Programación I

Profesor:

Lic. Ricardo Thompson

Lenguaje Python Versión 3 (3.x.y)

- Python oficial de python.org
- Anaconda Python
- WinPython
- Python(x,y)
- ActivePython
- Pydroid 3 (Android)

¿Qué es Python?

- Es un lenguaje de muy alto nivel.
- Es multiparadigma.
- Es interpretado: No se compila.
- El espíritu de Python ("zen") privilegia la legibilidad del código.
- Es portable.
- Es sensible a mayúsculas y minúsculas.

Estructura de los programas en Python

- No tienen un formato rígido.
- No hay llaves ni puntos y coma.
- Las instrucciones terminan con Enter.
- Los comentarios comienzan con el signo numeral: #

Salida por pantalla: print()

```
print('Hola Mundo')
print("Hola de nuevo")

print("Hola Mundo", end=" ")
```


Variables

- No se declaran.
- Se crean automáticamente con la asignación de un valor inicial.
- Intentar usar una variable no inicializada provoca un error.

Variables

Reglas para crear nombres de variables:

- **Sólo letras, números y el guión bajo.**
- **No pueden comenzar con un número.**
- **No pueden coincidir con las palabras reservadas del lenguaje.**

Variables

- **Los nombres de variables deben tener sentido.**
- **Deben evitarse variables llamadas "l" *(ele minúscula)* u "o" *(letra o)* porque pueden confundirse fácilmente con números.**

Variables

- Los caracteres se consideran cadenas de longitud 1.
- Hay valores booleanos: True y False.
- Se permite la asignación múltiple: a,b,c = 3, "Lunes", 5.18

Variables

- Para imprimir variables se las separa de las constantes con una coma:

```
dia = 5
print("Hoy es", dia)
```

Variables

- **Puede lograrse mayor control de la salida impresa con el operador %:**

  print("Precio: %5.2f" %precio)

- **No va coma luego de cerrar comillas.**

Variables

- **Los especificadores de conversión son los mismos que en Lenguaje C:**

  %d para números enteros
  %f para números reales
  %c para caracteres

Variables

- Si hay más de una variable, éstas deben encerrarse entre paréntesis:

```
print("X = %4d – Y = %4d" %(x,y))
```


Variables

- Para ingresar valores por teclado se utiliza la función input( ):

```
a = input("Mensaje")
```

- input( ) <u>siempre</u> devuelve un string.

Variables

- Existen funciones para convertir este string a otros tipos de dato:

```
n = int(input("Mensaje"))
r = float(input("Mensaje"))
```


Operadores aritméticos

+ Suma	- Resta
* Multiplicación	/ División real
// División entera	% Módulo o resto
** Potenciación	

Orden de evaluación

1. Potenciación
2. Menos unario
3. Multiplicación, división y módulo
4. Suma y resta

Asignación extendida

a += 1 ⇔ a = a + 1

a -= 2 ⇔ a = a - 2

a *= 3 ⇔ a = a * 3

a /= 4 ⇔ a = a / 4

a **= 5 ⇔ a = a ** 5

No hay operadores incrementales

Estructuras de Control

Estructura Alternativa

Formato 1

```
if <condición>:
    . . . . . .
    . . . . . .
    . . . . . .
```

Estructura Alternativa

Formato 2

```
if <condición>:
    . . . . . .
else:
    . . . . . .
```


Estructura Alternativa

Formato 3

```
if <condición>:
    . . . . . .
elif <condición>:
    . . . . . .
else:
    . . . . . .
```

Estructura Alternativa

- Las condiciones y el else van seguidas del carácter “dos puntos”.
- La sangría o indentación es lo que establece el alcance del if.
- Python recomienda una sangría standard de 4 espacios, sin tabs.
- La sangría debe ser uniforme.

Operadores relacionales

== igual

> mayor

< menor

>= mayor o igual

<= menor o igual

!= distinto

Operadores lógicos

and

or

not

Condiciones encadenadas

if a<b>c:

equivale a

if a<b and b>c:

Operador condicional

<var>= <valor1> if <condición> else <valor2>

Ejemplo:

```
a = b if b>=0 else -b
```


Selección múltiple

No existe en Python

Ejemplo 1

Ingresar un número entero e imprimir el nombre del mes correspondiente (1: Enero, 2: Febrero, 3: Marzo...)


```
mes = int(input("Mes ? "))
if mes == 1:
    print("Enero")
elif mes == 2:
    print("Febrero")
elif mes == 3:
    print("Marzo")
[. . . . .]
elif mes == 12:
    print("Diciembre")
else:
    print("Mes inválido")
```

Estructura Iterativa

Instrucción while

while *<condición>*:

. . . . . .

. . . . . .

. . . . . .

Estructura Iterativa

- **La condición va seguida del carácter "dos puntos".**
- **La sangría o indentación es lo que establece el alcance del ciclo.**
- **Python recomienda una sangría standard de 4 espacios, sin tabs.**
- **La sangría debe ser uniforme.**

Estructura Iterativa

- **Instrucción break**
- **Cláusula else:**

Ejemplo 2

Uso de while, break y else:

Leer un número entero e imprimir un mensaje indicando si se trata de un número primo o no.

```
n = int(input("Ingrese un número "))
i = 2
while i<n:
    if n % i == 0:
        print(n,"no es un número primo")
        break
    i = i + 1
else:
    print(n,"es un número primo")
```


Estructura Iterativa

- Instrucción continue

Ejemplo 3

Uso de while y continue

Imprimir los valores de 1/x entre -5 y 5


```
x = -6
while x<5:
    x = x + 1
    if x == 0:
        continue
    print("%2d: %5.2f" %(x,1/x))
```

Estructura Iterativa

Instrucción for

for <*variable*> in <*secuencia*>:

. . . . . .

. . . . . .

. . . . . .

Estructura Iterativa

- **for se utiliza para recorrer una secuencia.**
- **Esta secuencia puede ser cualquier *iterable*: Un rango, una lista, una cadena, un archivo, una tupla o un diccionario.**

Función range( )

- Genera una secuencia de números enteros.
- Admite tres formas de utilización.

Función range( )

Formato 1:

range(<vfinal>)

- Genera una secuencia de números enteros entre 0 y <vfinal>.
- <vfinal> no está incluido.

Función range( )

Formato 2:

range(<vinicial>, <vfinal>)

- **Genera una secuencia de enteros entre <vinicial> y <vfinal>.**
- **<vfinal> no está incluido.**

Función range( )

Formato 3:

range(<vinicial>, <vfinal>, <inc>)

- **Genera una secuencia de enteros entre <vinicial> y <vfinal> con incremento <inc>.**
- **<vfinal> no está incluido.**

Función range( )

Formato 3:

range(<vinicial>, <vfinal>, <inc>)

- **El incremento puede ser negativo.**
- **Las instrucciones break y continue hacen lo mismo que en while.**

Ejemplo 4

Uso de for y range()

Imprimir los números impares entre 1 y N

```
n = int(input("Ingrese un número: "))
for i in range(1, n+1, 2):
    print(i, end=" ")
print()
```


Ejercitación

- Buscar en Internet el *Zen de Python* y analizarlo.
- Escribir un programa para resolver el antiguo acertijo chino que se describe a continuación: *"He contado 35 cabezas y 94 patas entre las gallinas y los conejos de mi granja. Cuántos conejos y cuántas gallinas tengo?"* (Sugerencia: Verificar todas las combinaciones posibles hasta hallar la correcta).