroma o no

cadena = raw_input("Dame una cadena: ");

if(cadena==cadena[::-1]):
    print cadena,"es palindroma"
else:
    print cadena,"no es palindroma"

if("".join(cadena.split())=="".join(cadena.split()[::-1]):
    print cadena,"es palindroma"
else:
    print cadena,"no es palindroma"

#Segundo Reto B del curso Básico de Python
#Realizar un programa que lea la fecha de nacimiento
#del usuario con el siguiente formato "dd/mm/yyy"
#Imprimir si es mayor de edad o no
#Ejemplo
# Fecha de Nacimiento (dd/mm/yyyy): 11/08/1986
# Felicidades, ya eres mayor de edad, tu edad es de 22 años

year = int(strftime("%Y"))
cadena = raw_input("Fecha de Nacimiento (dd/mm/yyyy): ")
edad = year-int(cadena[-4:])

if edad >= 18:
    print "Felicidades, ya eres mayor de edad, tu edad es de",edad,"años"
else:
    print "Creo que no eres mayor de edad"
```

## 4. Sentencia IF

```
#Objetivo:
# El objetivo de este modulo es el aprendizaje de las sentencias
# que python incorpora.

#La sentencia IF
#En python no existen limitadores de bloques como en otros lenguajes
#como por ejemplo los simbolos "{}" muy comunes en java o C
#por lo que la identificacion de bloques en python se lleva a cabo
#con las identaciones, o mejor conocidas como tabulaciones.
#La sentencia IF implica la apertura de un bloque por lo que la sintaxis
#queda de la siguiente forma.

#Sintaxis

if 10>9:
    print "10 es mayor a 9"

#Como en otros lenguajes, la sentencia IF va acompañada de otra sentencia
#llamada else, esta es la sintaxis

if ("micadena"=="tucadena"):
    print "verdadero"
else:
    print "faslo"

#Si requerimos mas de dos comparaciones, podemos usar la sentencia ELIF que
es la
#union de ELSE con IF, es decir, "de lo contrario, si..."

a = 3
if(a==1):
    print "uno"
elif (a==2):
    print "dos"
elif (a==3):
    print "tres"

#Y la sentencia Switch?
#La sentencia no existe en python, la cuestion es debido a
#discusiones en si es necesaria o no, realmente, la sentencia
#switch es poco utilizada por los programadores, por ese motivo
#se preeven alternativas al switch en python.

def caso1():
    print "este es el caso 1"

def case2():
    print "este es el caso2"

casos = {
    "caso1": caso1,
    "caso2": caso2
}

casos["caso1"]()
```

## 5. Sentencia FOR

```
#Objetivo:
# El objetivo de este modulo es el aprendizaje de las sentencias
# que python incorpora.

#Range
#range es una funcion que genera listas con progresiones arimeticas

#sintaxis
print range(5,10)

#La sentencia FOR
#La función FOR es un poco distinta a lo acostumbrado a C o Java,
#en lugar de dar un inicio y un fin de una sucesion de numeros
#hace un recorrido de una lista.

#Sintaxis

for i in range(0,10):
    print i

for i in range(10):
    print i,

lista = ["java","C","C++","VB","python","ruby"]

for ele in lista:
    print ele

diccionario = {"manzana":"roja","platano":"amarillo"}

for clave, valor in diccionario.iteritems():
    print clave,valor

#utilizando algunas instrucciones mas

for i in range(10):
    for x in range(10):
        if i == 3:
            print i,x
        else:
            break
    else:
        print "fin del for"
```

### Reto 3

#Reto 3 del curso básico de python  
#Escribe un programa que pida la anchura y altura de un rectángulo  
#y lo dibuje de la siguiente manera:

```
#Anchura del rectangulo: 5  
#Altura del rectangulo: 5  
#* * * * *  
#* * * * *  
#* * * * *  
#* * * * *  
#* * * * *
```

```
width = input("Anchura del rectangulo: ")  
height = input("Altura del rectangulo: ")
```

```
for w in range(width):  
    for h in range(height):  
        print "*",  
    print ""
```

## 6. Sentencia While

```
#Objetivo:
# El objetivo de este modulo es el aprendizaje de las sentencias
# que python incorpora.

#While
#La sentencia while es mas parecida a las comunes usadas por java o C

#Sintaxis
i=0
while i <= 10:
    print i
    i+=1

#while infinito
while True:
    print "Este ciclo es infinito"
```

### Reto 4

```
#!/usr/bin/env python
# -*- coding: UTF-8 -*-

#Reto 4 del curso básico de python
#El problema 3n + 1
#El problema 3n+1 consiste en el siguiente algoritmo
# 1 leer n
# 2 imprimir n
# 3 si n = 1 entonces detente
# 4 si n es par entonces n = 3n + 1
# 5 de lo contrario n = n/2
# 6 ir al paso 2

n = input("N=")
while True:
    print n,

    if n == 1: break

    if n % 2 == 1: n = 3*n + 1
    else: n = n/2
```

## 7. Funciones

```
#Objetivo:
# El objetivo de este modulo es el aprendizaje del
# uso de las funciones en python.

#La palabra clave def introduce una definicion de funcion.
#Debe ir seguida del nombre de la funcion y la lista entre
#parentesis de los parametros formales.

#sintaxis
def mifuncion():
    print "Esta es una función"

mifuncion()

def funcion_con_parametros(parametro):
    print "El parametro que has enviado es:", parametro

funcion_con_parametros("Este es mi parametro")
funcion_con_parametros(10)
funcion_con_parametros(True)

#Parametros por default
def funcion_con_parametros_default(parametro="parametro por default"):
    print "El parametro que has enviado es:", parametro

funcion_con_parametros_default("ahora si envíe esto")

#Retorno de valor por una funcion
def funcion_con_retorno(tipo=None):
    if tipo == 1:
        return "retorno cadena"
    elif tipo == 2:
        return 10
    elif tipo == 3:
        return True
    else:
        return None

print funcion_con_retorno(1), type(funcion_con_retorno(1))
print funcion_con_retorno(2), type(funcion_con_retorno(2))
print funcion_con_retorno(3), type(funcion_con_retorno(3))

#Retornando mas de un valor
def funcion_retornos_multiples():
    a=1
    b=2
    return a,b

c,d = funcion_retornos_multiples()

print c,d
```

## Reto 5

```
#!/usr/bin/env python
# -*- coding: UTF-8 -*-

#Reto 5 del curso básico de python
#Escribir una función que imprima la serie Fibonacci
#hasta n números (donde n es un parámetro)

#Por ejemplo, la función fibonacci(2000) imprimirá
#1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597

def fibonacci(n=0):
    a,b = 0,1
    while b < n:
        print b,
        a,b = b,a+b

fibonacci(2000)
```

## 8. Excepciones

```
#Objetivo:
# El objetivo de este modulo es el manejo
# de errores llamados tambien excepciones

#Una excepcion es la gestion de un error ocurrido
#durante la ejecucion de nuestro script
#por ejemplo, al sumar una cadena con un entero

x = "10"
#print x+1

#sintaxis

try:
    print x+1
except:
    print "Huuuy, parece que ocurrio un error"

while True:
    try:
        x = int(raw_input("Dame un numero: "))
        break
    except:
        print "Ese no es un numero, intendolo de nuevo."

print "Bien, %d si es un numero" % x
```

## 9. Archivos

```
#Objetivo:
# El objetivo de este modulo es el manejo
# de Archivos (lectura, escritura, modificacion))

#leyendo un archivo
#Sintaxis

file = open('intro.txt','r')

#leemos todo el archivo de un jalo
print file.read()

#leemos una sola linea
print file.readline()

#leemos linea por linea el archivo
for linea in file.readlines():
    print linea,

#cerramos el archivo
file.close()

#abrimos un nuevo archivo como modo escritura
file = open('nuevo.txt','w')
#escribimos una cadena
file.write("mi nuevo archivo")
#cerramos el archivo
file.close()
#volvemos a abrir el archivo pero en modo append, es decir, agregar al
finald el archivo
file = open('nuevo.txt','a')
#escribimos en el archivo
file.write("\n otro")
#cerramos el archivo
file.close()
#lo abrimos de nuevo en modo lectura
file = open('nuevo.txt','r')
#imprimimos todo el archivo de un jalon
print file.read()
```

## Reto 6

```
#!/usr/bin/env python
# -*- coding: UTF-8 -*-

#Reto 6 del curso basico de python
#Escribir un programa que lea una cadena, la cual
#será el nombre de un archivo .txt
#el programa debera ver si ya existe
#si existe, mostrara el contenido
#de lo contrario, lo creara y le pedira
#al usuario escribir los datos que este
#nuevo archivo tendra.

#Ejemplo

Nombre del Archivo: intro.txt
Contenido:
Python es un lenguaje de programación creado por Guido van Rossum en el año
1990....

Nombre del Archivo: nuevo.txt
El archivo aun no existe.
Escribe una línea que se agregara al archivo:

try:
    name = raw_input("Nombre del Archivo: ")
    file = open(name,'r')
    print "Contenido:"
    for line in file.readlines():
        print line
except IOError:
    print "El archivo aun no existe."
    line = raw_input("Escribe una línea que se añadira al archivo: ")
    file = open(name,'w')
    file.write(line)
    file.close()
```

## 10. Programación Orientada a Objetos

```
#Objetivo:
# El objetivo de este modulo es el aprendizaje de la
# programación orientada a objetos en python
#Existen tres conceptos fundamentales en la P00
# encapsulamiento
# herencia
# polimorfismo (No existe)

class animal:
    #constructor
    def __init__(self, nombre, edad=1):
        self.nombre = nombre
        self.edad = edad

    #destructor
    def __del__(self):
        print self.nombre,"murio..."

class perro(animal):
    def ladra(self):
        print "wooooo"

class gato(animal):
    def aulla(self):
        print "miiiiiau"

    def __esta_enfermo(self):
        print "no"

if __name__ == "__main__":
    miperro = perro("Donald",5)
    miperro.ladra()

    del (miperro)

    migata = gato("Sofy",7)
    migata.aulla()
```

### Reto 7

```
#Escribir en forma de programacion orientada a objetos lo siguiente.
#La familia Perez tiene dos hijos, Lalo que es varon y Jessica que es
#mujer, los padres se llaman Miguel y Sofia, los dos hijos tiene
#caracteristicas similares que heredaron de los padres, por ejemplo, ambos
#tiene el cabello de color negro, ojos claros, y nariz respingada, pero
#tambien tiene caracteristiza que solo heredan los del mismo sexo, asi, si
#la madre tiene 5 lunares, la hija tambien, y el padre mide 1.80 el hijo
#tambien. Sin embargo, todos tiene algo distinto, que es la edad: Sofia
#tiene 39 años, Miguel 40, Lalo tiene 22 y Jessica tiene 20. Como
#representarias a la familiar perez en P00.
```