

# Funciones

Clase #08

IIC1103 – Introducción a la Programación

Marcos Sepúlveda ([marcos@ing.puc.cl](mailto:marcos@ing.puc.cl))

# Veremos hoy ...

---

- ▶ Librerías
- ▶ Parámetros
- ▶ Parámetros por defecto
- ▶ Ejercicios
- ▶ Ámbito (*scope*) de las variables
  - Locales
  - Globales
- ▶ Ejercicios

# Recordemos

```
import math
```

Importar módulo

```
def bienvenida():  
    print("Calculo de distancia")
```

Función sin retorno

```
def distancia(x1, y1, x2, y2):  
    dx = x2 - x1  
    dy = y2 - y1  
    dalcuadrado = math.pow(dx,2) + math.pow(dy,2)  
    resultado = math.sqrt(dalcuadrado)  
    return resultado
```

Llamada a  
funciones  
importadas

Función  
retorna  
resultado

```
#utilizando la funcion  
bienvenida()  
valor = distancia(7,5,4,1)  
print(valor)
```

Llamada a  
funciones

# ¿Qué son los algoritmos numéricos?

- ▶ Permiten resolver problemas matemáticos, obteniendo aproximaciones para las soluciones mediante algoritmos iterativos.
- ▶ Ejemplos:
  - Evaluación de funciones como log, sqrt, cos, ...
  - Interpolación y extrapolación
  - Sistemas de ecuaciones
  - Integración numérica
  - Generación de números aleatorios
  - Minimización y maximización de funciones
  - Aplicaciones espectrales
  - Análisis estadístico
  - Ecuaciones diferenciales
- ▶ Permiten resolver problemas cuando no hay buenas soluciones algebraicas

# Cálculo de $e^x$ mediante series

- La serie empleada es:

$$e^x = 1 + \frac{x^1}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \dots$$

- Criterio de detención:
  - Cuando la diferencia entre el valor obtenido entre dos iteraciones sucesivas no sea mayor que una precisión dada por el usuario.

# Cálculo de $e^x$ mediante series

```
import sys

def Factorial(numero):
    factorial = 1
    i = 1
    while i <= numero:
        factorial *= i
        i += 1
    return factorial

def E_X(x, precision):
    y = 1 # Valor para e**x, inicialmente 1
    y_anterior = 0 # Valor para e**x en iteración anterior
    i = 0 # Contador de iteraciones

    while (y - y_anterior) >= precision:
        i += 1
        y_anterior = y
        y = y + (x ** i) / Factorial(i)

    print("Calculo de e(x) con precision:", precision)
    print(y)
    print("Se emplearon", i, "iteraciones\n")

    return y
```

# Cálculo de $e^x$ mediante series

```
print("Calculo de e ** x")

print("Ingrese el valor de x:")
x = float(sys.stdin.readline())

print("Ingrese la precisión requerida (ej: 1e-5):")
precision = float(sys.stdin.readline())

e_x = E_X(x, precision)

print("Valor calculado de e **", x, "es", e_x)
```

```
>>>
Calculo de e ** x
Ingrese el valor de x:
1
Ingrese la precisión requerida (ej: 1e-5):
1e-10
Calculo de e(x) con precision: 1e-10
2.71828182845823
Se emplearon 14 iteraciones

Valor calculado de e ** 1.0 es 2.71828182845823
```

# Librerías – motivación

- ▶ Vimos como implementar cálculo numérico de  $e^x$  en base a series.
- ▶ Podríamos usar series parecidas para calcular otras funciones numéricas.
- ▶ Sin embargo, muchas de estas funciones ya están predefinidas en librerías conocidas como módulos de Python.
- ▶ Ejemplo de cálculo de  $e^x$ :

```
import sys
import math
print("Calculo de e ** x")
print("Ingrese el valor de x:")
x = float(sys.stdin.readline())
print("Valor calculado de e **", x, "es", math.exp(x))
```



# Uso de *import*

- ▶ ¿Para qué nos sirve *import*?
  - Nos permite traer a nuestro código funciones previamente creadas en Python o traer nuestras propias funciones que tenemos definidas en otros archivos.
- ▶ Algunos ejemplos de módulos (o librerías) disponibles en Python
  - *math*
  - *random*
  - *sys*



# Módulos presentes en Python

---

## ► ***math***

- Contiene funciones y constantes matemáticas.
- <https://docs.python.org/3/library/math.html>

## ► ***random***

- Contiene funciones para producir números aleatorios.
- <https://docs.python.org/3/library/random.html>

## ► ***datetime***

- Provee tipos de datos para manipular fechas y horas.
- <https://docs.python.org/3/library/datetime.html>

## ► Y hay más:

- <https://docs.python.org/3/library/index.html>

# Ventajas de usar módulo

---

- ▶ Funciones y variables definidas sólo una vez, y luego pueden ser utilizadas en muchos programas sin necesidad de reescribir el código.
- ▶ Permiten que un programa pueda ser organizado en varias secciones lógicas, escritas cada una en un archivo separado.
- ▶ Hacen más fácil compartir componentes con otros programadores.

# Formas de usar módulos

## ► *from*

```
from math import exp, cos, pi
from random import randint, randrange
```

```
print(exp(5.5))
print(cos(pi / 2))
```

```
# entero aleatorio entre 1 y 3
print(randint(1,3))
#entero aleatorio en rango [10,20) y en intervalos de 2
print(randrange(10, 20, 2))
```

## ► *import*

```
import math
import random
```

```
print(math.exp(5.5))
print(math.cos(math.pi / 2))
```

```
# entero aleatorio entre 1 y 3
print(random.randint(1,3))
#entero aleatorio en rango [10,20) y en intervalos de 2
print(random.randrange(10, 20, 2))
```

```
>>>
```

```
244.69193226422038
```

```
6.123233995736766e-17
```

```
2
```

```
14
```

# Algunos elementos del módulo *math*

Módulo.elemento	Retorna	Ejemplo
<code>math.exp(x)</code>	$e^{**}x$	<pre>&gt;&gt;&gt; math.exp(5) 148.4131591025766</pre>
<code>math.log2(x)</code>	Logaritmo en base 2 de x	<pre>&gt;&gt;&gt; math.log2(16) 4.0</pre>
<code>math.pow(x, y)</code>	Eleva x a y	<pre>&gt;&gt;&gt; math.pow(2, 3) 8.0</pre>
<code>math.sqrt(x)</code>	Raíz cuadrada	<pre>&gt;&gt;&gt; math.sqrt(25) 5.0</pre>
<code>math.factorial(x)</code>	Factorial	<pre>&gt;&gt;&gt; math.factorial(5) 120</pre>
<code>math.pi</code>	Constante pi	<pre>&gt;&gt;&gt; math.pi 3.141592653589793</pre>
<code>math.e</code>	Constante e	<pre>&gt;&gt;&gt; math.e 2.718281828459045</pre>

# ¿Cómo crear mis propios módulo?

- ▶ Crear un archivo con extensión **.py** con el código necesario
  - El nombre del archivo indica cuál es el nombre del módulo.
- ▶ **Ejemplo:** crear módulo que permita convertir unidades de medidas de longitud: millas a kilómetros, pies a metros, centímetros a pulgadas, y viceversa.

Fragmento módulo **conversor** en archivo **conversor.py**

```
kms_por_milla = 1.609344

def millas_a_kms(mi) :
    return mi * kms_por_milla
```

Fragmento archivo **principal.py**

```
x = float(input("Ingrese millas: "))
print(x, "millas equivale a", conversor.millas_a_kms(x), "km")
```

# Módulo *conversor* – archivo *conversor.py*

```
kms_por_milla = 1.609344
pies_por_metro = 3.2808399
cms_por_pulgada = 2.54

def millas_a_kms(mi):
    return mi * kms_por_milla

def kms_a_millas(km):
    return km / kms_por_milla

def pies_a_mts(pies):
    return pies / pies_por_metro

def mts_a_pies(mt):
    return mt * pies_por_metro

def pulgadas_a_cms(p):
    return p * cms_por_pulgada

def cms_a_pulgadas(cm):
    return cm / cms_por_pulgada
```

# Programa principal – archivo *principal.py*

```
import conversor

def Menu():
    print("-----")
    print("¿Qué conversión desea hacer?")
    print("1) millas a kilómetros")
    print("2) kilometros a millas")
    print("3) pies a metros")
    print("4) metros a pies")
    print("5) pulgadas a centímetros")
    print("6) centímetros a pulgadas")
    print("0) salir")

opcion = 99

while opcion != 0:
    Menu()
    opcion = int(input("ingrese opción: "))
    if opcion==1:
        x = float(input("Ingrese millas: "))
        print(x, "millas equivale a", conversor.millas_a_kms(x), "km")
    elif opcion==2:
        x = float(input("Ingrese kilómetros: "))
        print(x, "kilómetros equivale a", conversor.kms_a_millas(x), "millas")
    elif opcion==3:
        x = float(input("Ingrese pies: "))
        print(x, "pies equivale a", conversor.pies_a_mts(x), "metros")
    elif opcion==4:
        x = float(input("Ingrese metros: "))
        print(x, "metros equivale a", conversor.mts_a_pies(x), "pies")
    elif opcion==5:
        x = float(input("Ingrese pulgadas: "))
        print(x, "pulgadas equivale a", conversor.pulgadas_a_cms(x), "centímetros")
    elif opcion==6:
        x = float(input("Ingrese centímetros: "))
        print(x, "centímetros equivale a", conversor.cms_a_pulgadas(x), "pulgadas")
    elif opcion==0:
        print("Adiós")
    else:
        print(opcion, "no es una opción válida; intente de nuevo.")
```



# Parámetros por defecto

- ▶ Cuando se llama a una función que tiene parámetros, ¿siempre hay que enviar datos en todos los parámetros?
- ▶ **No, ... se pueden definir valores por defecto.**
- ▶ Ejemplo:

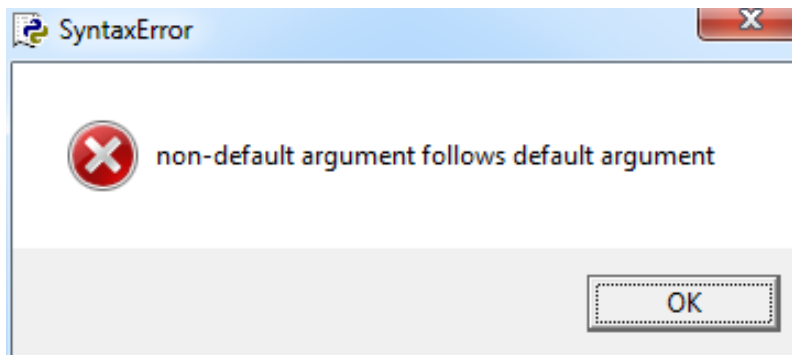
```
def Incrementar(n, incr=1):  
    return n + incr
```

```
>>> Incrementar(10,15)  
25  
>>> Incrementar(10)  
11  
>>> Incrementar(  
n, incr=1)
```

# Parámetros por defecto

- ▶ A partir de un parámetro por defecto, todos los demás deben tener valores por defecto.

```
def f(a=2, b):  
    return a+b
```



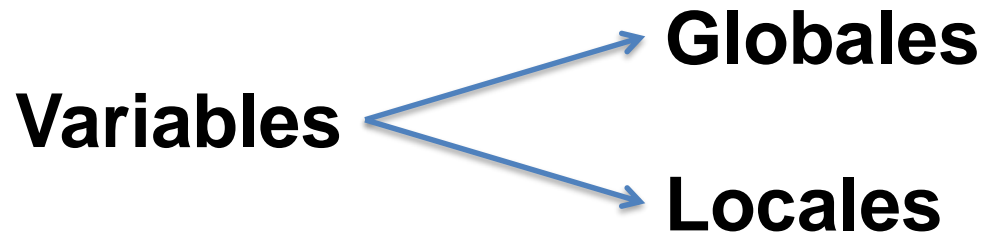
# Flujo de ejecución de un programa

---

- ▶ La ejecución siempre empieza con la primera instrucción del programa.
- ▶ Las instrucciones se ejecutan una a una, desde arriba hacia abajo.
- ▶ Si es la definición de una función, ésta se crea y queda disponible para ser llamada.
- ▶ Las llamadas a función son como un desvío en el flujo de ejecución del programa.

# Ámbito de variables

- ▶ El ámbito de una variable es el espacio donde está variable es visible y puede ser utilizada.
- ▶ Hay dos tipos de variables



- ▶ **Variables globales:** residen fuera de toda función.
- ▶ **Variables locales:** aquellas que son creadas dentro de una función.
  - Tienen una vida efímera. Solo existen durante la llamada a la función, y sólo son visibles dentro de ella. En el momento en el que ésta concluye, desaparecen.

# Ámbito de variables – ejemplo

este es un mundo nuevo, que incluye todo lo del mundo global y todo lo creado dentro del cuerpo de la función

Todos ven var1

mundo global

```
var1 = 1
```

```
def funcion(param1, ...):
```

```
# cuerpo de  
# la funcion  
var2 = 2
```

Sólo la función,  
que se llama  
*funcion*, puede  
ver var2

```
var3 = 3
```

Todos ven var3

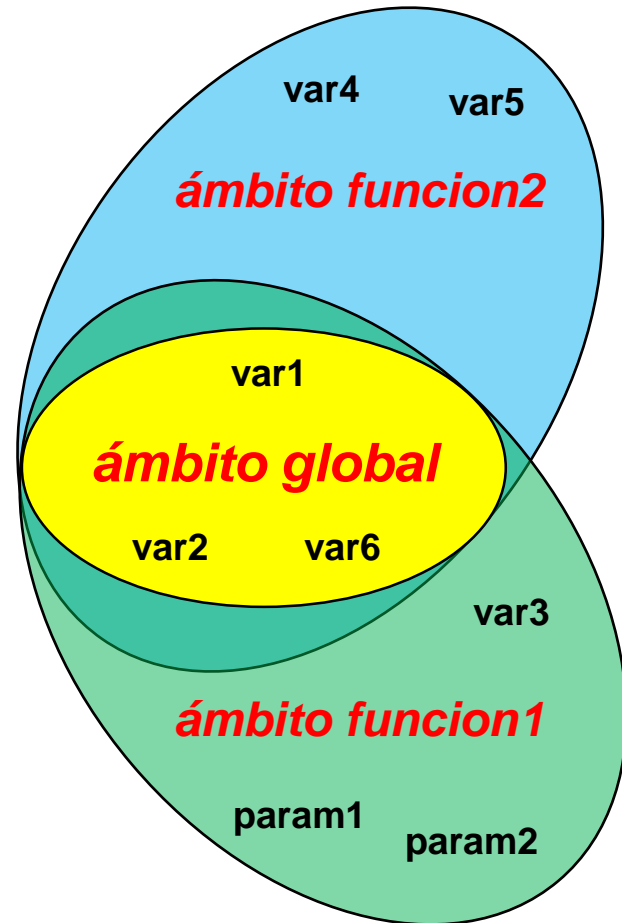
# Ámbito de variables – ejemplo

```
var1 = 1
var2 = 2

def funcion1(param1, param2):
    var3 = 3
    print(param1, param2, var3)

def funcion2():
    var4 = 4
    var5 = 5
    print(var1, var2, var4, var5, var6)

var6 = 6
funcion1(10, 20)
funcion2()
```



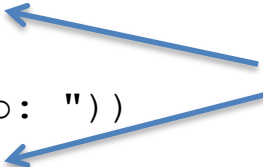
# Ejemplo de variable local

```
import math
```

```
def AreaRectangulo():  
    largo = float(input("Ingrese largo: "))  
    ancho = float(input("Ingrese ancho: "))  
    area = largo * ancho
```

```
def AreaCirculo():  
    radio = float(input("Ingrese radio: "))  
    area = math.pi * radio ** 2
```

El problema es que **area** no  
está en el ámbito **global**



```
print("Este programa calcula el área de una rectángulo o un círculo")
```

```
respuesta = input("¿Es un rectángulo? (si/no): ")
```

```
if respuesta == "si":
```

```
    AreaRectangulo()
```

```
else:
```

```
    AreaCirculo()
```

```
print("El área es:", area)
```

**¡ no funciona !**

# Ejemplo de variable local – solución usando *global*

```
import math
```

```
def AreaRectangulo():  
    global area  
    largo = float(input("Ingrese largo: "))  
    ancho = float(input("Ingrese ancho: "))  
    area = largo * ancho
```

```
def AreaCirculo():  
    global area  
    radio = float(input("Ingrese radio: "))  
    area = math.pi * radio ** 2
```

```
print("Este programa calcula el área de una rectángulo o un círculo")
```

```
respuesta = input("¿Es un rectángulo? (si/no): ")
```

```
if respuesta == "si":  
    AreaRectangulo()  
else:  
    AreaCirculo()
```

```
print("El área es:", area)
```

***global*** indica que queremos usar una variable que está definida en el ámbito global; podemos usar o modificar su valor

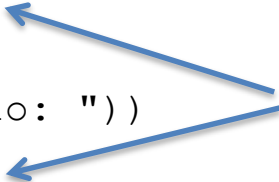


# Ejemplo de variable local – solución retornando valores

```
import math
```

```
def AreaRectangulo():  
    largo = float(input("Ingrese largo: "))  
    ancho = float(input("Ingrese ancho: "))  
    area = largo * ancho  
    return area
```

```
def AreaCirculo():  
    radio = float(input("Ingrese radio: "))  
    area = math.pi * radio ** 2  
    return area
```

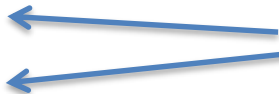


Las funciones retornan el valor obtenido

```
print("Este programa calcula el área de una rectángulo o un círculo")
```

```
respuesta = input("¿Es un rectángulo? (si/no): ")
```

```
if respuesta == "si":  
    area = AreaRectangulo()  
else:  
    area = AreaCirculo()
```



Se debe guardar el valor resultante en una variable

```
print("El área es:", area)
```

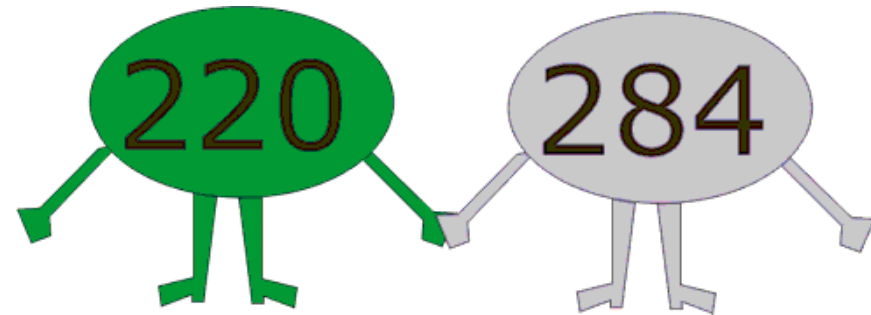
# EJERCICIOS

# Ejercicio 1 – números amigos

- ▶ Dos números amigos son dos enteros positivos distintos, tal que la suma de los divisores propios de uno es igual al otro, y viceversa.

- ▶ Ejemplo:

- Divisores propios de 220: 1,2,4,5,10,11,20,22,44,55,110
- Suma = 284
- Divisores propios de 284: 1,2,4,71,142
- Suma= 220



- ▶ Escribir función ***SumaDivisores(x)*** que entregue la suma de los divisores de ***x***.
- ▶ Escribir función ***SomosAmigos(x,y)*** que retorne ***True/False*** si dos números son amigos o no.

# Ejercicio 1 – números amigos – solución

```
def SumaDivisores(x):  
    suma = 0  
    i = 1  
    while i < x:  
        if x % i == 0:  
            suma += i  
        i += 1  
    return suma
```

```
def SomosAmigos(x, y):  
    sumaDivX = SumaDivisores(x)  
    sumaDivY = SumaDivisores(y)  
    return sumaDivX == y and sumaDivY == x
```

```
x = 1  
while x < 10000:  
    sumaDivX = SumaDivisores(x)  
    if sumaDivX > x and SomosAmigos(x, sumaDivX):  
        print(x, sumaDivX, " son números amigos")  
    x += 1
```

```
>>>  
220 284 son números amigos  
1184 1210 son números amigos  
2620 2924 son números amigos  
5020 5564 son números amigos  
6232 6368 son números amigos
```

## Ejercicio 2 – cobro taxi

- ▶ Estás en el centro de Santiago y necesitas ir de inmediato a una reunión fuera de la ciudad. Lo primero que ves es un taxi, lo haces parar y te subes, pero no sabes cuánto te va a salir el cobro. Con miedo, le preguntas al chofer cuánto te saldría, y él te responde que un alumno de la UC ha implementado un sistema que entrega el precio exacto de cuánto será el cobro a destino. Tu misión es implementar el mismo programa en Python.
- ▶ El cobro por defecto es de \$250 por kilometro en el área de Santiago. En el caso de ir a Valparaíso, el cobro es de \$500 por kilometro, y en el caso de ir a Rancagua es de \$320 por kilometro.
- ▶ Distancia a Valparaíso = 120 km
- ▶ Distancia a Rancagua = 65 km



## Ejercicio 2 – cobro taxi – solución

```
def Cobro(distancia, destino="Santiago"):
    if destino=="Santiago":
        tarifa = 250*distancia
    elif destino=="Valparaíso":
        tarifa = 500*distancia
    elif destino=="Rancagua":
        tarifa = 320*distancia
    return tarifa

destino=input("Hola, ¿dónde viaja? (Santiago/Valparaíso/Rancagua) ")

if destino == "Santiago":
    distancia=int(input("¿A cuántos kilómetros queda? "))
    cobro = Cobro(distancia)
elif destino == "Valparaíso":
    cobro = Cobro(120,destino)
elif destino == "Rancagua":
    cobro = Cobro(65,destino)
else:
    cobro = 0

print("Cobro: $" + str(cobro))
```

## Ejercicio 3 – conversión de 24 horas a AM/PM

---

- ▶ Hay dos formatos típicos en los cuales se especifica una hora:
  - 24 horas – horas van desde las 00:00 (medianoche) hasta las 23:59; y
  - 12 horas – horas van desde las 12:00 AM (medianoche) a las 12:59 AM, desde la 1:00 AM a las 11:59 AM, desde las 12:00 PM (mediodía) a las 12:59 PM, y desde la 1:00 PM a las 11:59 PM.
- ▶ Por ejemplo, las 21:53 (en formato de 24 horas) corresponde a las 9:53 PM y las 00:27 corresponde a las 12:27 AM.
- ▶ Escriba una función ***MostrarHora(hora, minutos)*** que recibe del usuario dos valores de tipo entero, correspondientes a la hora y los minutos en formato de 24 horas, y muestra en pantalla la misma hora pero en formato de 12 horas.

## Ejercicio 3 – conversión de 24 horas a AM/PM

- Para probarlo, ejecute el siguiente código:

```
def MostrarHora(hora, minutos):  
    # Implemente aquí el código
```

```
MostrarHora(0,0)  
MostrarHora(0,59)  
MostrarHora(1,0)  
MostrarHora(11,59)  
MostrarHora(12,0)  
MostrarHora(12,1)  
MostrarHora(12,59)  
MostrarHora(13,0)  
MostrarHora(13,1)  
MostrarHora(23,59)
```

```
>>>  
0:0 == 12:0 AM  
0:59 == 12:59 AM  
1:0 == 1:0 AM  
11:59 == 11:59 AM  
12:0 == 12:0 PM  
12:1 == 12:1 PM  
12:59 == 12:59 PM  
13:0 == 1:0 PM  
13:1 == 1:1 PM  
23:59 == 11:59 PM
```



# Ejercicio 3 – conversión de 24 horas a AM/PM

## Algoritmo:

- ▶ Para la primera hora del día (de 0:00 a 0:59), suma 12 horas y añade "AM"
  - Ejemplos: 0:10 = 12:10 AM, 0:40 = 12:40 AM
- ▶ De 1:00 a 11:59, sólo añade "AM"
  - Ejemplos: 1:15 = 1:15 AM, 11:25 = 11:25 AM
- ▶ De 12:00 a 12:59, sólo añade "PM"
  - Ejemplos: 12:10 = 12:10 PM, 12:55 = 12:55 PM
- ▶ De 13:00 a 23:59, resta 12 horas y añade "PM"
  - Ejemplos: 14:55 = 2:55 PM, 23:30 = 11:30 PM

# Ejercicio 3 – conversión de 24 horas a AM/PM – solución

## ► En base al algoritmo:

```
def MostrarHora(hora, minutos):
    if (hora == 0):
        horaM = 12
        extM = "AM"
    elif (hora < 12):
        horaM = hora
        extM = "AM"
    elif (hora == 12):
        horaM = hora
        extM = "PM"
    else:
        horaM = hora - 12
        extM = "PM"

    print(hora, ":", minutos, " == ",
          horaM, ":", minutos, " ", extM, sep='')

```

## ► Otra alternativa:

```
def MostrarHora(hora, minutos):
    if (hora == 0):
        horaM = 12
    elif (hora <= 12):
        horaM = hora
    else:
        horaM = hora - 12

    if (hora < 12):
        extM = "AM"
    else:
        extM = "PM"

```

## Ejercicio 4 – cálculo inmobiliario

- ▶ Una empresa dedicada a la compra y venta de Edificios ha solicitado de su ayuda. Para estos fines le ha entregado el siguiente texto informativo.
- ▶ Un edificio está caracterizado por el número de pisos que tiene y los metros cuadrados por piso. Los edificios, además, pueden estar en dos tipos de sectores: residencial o comercial. La tasación que tiene el edificio parte con un precio base para todos de \$100.000.000, y se le agrega un valor adicional dependiendo de sus características, como indica la tabla:

Características	Valor Adicional
Nº de Pisos entre 5 y 10 (ambos inclusive)	\$10.000.000
Nº de Pisos entre 11 y 20 (ambos inclusive)	\$20.000.000
Nº de Pisos mayor a 20	\$50.000.000
Sector Residencial	\$60.000.000
Sector Comercial	\$110.000.000
Total Metros Cuadrados mayor a 3000 y menor o igual que 5000	\$45.000.000
Total Metros Cuadrados mayor a 5000	\$67.000.000

- ▶ Dos edificios pueden ser comparados de distintas formas: comparando el nº de pisos, comparando el total de metros cuadrados, y comparando las tasaciones.
- ▶ La empresa le ha solicitado que implemente el módulo **edificio.py**, que contenga funciones para desplegar información sobre un edificio y compararlo con otro.

## Ejercicio 4 – cálculo inmobiliario

- El módulo ***edificio.py*** debería contener la definición de las siguientes funciones:

```
# Mostrar edificio en pantalla
def MostrarEdificios(nombre, pisos, m2, sector)

# Calcula metros cuadrados totales de un edificio
def Metros2(pisos, m2)

# Calcula tasación de un edificio
def Tasacion(pisos, m2, sector)

# Compara dos edificios
def ComparaEdificios(nombre1, pisos1, m2_1, sector1,
                      nombre2, pisos2, m2_2, sector2)
```

## Ejercicio 4 – cálculo inmobiliario

- Y podría ser llamado de la siguiente manera:

```
import edificio
```

```
e1_nombre = "Alto Cordillera"  
e1_pisos = 30  
e1_m2 = 120  
e1_sector = "residencial"
```

```
e2_nombre = "Espacio VIP"  
e2_pisos = 20  
e2_m2 = 180  
e2_sector = "comercial"
```

```
edificio.MostrarEdificios(e1_nombre, e1_pisos, e1_m2, e1_sector)  
edificio.MostrarEdificios(e2_nombre, e2_pisos, e2_m2, e2_sector)
```

```
edificio.ComparaEdificios(e1_nombre, e1_pisos, e1_m2, e1_sector,  
                           e2_nombre, e2_pisos, e2_m2, e2_sector)
```

## Ejercicio 4 – cálculo inmobiliario

- Arrojando el siguiente resultado:

```
>>>
-----
Edificio Alto Cordillera
- número de pisos: 30
- metros cuadrados por piso: 120
- metros cuadrados totales: 3600
- sector: residencial
- tasación: 255000000
-----
-----
Edificio Espacio VIP
- número de pisos: 20
- metros cuadrados por piso: 180
- metros cuadrados totales: 3600
- sector: comercial
- tasación: 275000000
-----
Edificio 1: Alto Cordillera
Edificio 2: Espacio VIP
Alto Cordillera tiene más pisos que Espacio VIP
Ambos edificios tienen el mismo número de metros cuadrados
Espacio VIP tiene mejor tasación que Alto Cordillera
```

## Ejercicio 4 – cálculo inmobiliario – solución

---

```
# Mostrar edificio en pantalla
def MostrarEdificios(nombre, pisos, m2, sector):
    print("-----")
    print("Edificio", nombre)
    print("- número de pisos:", pisos)
    print("- metros cuadrados por piso:", m2)
    print("- metros cuadrados totales:", Metros2(pisos, m2))
    print("- sector:", sector)
    print("- tasación:", Tasacion(pisos, m2, sector))
    print("-----")

# Calcula metros cuadrados totales de un edificio
def Metros2(pisos, m2):
    return pisos * m2
```

# Ejercicio 4 – cálculo inmobiliario – solución

---

```
# Calcula tasación de un edificio
def Tasacion(pisos, m2, sector):
    tasacion = 100000000

    if (pisos >= 5) and (pisos <=10):
        tasacion += 10000000
    elif (pisos >=11) and (pisos <=20):
        tasacion += 20000000
    elif (pisos >20):
        tasacion += 50000000

    if sector == "comercial":
        tasacion += 110000000
    elif sector == "residencial":
        tasacion += 60000000

    if Metros2(pisos, m2) > 3000 and Metros2(pisos, m2) <= 5000:
        tasacion += 45000000
    elif Metros2(pisos, m2) > 5000:
        tasacion += 67000000

    return tasacion
```



# Ejercicio 4 – cálculo inmobiliario – solución

```
# Compara dos edificios
def ComparaEdificios(nombre1, pisos1, m2_1, sector1,
                      nombre2, pisos2, m2_2, sector2):
    print("Edificio 1:", nombre1)
    print("Edificio 2:", nombre2)

    if (pisos1 > pisos2):
        print(nombre1, "tiene más pisos que", nombre2)
    elif (pisos1 == pisos2):
        print("Ambos edificios tienen el mismo número de pisos")
    else:
        print(nombre2, "tiene más pisos que", nombre1)

    if Metros2(pisos1, m2_1) > Metros2(pisos2, m2_2):
        print(nombre1, "tiene más metros cuadrados que", nombre2)
    elif Metros2(pisos1, m2_1) == Metros2(pisos2, m2_2):
        print("Ambos edificios tienen el mismo número de metros cuadrados")
    else:
        print(nombre2, "tiene más metros cuadrados que", nombre1)

    if Tasacion(pisos1, m2_1, sector1) > Tasacion(pisos2, m2_2, sector2):
        print(nombre1, "tiene mejor tasación que", nombre2)
    elif Tasacion(pisos1, m2_1, sector1) == Tasacion(pisos2, m2_2, sector2):
        print("Ambos edificios tienen igual tasación")
    else:
        print(nombre2, "tiene mejor tasación que", nombre1)
```