UNIVERSIDAD TECNOLOGICA NACIONAL

Capítulo 3: Control de Flujo

Programación 2

Bloque IF

Indentación

- Importante que siempre exista.
- Un nuevo nivel de indentación debe estar siempre que haya dos puntos (:)

```
una_variable = 5

if una_variable > 10:

    print("Valor mayor a 10")

if una_variable:

    print("La variable no debe ser 0")

# Else

if una_variable >= 0:

    print("El valor es positivo")

else:

    print("El valor es negativo")
```

Bloque IF: múltiples condiciones

```
if una_variable % 2 == 0:
    print("El valor es par")
elif una_variable < 0:
    print("El valor es impar y negativo")
elif una_variable > 100:
    print("El valor es impar y mayor a 100")
else:
    print("El valor no cumple las condiciones")
```

Operador ternario

```
edad = 30

mayor_de_edad = "Si" if edad >= 18 else "No"


# Equivalente en C/Java/Net a mayor_de_edad = edad >= 18 ? "Si" : "No"
```

Bucle FOR

```
list(range(4))    # => [0, 1, 2, 3]

for i in range(4):
    print(i)
# 0
# 1
# 2
# 3

for animal in ["perro", "gato", "raton"]:
    print(f"{animal} es un mamifero")
# perro es un mamifero
# gato es un mamifero
# raton es un mamifero
```

Bucle FOR: función enumerate

```
nombres = ["Juan", "Pedro", "Maria"]


for indice, nombre in enumerate(nombres):  # Enumerate agrega indices
    print(f"{indice} - {nombre}")

# 0 - Juan

# 1 - Pedro

# 2 - Maria
```

Bucle FOR: función zip

```
edades = [60, 15, 84]
for nombre, edad in zip(nombres, edades):  # Zip combina listas
    print(f"{nombre} tiene {edad} años")
# Juan tiene 60 años
# Pedro tiene 15 años
# Maria tiene 84 años


# Zip y Enumerate pueden combinarse
for indice, (nombre, edad) in enumerate(zip(nombres, edades)):
    print(f"{indice} - {nombre} tiene {edad} años")
# 0 - Juan tiene 60 años
# 1 - Pedro tiene 15 años
# 2 - Maria tiene 84 años
```

Bucle FOR: iteraciones sobre diccionarios

```
alumnos = {"Juan": 60, "Pedro": 15, "Maria": 84}


for nombre, edad in alumnos.items():
    print(f"{nombre} tiene {edad} años")
# Juan tiene 60 años
# Pedro tiene 15 años
# Maria tiene 84 años
```

Bucle FOR: iteraciones sobre diccionarios

```
# Nombre: [Edad, Nota]

alumnos = {"Juan": [60, 7.5], "Pedro": [15, 4.1], "Maria": [84, 9.5]}


for nombre, (edad, nota) in alumnos.items():

    if nota >= 6:

        print(f"{nombre} aprobó con {nota} puntos, teniendo {edad} años")

# Juan aprobó con 7.5 puntos, teniendo 60 años

# Maria aprobó con 9.5 puntos, teniendo 84 años
```

Bucle FOR: continue, break y else

```
buscado = "Pedro"
for nombre, (_, nota) in alumnos.items():
    if nombre == buscado:
        print(f"{nombre} obtuvo {nota} puntos")
        break
# Pedro obtuvo 4.1 puntos


buscado = "Martín"
for nombre, (_, nota) in alumnos.items():
    if nombre == buscado:
        print(f"{nombre} obtuvo {nota} puntos")
        break
else:
    print("No existe ese alumno")
# No existe ese alumno
```

Bucle FOR: continue, break y else

```
precios = [6.43, 7.94, 9.23, 7.97, 4.84, 9.71, 6.52, 8.94, 8.62, 9.72]

for indice, precio in enumerate(precios):

    if precio <= 8.0:

        continue



    precios[indice] *= 0.8



precios  # => [6.43, 7.94, 7.38, 7.97, 4.84, 7.77, 6.52, 7.15, 6.9 ,
7.78]
```

Bucle WHILE

El bucle while opera identicamente al bucle for con break, continue y else

```
# Bucle While tradicional

x = 0

while x < 4:

    print(x)

    x += 1

# Bucle Do-While

x = 0

while True:

    print(x)

    x += 1

    if x < 4:

        break
```

Excepciones: try, except, else, finally

```
try:
    a = 1 / 0
except ZeroDivisionError as exception:
    print(f"Ha ocurrido un error | {exception}")
# Ha ocurrido un error | division by zero
try:
    a = 1 / 0
except ZeroDivisionError as exception:
    print(f"Ha ocurrido un error | {exception}")
finally:
    print("Proceso finalizado")
# Ha ocurrido un error | division by zero
# Proceso finalizado
```

Excepciones: try, except, else, finally

```
try:
    lista = [1, 2, 3]
    lista[3]
except IndexError as exception:
    print(f"No pueden utilizarse índices fuera del rango | {exception}")
try:
    a = 1 / 0
except ZeroDivisionError as exception:
    print(f"No puede dividirse por cero | {exception}")
try:
    notas = {"Juan": 2, "María": 3}
    notas["Alejandro"]<
except KeyError as exception:
    print(f"Sólo pueden usarse claves definidas | {exception}")
try:
    print(hola)
except NameError as exception:
    print(f"Sólo pueden usarse variables definidas | {exception}")
```

Excepciones: try, except, else, finally

```
try:
    a, b = [1, 2, 3]
except ValueError as exception:
    print(f"Valores provistos no coinciden con experados | {exception}")
```