

Programación

intermédio I

1

Variables

variables

Una parte de la memoria que
tiene un valor.

variables

tipoVariable **nombre**

ej: **ent** edad

variables

podemos agregar un valor

```
ent edad
```

```
edad = 15
```

variables

podemos iniciar con un valor

```
ent edad = 15
```

variables

podemos iniciar con un valor

```
ent dos = 1 + 1
```

```
ent siete = 10 - 3
```

```
ent cuatro = 2 * 2
```

```
ent diez = 40 / 4
```

variables

```
ent un = 5 % 2
```

$$5 / 2 = 2,5$$

Pero esperamos una variable entera

o sea

$$5/2 = 2 \text{ con resto } 1$$

el simbolo de % significa resto

variables

calcula la edad al año siguiente

```
ent edadAnoSiguiente, edad;  
edad = 10  
edadAnoSiguiente = edad + 1;
```

un pequeño programa

variables

```
ent edadAnoSeguiente, edad
```

```
edad = 20
```

```
edadAnoSeguiente = edad + 1
```

```
print edadAnoSeguiente
```

PRINT = imprimir
en una pantalla

un pequeño programa en Java

variables

```
class SumaEdad {  
  
    public static void main(String[] args) {  
  
        ent edade = 20;  
  
        System.out.println(edade);  
  
        ent edadAnoSeguiente;  
  
        edadAnoSeguiente = edad + 1;  
  
        System.out.println(edadAnoSeguiente);  
  
    }  
  
}
```

un pequeño programa en Java

variables

```
class SumaEdad {
```

```
    public static void main(String[] args) {
```

```
        ent edade = 20;
```

```
        System.out.println(edade);
```

```
        ent edadAnoSeguiente;
```

```
        edadAnoSeguiente = edad + 1;
```

```
        System.out.println(edadAnoSeguiente);
```

```
    }
```

```
}
```

PRINT = imprimir
en una pantalla

Comando necesario
para iniciar un
programa en JAVA

JAVA necesita
un punto coma

un pequeño programa en Java

variables

```
class SumaEdad {  
    public static void main(String[] args) {  
        ent edade = 20;  
  
        System.out.println(edade);  
  
        ent edadAnoSeguiente;  
  
        edadAnoSeguiente = edad + 1;  
  
        System.out.println(edadAnoSeguiente);  
    }  
}
```

The diagram illustrates the scope of variables in the provided Java code. A red dashed line connects the opening curly brace of the `class SumaEdad` to the closing curly brace of the class, indicating the scope of the class-level variables. An orange dashed line connects the opening curly brace of the `main` method to the closing curly brace of the method, indicating the scope of the local variables. The variables `edade` and `edadAnoSeguiente` are highlighted in bold in the original image.

variables

un pequeño programa en Python

```
if __name__ == '__main__':  
  
    edad= 20  
  
    print(edad)  
  
    edadAnoSeguiente = edad + 1  
  
    print(edadAnoSeguiente)
```

variables

un pequeño programa en Python

```
if __name__ == '__main__':
```

```
    edad= 20
```

```
    print(edad)
```

```
    edadAnoSeguiente = edad + 1
```

```
    print(edadAnoSeguiente)
```

PRINT = imprimir
en una pantalla

Comando necesario
para iniciar un
programa en PYTHON

variables

Otros tipos de variables

```
double pi = 3.14
```

```
boolean menorDeEdade = edade < 18
```

```
char letra = 'a'
```

```
string nombre = 'Juan Pablo'
```


variables

Otros tipos de variables

```
double pi = 3.14
```

```
boolean menorDeEdade = edade < 18
```

```
char letra = 'a'
```

```
string nombre = 'Juan Pablo'
```

2

Operadores - parte 1 - < > <= >=

OPERADORES

> Mayor que

< Menor que

> = Mayor o igual

< = Menor o igual

= igual

!= diferente

OPERADORES

```
ent edad = 22
```

```
if edad > 18
```

```
    print "Mayor"
```

```
end_if
```

Mayor

```
ent edad = 15
```

```
if edad < 18
```

```
    print "Menor"
```

```
end_if
```

Menor

OPERADORES

```
ent edad = 18
```

```
if edad >= 18
```

```
    print "Puede conducir"
```

```
end_if
```

Puede conducir

```
ent edad = 17
```

```
if edad <= 17
```

```
    print "No puede conducir"
```

```
end_if
```

No puede conducir

OPERADORES

```
ent edad = 18
```

```
if edad = 18
```

```
    print "Puede conducir"
```

```
end_if
```

Puede conducir

```
ent edad = 10
```

```
if edad != 17
```

```
    print "No puede conducir"
```

```
end_if
```

No puede conducir

OPERADORES

```
ent edad = 19

ent multas = 10

if edad >= 18

    print "Puede conducir"

end_if
```

Acá tenemos un problema, necesitamos validar más cosas.

O sea no es suficiente tener más de 18 años, no se puede tener multas también.

3

Operadores - parte 2 - && || = !=

OPERADORES

&& y

|| o

= igual

&& - (y)

Piense en una situación como:

Tengo que tener más de 18 años y nunca haber tenido multas para tener una licencia de conducir.

Si no tengo alguna de las partes no puedo sacar la licencia de conducir

OPERADORES

```
ent edad = 19
```

```
ent multas = 0
```

```
if edad >= 18 && multas < 1
```

```
    print "Puede conducir"
```

```
end_if
```

→ Puede conducir

```
ent edad = 25
```

```
ent multas = 8
```

```
if edad >= 18 && multas < 1
```

```
    print "Puede conducir"
```

```
end_if
```



→ NO puede conducir

&&

Si una parte de la ecuación es falsa, no necesitamos ni leer el otro lado.
El operador AND, para funcionar, necesita que las dos partes sean VERDADERAS.


OPERADORES

```
ent edad = 18  
  
ent multas = 9  
  
if edad >= 18 && multas < 1  
    print "Puede conducir"  
end_if
```



if  edad >= 18  multas < 1

print "Puede conducir"

end_if




```
ent edad = 23  
  
ent multas = 0  
  
if edad >= 18 && multas < 1  
    print "Puede conducir"  
end_if
```

if  edad >= 18  multas < 1

print "Puede conducir"

end_if



|| - (y)

Piense en una situación como:

Hoy voy a trabajar con micro o auto.

Si no tengo alguna de las partes no importa. Basta 1 ser verdadera que yo consigo llegar a mi trabajo

OPERADORES

```
ent edad = 30
```

```
ent multas = 4
```

```
if edad < 18 || multas > 0
```

```
    print "No puede conducir"
```

```
end_if
```

→ NO puede conducir

```
ent edad = 12
```

```
ent multas = 0
```

```
if edad < 18 || multas > 0
```

```
    print "No puede conducir"
```

```
end_if
```

→ NO puede conducir

||

Si una parte de la ecuación es falsa, no importa.
El operador OR, para funcionar necesita que sólo una parte sea VERDADERA.

OPERADORES

```
ent edad = 23
```

```
ent multas = 9
```

```
if edad <= 18 || multas < 1
```

```
    print "NO puede conducir"
```

```
end_if
```

```
ent edad = 43
```


```
ent multas = 9
```

```
if edad <= 18 || multas < 1
```

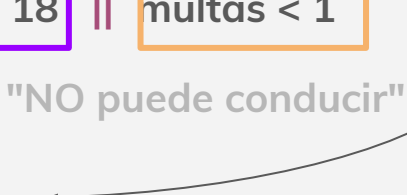


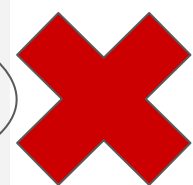
```
    print "NO puede conducir"
```

```
end_if
```

```
if edad <= 18 || multas < 1
    print "NO puede conducir"
end_if
```



```
if edad <= 18 || multas < 1
    print "NO puede conducir"
end_if
```



OPERADORES

```
ent edad = 19  
  
ent multas = 10  
  
if edad >= 18  
    print "Puede conducir"  
end_if
```

Acá tenemos un problema.
Falta una parte.
O que hacer si la edad es menor que 18??

4

IF y ELSE

IF Y ELSE

```
ent edad = 18
```

```
if edad >= 18
```

```
    print "Puede conducir"
```

```
end_if
```

```
ent edad = 18
```

```
if edad >= 18
```

```
    print "Puede conducir"
```

```
else
```

```
    print "NO puede conducir"
```

```
end_if
```

→ Puede conducir

ELSE

si la línea del **IF** es falsa el programa ignora y busca la otra opción que es el **ELSE**, que significa SI NO

IF Y ELSE

```
ent edad = 18  
  
if edad >= 18  
    print "Puede conducir"  
  
else  
    print "NO puede conducir"
```

→ Puede conducir

```
ent edad = 14  
  
if edad >= 18 ✖  
    print "Puede conducir"  
  
else  
    print "NO puede conducir"
```

→ NO puede conducir

5

WHILE

WHILE

```
ent edade = 15  
while (edad < 18)  
    print (edad)  
    edad = edad + 1
```

15
16
17

```
ent i = 0  
while (i < 10)  
    print(i)  
    i = i + 1
```

0
1
2
3
4
5
6
7
8
9

WHILE

es un bucle que repite un fragmento de código mientras la condición sea verdadera.

```
ent edade = 15
```

```
while (edad < 18)
```

```
    print (edad)
```

```
    edad = edad + 1
```

WHILE

```
ent edade = 15
```

```
while (15 < 18)
```

```
    print (15)
```

```
    edad = edad + 1
```

16

```
while (16 < 18)
```

```
    print (16)
```

```
    edad = edad + 1
```

17

```
while (17 < 18)
```

```
    print (17)
```

```
    edad = edad + 1
```

18

```
while (18 < 18) ❌
```

```
    print (edad)
```

```
    edad = edad + 1
```

6

FOR

FOR

```
for (int a = 0; a < 4; a = a + 1)  
    print ("Hola")
```

Hola
Hola
Hola

```
ent a = 0  
while (a < 4)  
  
    print ("Hola")  
  
    a = a + 1
```

FOR

La idea es la misma del while: hacer un fragmento de código ser repetido mientras una condición sigue siendo verdadera.

Pero además usted puede inicializar variables y el modificador de ellas.


```
for (int a = 0; a < 4; a = a + 1)
    print ("Hola")
```

FOR

```
for (int a = 0; a < 4; a = a + 1)
    print ("Hola")
```

a = 1

Hola

```
for (int a = 0; a < 4; a = a + 1)
    print ("Hola")
```

a = 2

Hola

```
for (int a = 0; a < 4; a = a + 1)
    print ("Hola")
```

a = 3

Hola

```
for (int a = 0; a < 4; a = a + 1)
    print ("Hola")
```

a = 4



7

Bloques

BLOQUES

```
while (a < 4)

    print ("Hola")

    a = a + a

end_while
```

```
if edad >= 18 {

    print "Puede conducir"

else

    print "NO puede conducir"

}
```

En algunos lenguajes de programación utilizamos claves para abrir y cerrar bloques.

BLOQUES

while (condición)

for (int i = 0; i < 10; i++)

if (xxxxx > a)

end_if

end_for

end_while

while (condición)

for (int i = 0; i < 10; i++)

if (xxxxx > a)

end_if

end_for

end_while

Podemos tener bloques dentro de otros bloques.

GRACIAS
