

Fundamentos de Programación

TEMA: ESTRUCTURA DE CONTROL 1

Semana 05

OBJETIVO DEL LABORATORIO

Aplica las condicionales simples en un programa de computadora.

MARCO TEÓRICO

Entendiendo el antecedente y consecuente, se podrán diseñar algoritmos que respondan ante determinada situación que se nos plantee.

El uso de las condiciones implica, que podamos contemplar las diversas opciones que tengamos en un solo algoritmo.

Como fuente de información, se tiene la misma plataforma del Netacad. Tanto para realizar los ejemplos propuestos en la interfaz integrada, así como para repasar lo visto en clase.

RECURSOS

a. Hardware

- Pc Pentium IV a superior
- Conexión de red

b. Software

- Sistema Operativo Windows XP a superior
- Navegador Chrome o Firefox
- Edube Sandbox de Python desde Netacad

PROCEDIMIENTO

1. SENTENCIAS CONDICIONALES: **if ...**

La estructura de control **if ...** permite que un programa ejecute unas instrucciones cuando se cumplan una condición. En inglés "if" significa "si" (condición). La orden en Python se escribe así:

```
if condición:  
    aquí van las órdenes que se ejecutan si la condición es cierta  
    y que pueden ocupar varias líneas
```

La primera línea contiene la condición a evaluar y es una expresión lógica. Esta línea debe terminar siempre por dos puntos (:).

A continuación, viene el bloque de órdenes que se ejecutan cuando la condición se cumple (es decir, cuando la condición es verdadera). Es importante señalar que este bloque debe ir sangrado, puesto que Python utiliza el sangrado para reconocer las líneas que forman un bloque de instrucciones. El sangrado que se suele utilizar en Python es de cuatro espacios, pero se pueden utilizar más o menos espacios. Al escribir dos puntos (:) al final de una línea, IDLE sangrará automáticamente las líneas siguientes. Para terminar un bloque, basta con volver al principio de la línea.



Veamos un ejemplo. El programa siguiente pide un número positivo al usuario y almacena la respuesta en la variable "numero". Después comprueba si el número es negativo. Si lo es, el programa avisa que no era eso lo que se había pedido. Finalmente, el programa imprime siempre el valor introducido por el usuario. A continuación, se pueden ver dos ejecuciones paso a paso de ese programa. En la primera el usuario escribe un valor negativo y en la segunda el usuario escribe un valor positivo:

Ejemplo de `if ... 1`

```
numero = int(input("Escriba un número positivo: "))
if numero < 0:
    print(";Le he dicho que escriba un número positivo!")
print("Ha escrito el número", numero)
```

RESULTADO:

```
Escriba un número positivo: -5
;Le he dicho que escriba un número positivo!
Ha escrito el número -5
```

Ejemplo de `if ... 2`

```
numero = int(input("Escriba un número positivo: "))
if numero < 0:
    print(";Le he dicho que escriba un número positivo!")
print("Ha escrito el número", numero)
```

RESULTADO:

```
Escriba un número positivo: 5
Ha escrito el número 5
```

2. BIFURCACIONES: `if ... else ...`

La estructura de control `if ... else ...` permite que un programa ejecute unas instrucciones cuando se cumple una condición y otras instrucciones cuando no se cumple esa condición. En inglés "if" significa "si" (condición) y "else" significa "si no". La orden en Python se escribe así:

```
if condición:
    aquí van las órdenes que se ejecutan si la condición es cierta
    y que pueden ocupar varias líneas
else:
    y aquí van las órdenes que se ejecutan si la condición es
    falsa y que también pueden ocupar varias líneas
```



La primera línea contiene la condición a evaluar. Esta línea debe terminar siempre por dos puntos (:).

A continuación, viene el bloque de órdenes que se ejecutan cuando la condición se cumple (es decir, cuando la condición es verdadera). Es importante señalar que este bloque debe ir sangrado, puesto que Python utiliza el sangrado para reconocer las líneas que forman un bloque de instrucciones. El sangrado que se suele utilizar en Python es de cuatro espacios, pero se pueden utilizar más o menos espacios. Al escribir dos puntos (:) al final de una línea, IDLE sangrará automáticamente las líneas siguientes. Para terminar un bloque, basta con volver al principio de la línea.

Después viene la línea con la orden **else**, que indica a Python que el bloque que viene a continuación se tiene que ejecutar cuando la condición no se cumpla (es decir, cuando sea falsa). Esta línea también debe terminar siempre por dos puntos (:). La línea con la orden **else** no debe incluir nada más que el **else** y los dos puntos.

En último lugar está el bloque de instrucciones sangrado que corresponde al **else**.

Veamos un ejemplo. El programa siguiente pregunta la edad al usuario y almacena la respuesta en la variable "edad". Después comprueba si la edad es inferior a 18 años. Si esta comparación es cierta, el programa escribe que es menor de edad y si es falsa escribe que es mayor de edad. Finalmente, el programa siempre se despide, ya que la última instrucción está fuera de cualquier bloque y por tanto se ejecuta siempre. A continuación, se pueden ver dos ejecuciones paso a paso de ese programa:

Ejemplo de **if ... else ... 1**

```
edad = int(input("¿Cuántos años tiene? "))
if edad < 18:
    print("Es usted menor de edad")
else:
    print("Es usted mayor de edad")
print("¡Hasta la próxima!")
```

RESULTADO:

```
¿Cuántos años tiene? 17
Es usted menor de edad
¡Hasta la próxima!
```

Ejemplo de **if ... else ... 2**

```
edad = int(input("¿Cuántos años tiene? "))
if edad < 18:
    print("Es usted menor de edad")
else:
    print("Es usted mayor de edad")
print("¡Hasta la próxima!")
```



RESULTADO:

```
¿Cuántos años tiene? 25
Es usted mayor de edad
¿Hasta la próxima!
```

Aunque no es aconsejable, en vez de un bloque `if ... else ...` se podría escribir un programa con dos bloques `if ...`

```
edad = int(input("¿Cuántos años tiene? "))
if edad < 18:
    print("Es usted menor de edad")
if edad >= 18:
    print("Es usted mayor de edad")
print("¿Hasta la próxima!")
```

Es mejor no hacerlo así por dos motivos:

- al poner dos bloques `if` estamos obligando a Python a evaluar siempre las dos condiciones, mientras que en un bloque `if ... else ...` sólo se evalúa una condición. En un programa sencillo la diferencia no es apreciable, pero en programas que ejecutan muchas comparaciones, el impacto puede ser apreciable.
- utilizando `else` nos ahorramos escribir una condición (además, escribiendo la condición nos podemos equivocar, pero escribiendo `else` no).

3. IDENTACIÓN (SANGRADO) DE LOS BLOQUES

Un bloque de instrucciones puede contener varias instrucciones. Todas las instrucciones del bloque deben tener el mismo sangrado:

```
edad = int(input("¿Cuántos años tiene? "))
if edad < 18:
    print("Es usted menor de edad")
    print("Recuerde que está en la edad de aprender")
else:
    print("Es usted mayor de edad")
    print("Recuerde que debe seguir aprendiendo")
print("¿Hasta la próxima!")
```

Se aconseja utilizar siempre el mismo número de espacios en el sangrado, aunque Python permite que cada bloque tenga un número distinto. El siguiente programa es correcto:

```
edad = int(input("¿Cuántos años tiene? "))
if edad < 18:
    print("Es usted menor de edad")
    print("Recuerde que está en la edad de aprender")
else:
    print("Es usted mayor de edad")
```



```
print("Recuerde que debe seguir aprendiendo")
print(";Hasta la próxima!")
```

Lo que no se permite es que en un mismo bloque haya instrucciones con distintos sangrados. Dependiendo del orden de los sangrados, el mensaje de error al intentar ejecutar el programa será diferente:

- En este primer caso, la primera instrucción determina el sangrado de ese bloque, por lo que, al encontrar la segunda instrucción, con un sangrado mayor, se produce el error "unexpected indent" (sangrado inesperado).

```
edad = int(input(";Cuántos años tiene? "))
if edad < 18:
    print("Es usted menor de edad")
    print("Recuerde que está en la edad de aprender")
else:
    print("Es usted mayor de edad")
    print("Recuerde que debe seguir aprendiendo")
print(";Hasta la próxima!")
```

```
File ".main.py", line 4
    print("Recuerde que está en la edad de aprender")
    ^
IndentationError: unexpected indent
```

- En este segundo caso, la primera instrucción determina el sangrado de ese bloque, por lo que, al encontrar la segunda instrucción, con un sangrado menor, Python entiende que esa instrucción pertenece a otro bloque, pero como no hay ningún bloque con ese nivel de sangrado, se produce el error "unindent does not match any outer indentation level" (el sangrado no coincide con el de ningún nivel superior).

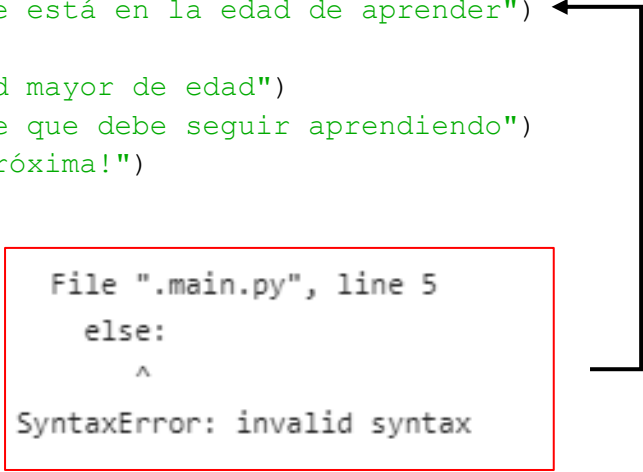
```
edad = int(input(";Cuántos años tiene? "))
if edad < 18:
    print("Es usted menor de edad")
    print("Recuerde que está en la edad de aprender")
else:
    print("Es usted mayor de edad")
    print("Recuerde que debe seguir aprendiendo")
print(";Hasta la próxima!")
```

```
File ".main.py", line 4
    print("Recuerde que está en la edad de aprender")
    ^
IndentationError: unindent does not match any outer indentation level
```



- En este tercer caso, como la segunda instrucción no tiene sangrado, Python entiende que la bifurcación `if` ha terminado, por lo que al encontrar un `else` sin su `if` correspondiente se produce el error "invalid syntax" (sintaxis no válida).

```
edad = int(input("¿Cuántos años tiene? "))
if edad < 18:
    print("Es usted menor de edad")
print("Recuerde que está en la edad de aprender")
else:
    print("Es usted mayor de edad")
    print("Recuerde que debe seguir aprendiendo")
print("¡Hasta la próxima!")
```



```
File ".main.py", line 5
    else:
      ^
SyntaxError: invalid syntax
```

4. LAUNCH IN EDUBE

- usando la instrucción `if-else` para bifurcar la ruta de control;
- construyendo un programa completo que resuelva problemas simples de la vida real.

Érase una vez una tierra: una tierra de leche y miel, habitada por gente feliz y próspera. La gente pagaba impuestos, por supuesto, su felicidad tenía límites. El impuesto más importante, denominado Impuesto sobre la renta personal (PIT, por sus siglas en inglés) tuvo que pagarse una vez al año, y se evaluó utilizando la siguiente regla:

- si el ingreso del ciudadano no era superior a 85,528 soles, el impuesto era igual al 18% del ingreso menos 556 soles y 2 céntimos (esta fue la llamada reducción de impuestos);
- si el ingreso era superior a este monto, el impuesto era igual a 14,839 soles y 2 céntimo, más el 32% del excedente sobre 85,528 soles.

Su tarea es escribir una simple "calculadora de impuestos"; debe aceptar un valor de punto flotante: el ingreso. A continuación, se debe imprimir el impuesto calculado, redondeado a soles completos. Hay una función llamada `round` que hará el redondeo por usted, la encontrará en el código de esqueleto a continuación.

Nota: este país feliz nunca devuelve dinero a sus ciudadanos. Si el impuesto calculado es menor que cero, solo significa que no hay impuesto (el impuesto es igual a cero). Toma esto en consideración durante tus cálculos.

Mire el código a continuación: solo lee un valor de entrada y genera un resultado, por lo que debe completar con algunos cálculos inteligentes.



Pruebe su código utilizando los datos que hemos proporcionado.

```
# Sample Solution

ingreso = float(input("Ingrese el ingreso anual: "))
if ingreso < 85528:
    impuesto = ingreso * 0.18 - 556.02
else:
    impuesto = (ingreso - 85528) * 0.32 + 14839.02
if impuesto < 0.0:
    impuesto = 0.0
impuesto = round(impuesto,0)
print("El impuesto es:", impuesto)
```

CONCLUSIONES Y RECOMENDACIONES DE LA EXPERIENCIA

- Al momento de construir la condición, hay que verificar los intervalos para saber hasta qué rango considera; para evitar un error de lectura.
- Se recomienda usar el Edube SandBox para realizar nuestros ejercicios sin necesidad de instalar algún software adicional.
- Así como en algún momento se solicita cálculos con números enteros, sería factible buscar realizar lo mismo; pero con números reales.

ACTIVIDAD VIRTUAL

1. Observa y analiza el siguiente video: https://www.youtube.com/watch?v=TMearPvj_rA, luego responde las siguientes preguntas:
 - Realiza ese algoritmo considerando la "Entrada" "Proceso" y "Salida" (Desarrollalo en Python)

