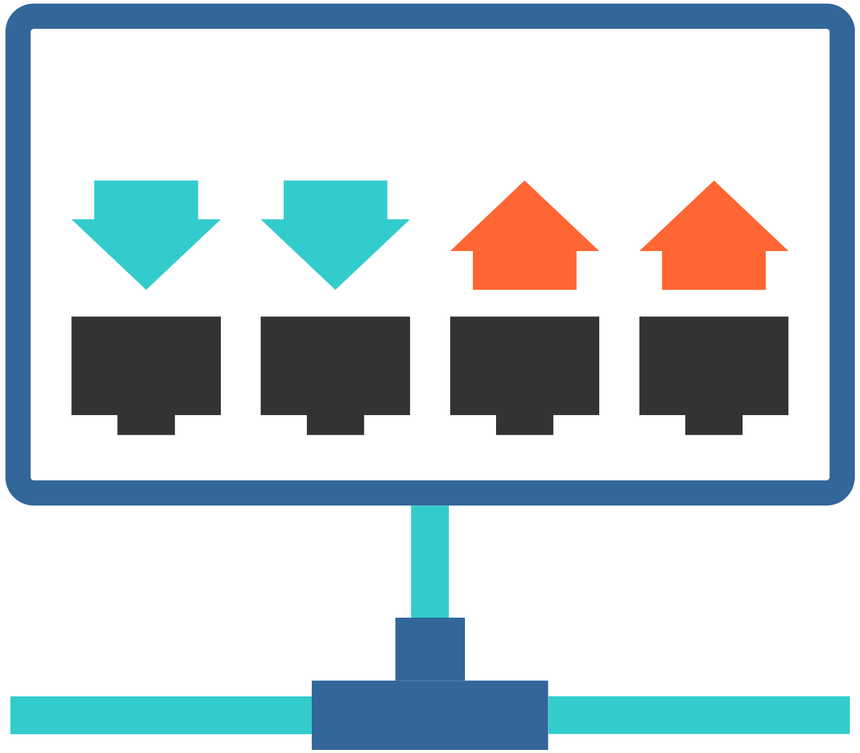
Ministerio de Educación
Gobierno de Chile

UNIVERSIDAD CATÓLICA DE TEMUCO

SENTENCIAS BÁSICAS DE PYTHON



Módulo 3: **Instalación y explotación de software de aplicaciones productivas**

Conectividad y Redes

Objetivos de Aprendizaje de la Especialidad

Módulo 1	**OA1** Leer y utilizar técnicamente proyectos de conectividad y redes, considerando planos o diagramas de una red de área local (red LAN), basándose en los modelos TCP/IP y OSI. **OA3** Instalar y mantener cableados estructurados, incluyendo fibra óptica, utilizados en la construcción de redes, basándose en las especificaciones técnicas correspondientes. **OA7** Instalar y configurar una red inalámbrica según tecnologías y protocolos establecidos.
Módulo 2	**OA2** Instalar y configurar sistemas operativos en computadores personales con el fin de incorporarlos a una red LAN, cumpliendo con los estándares de calidad y seguridad establecidos. **OA11** Armar y configurar un equipo personal, basándose en manuales de instalación, utilizando las herramientas apropiadas y respetando las normas de seguridad establecidos.
Módulo 3	**OA8** Aplicar herramientas de software que permitan obtener servicios de intranet e internet de manera eficiente.
Módulo 4	**OA4** Realizar pruebas de conexión y señales en equipos y redes, optimizando el rendimiento de la red y utilizando instrumentos de medición y certificación de calidad de la señal, considerando las especificaciones técnicas.
Módulo 5	**OA5** Aplicar métodos de seguridad informática para mitigar amenazas en una red LAN, aplicando técnicas como filtrado de tráfico, listas de control de acceso u otras.
Módulo 6	**OA9** Mantener y actualizar el hardware de los computadores personales y de comunicación, basándose en un cronograma de trabajo, de adcuerdo a las especificaciones técnicas del equipo.
Módulo 7	**OA10** Mantener actualizado el software de productividad y programas utilitarios en un equipo personal, de acuerdo a los requerimientos de los usuarios.
Módulo 8	**OA6** Aplicar procedimientos de recuperación de fallas y realizar copias de respaldo de los servidores, manteniendo la integridad de la información.
Módulo 9	No esta asociado a Objetivos de Aprendizaje de la Especialidad (OAE), sino a Genéricos. No obstante, puede asociarse a un OAE como estrategia didáctica.

Perfil de Egreso – Objetivos de Aprendizaje Genéricos

A- Comunicarse oralmente y por escrito con claridad, utilizando registros de habla y de escritura pertinentes a la situación laboral y a la relación con los interlocutores.	**B-** Leer y utilizar distintos tipos de textos relacionados con el trabajo, tales como especificaciones técnicas, normativas diversas, legislación laboral, así como noticias y artículos que enriquezcan su experiencia laboral.	**C-** Realizar las tareas de manera prolija, cumpliendo plazos establecidos y estándares de calidad, y buscando alternativas y soluciones cuando se presentan problemas pertinentes a las funciones desempeñadas.
D- Trabajar eficazmente en equipo, coordinando acciones con otros in situ o a distancia, solicitando y prestando cooperación para el buen cumplimiento de sus tareas habituales o emergentes.	**E-** Tratar con respeto a subordinados, superiores, colegas, clientes, personas con discapacidades, sin hacer distinciones de género, de clase social, de etnias u otras.	**F-** Respetar y solicitar respeto de deberes y derechos laborales establecidos, así como de aquellas normas culturales internas de la organización que influyen positivamente en el sentido de pertenencia y en la motivación laboral.
G- Participar en diversas situaciones de aprendizaje, formales e informales, y calificarse para desarrollar mejor su trabajo actual o bien para asumir nuevas tareas o puestos de trabajo, en una perspectiva de formación permanente.	**H-** Manejar tecnologías de la información y comunicación para obtener y procesar información pertinente al trabajo, así como para comunicar resultados, instrucciones e ideas.	**I-** Utilizar eficientemente los insumos para los procesos productivos y disponer cuidadosamente los desechos, en una perspectiva de eficiencia energética y cuidado ambiental.
J- Emprender iniciativas útiles en los lugares de trabajo y/o proyectos propios, aplicando principios básicos de gestión financiera y administración para generarles viabilidad.	**K-** Prevenir situaciones de riesgo y enfermedades ocupasionales, evaluando las condiciones del entorno del trabajo y utilizando los elementos de protección personal según la normativa correspondiente.	**L-** Tomar decisiones financieras bien informadas, con proyección a mediano y largo plazo, respecto del ahorro, especialmente del ahorro previsional, de los seguros, y de los riesgos y oportunidades del endeudamiento crediticio así como de la inversión.

Marco de Cualificaciones Técnico Profesional (MCTP) Nivel 3 y su relación con los OAG

HABILIDADES

1. Información

1.Analiza y utiliza información de acuerdo a parámetros establecidos para responder a las necesidades propias de sus actividades y funciones.
2.Identifica y analiza información para fundamentar y responder a las necesidades propias de sus actividades.

2. Resolución de problemas

1.Reconoce y previene problemas de acuerdo a parámetros establecidos en contextos conocidos propios de su actividad o funcion.
2.Detecta las causas que originan problemas en contextos conocidos de acuerdo a parámetros establecidos.
3.Aplica soluciones a problemas de acuerdo a parámetros establecidos en contextos conocidos propios de una función.

3. Uso de recursos

1.Selecciona y utiliza materiales, herramientas y equipamiento para responder a una necesidad propia de una actividad o función especializada en contextos conocidos.
2.Organiza y comprueba la disponibilidad de los materiales, herramientas y equipamiento.
3.Identifica y aplica procedimientos y técnicas específicas de una función de acuedo a parámetros establecidos.

4. Comunicación

4. Comunica y recibe información relacionada a su actividad o función, a través de medios y soportes adecuados en contextos conocidos.

APLICACIÓN EN CONTEXTO

5. Trabajo con otros

1.Trabaja colaborativamente en actividades y funciones coordinándose con otros en diversos contextos.

6. Autonomía

1.Se desempeña con autonomía en actividades y funciones especializadas en diversos contextos con supervisión directa.
2.Toma decisiones en actividades propias y en aquellas que inciden en el quehacer de otros en contextos conocidos.
3.Evalua el proceso y el resultado de sus actividades y funciones de acuerdo a parámetros establecidos para mejorar sus prácticas.
4.Busca oportunidades y redes para el desarrollo de sus capacidades

7. Ética y responsabilidad

1.Actúa de acuerdo a las normas y protocolos que guían su desempeño y reconoce el impacto que la calidad de su trabajo tiene sobre el proceso productivo o la entrega de servicios.
2.Responde por cumplimiento de los procedimientos y resultados de sus actividades.
3.Comprende y valora los efectos de sus acciones sobre la salud y la vida, la organización, la sociedad y el medio ambiente.
4.Actúa acorde al marco de sus conocimientos. experiencias y alcance de sus actividades y funciones

CONOCIMIENTO

8. Conocimientos

1.Demuestra conocimientos especificos de su área y de las tendencias de desarrollo para el desempeño de sus actividades y funciones.

Metodología seleccionada

Demostración guiada

- Esta presentación les ayudará a poder comprender los conceptos necesarios para el desarrollo de su actividad

Aprendizaje Esperado

- **AE3.** Diseña programas de baja complejidad aplicados a su entorno, empleando el lenguaje de programación Python, considerando tipos de datos, sentencias básicas (condicionales e iterativas) y diversas estructuras de datos.

¿Qué vamos a lograr con esta actividad para llegar al Aprendizaje Esperado (AE)?

- **Diseñar** aplicaciones en Python, utilizando sentencias básicas del lenguaje de programación (I/O, condicionales, iterativas), para solucionar problemas de enunciado aplicados a la vida cotidiana.

Contenidos

01 SENTENCIAS BÁSICA DE Python

- ¿Qué es Python?
- ¿Cómo escribimos un programa en Python?
- El ingreso de datos al programa.
- Las variables.
- Tipos de datos.
- Mostrar un mensaje por pantalla.
- Ingresar datos al programa.
- Los operadores.
- Ejemplos de aplicación.

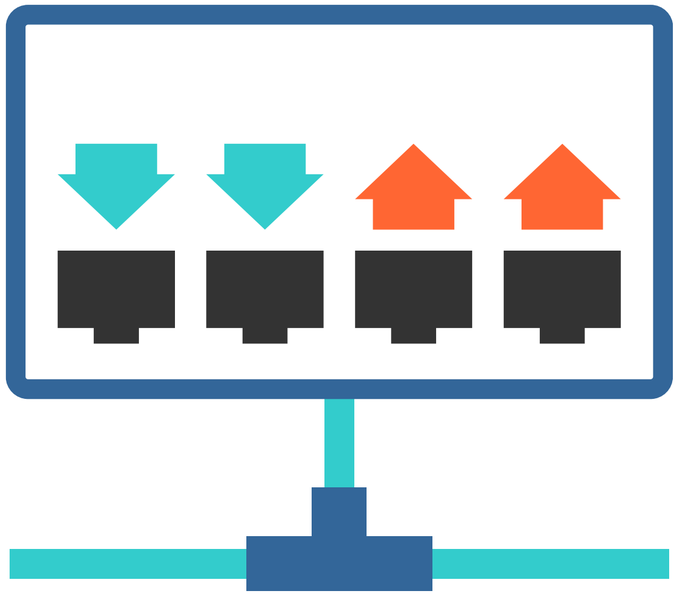
02 SENTENCIA CONDICIONAL

- Las condiciones.
- La sentencia ‘if’.
- Consideraciones.
- Ejemplos de uso.
- La sentencia ‘IF’ compuesta.

03 SENTENCIAS ITERATIVAS

- Las bucles o ciclos.
- El ciclo ‘while’.
- El ciclo ‘for’.
- Ejemplos de aplicación.
- Consideraciones generales.

Sentencia básica de Python



Recordando contenidos vistos anteriormente...

Si escribimos un algoritmo para sumar dos números, ¿cómo logramos que el computador lo entienda?

¿Qué es Python?

- Python es un lenguaje de programación muy flexible y potente. Fue creado por Guido Van Rossum en 1991. Dado la claridad y simpleza de su sintaxis, Python se ha transformado en uno de los lenguajes con mayor demanda en la industria.

¿Qué es Python?

- Algunos de sus usos frecuentes son:

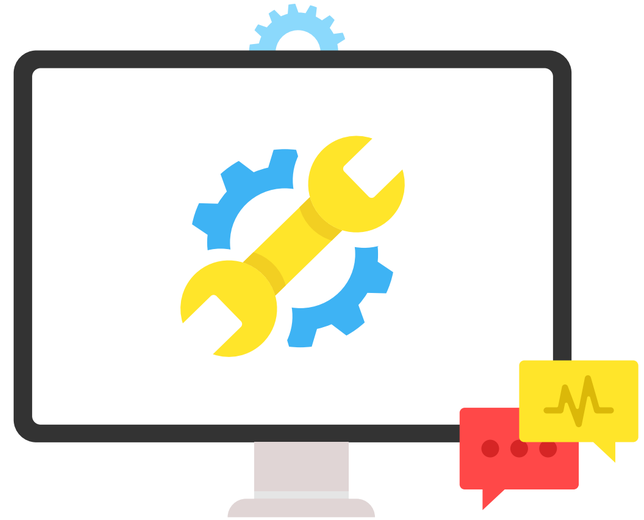
  - Construir de forma sencilla aplicaciones web con manejos de bases de datos.
  - Hacer análisis de datos y visualización de éstos.
  - Realizar captura de datos de una página web.
  - Desarrollar programas de aplicación.
  - Diseñar aplicaciones en el ámbito de las redes.

¿Cómo escribimos un programa en Python?

Para poder escribir un programa en Python necesitamos dos cosas:

- Primero, la parte más importante, que es la que ya hemos aprendido, y se refiere a analizar un enunciado y resolver el problema. A esta parte se le llama la **lógica** de la programación.
- En segundo lugar, necesitamos conocer el lenguaje, es decir, qué instrucciones utiliza, como se escriben y qué estructura tiene el programa. A esta parte le llamamos la **sintaxis** del programa o las reglas del lenguaje.

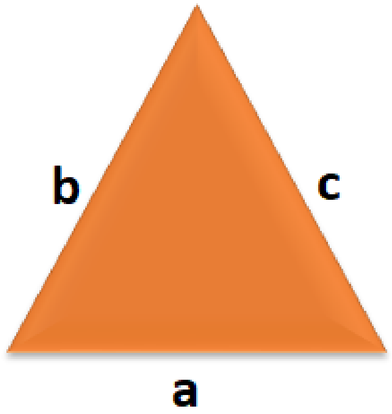
A continuación, conoceremos más acerca del lenguaje Python, ya que es el que utilizaremos para escribir los programas.



¿Cómo pasamos de un algoritmo a un programa en Python?

- Recordemos el problema inicial: “Calcular el perímetro de un triángulo”.

  Ya sabemos cómo se resuelve algorítmicamente:



P = a + b + c

INICIO

PASO 1: Conocer el primer lado.

PASO 2: Conocer el segundo lado.

PASO 3: Conocer el tercer lado.

PASO 4: Sumar los tres lados.

PASO 5: Mostrar el resultado.

FIN

Algoritmo

INICIO

PASO 1: Conocer el primer lado.

PASO 2: Conocer el segundo lado.

PASO 3: Conocer el tercer lado.

PASO 4: Sumar los tres lados.

PASO 5: Mostrar el resultado.

FIN

Python

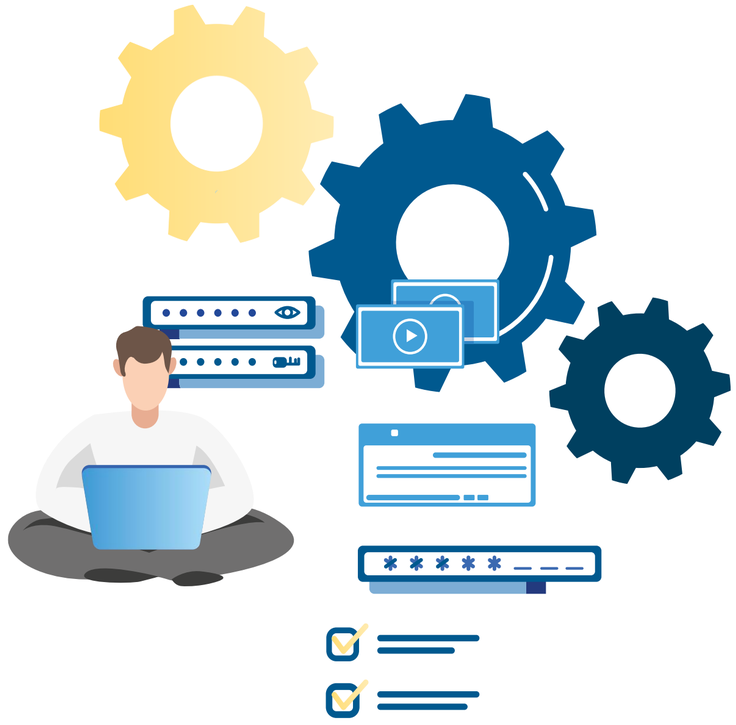
```
print("Ingrese valor del primer lado: ")
a=int(input())
print("Ingrese valor del segundo lado: ")
b=int(input())
print("Ingrese valor del tercer lado: ")
c=int(input())
perimetro=a+b+c
print("El perimetro del triangulo es: ",perimetro)
```

SALIDA POR PANTALLA

```
Ingrese valor del primer lado:
 2
Ingrese valor del segundo lado:
 3
Ingrese valor del tercer lado:
 4
El perimetro del triangulo es:  9
```

Analicemos el programa

- Como pudiste observar, en el programa escrito en Python hay una serie de instrucciones escritas en lenguaje de programación, las cuales aún no conoces.



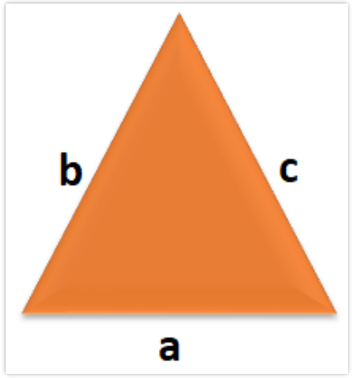
Analicemos el programa

Además, recordemos que, en un programa:

- La “**Entrada**” tiene que ver con valores que se ingresan al programa, los cuales deben ser solicitados al usuario a través de un mensaje.
- El “**Proceso**” tiene que ver con los cálculos, fórmulas y procedimientos aplicados para resolver el problema o el requerimiento.
- La “**Salida**” tiene que ver con los resultados que se muestran por pantalla a través de mensajes.

Analicemos el programa

- Las instrucciones de un programa deben ser claras, para que el usuario que ejecuta el programa entienda qué se le pide que ingrese, y qué respuesta se le esta entregando. Por ejemplo, en el caso del Triángulo:

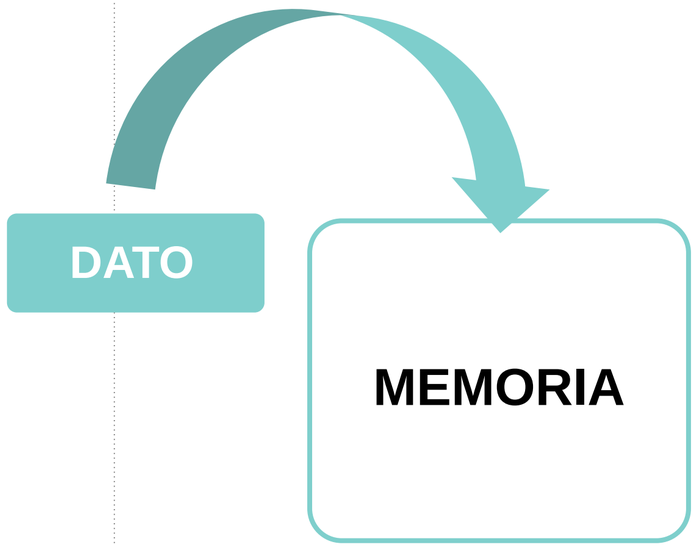



```
Ingrese valor del primer lado:
 2
Ingrese valor del segundo lado:
 3
Ingrese valor del tercer lado:
 4
El perimetro del triangulo es:  9
```

- Observando los mensajes, se puede entender con claridad qué valores debe ingresar y a qué corresponde el valor mostrado como respuesta.

El ingreso de datos al programa

Si nos enfocamos en la **Entrada** o ingreso de los datos por parte del usuario, nos damos cuenta que se requieren dos cosas:



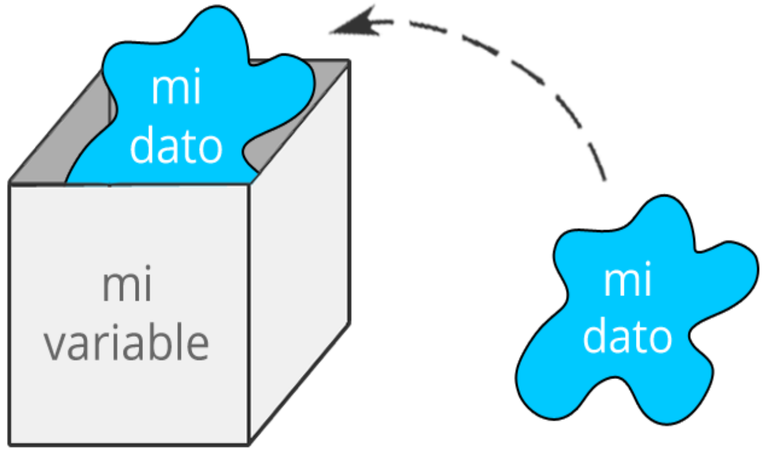
- Un mensaje que solicite el valor que se debe ingresar.
- Una sentencia que permita leer el valor ingresado.

¿Qué sucede con los valores ingresados al programa? Estos valores son almacenados en la memoria del computador mientras se ejecuta el programa. A dicho lugar de la memoria donde se guarda el dato le asignamos un nombre para identificarlo. A este proceso lo denominamos “almacenar el dato en una variable”.

Las variables

Pero, ¿qué es una variable?

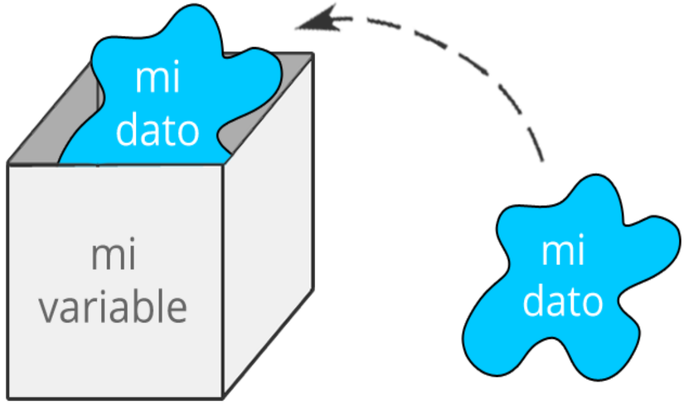
En términos simples, una variable es un espacio en la memoria de la computadora al cual se le asigna un nombre para identificarlo.



Las variables

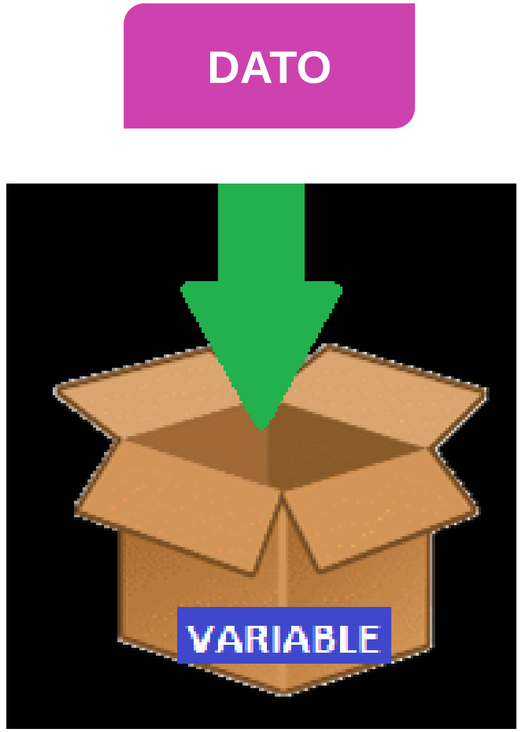
- Una variable permite guardar o almacenar un **dato** (número, letra, símbolo, nombre, etc).

El nombre variable proviene del hecho de que el contenido puede variar durante la ejecución del programa. Es decir, primero podemos guardar un valor y luego otro en la misma variable.



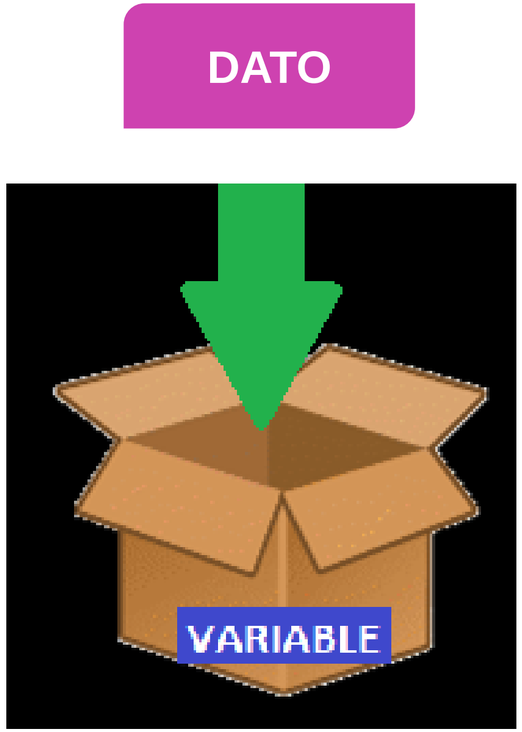
Las variables

- Una variable se llama por su **nombre**.
- El nombre de una variable puede ser cualquiera, pero se recomienda que se utilice un nombre de acuerdo al dato que va a guardar, y que siempre se utilicen letras o texto para los nombres.



Las variables

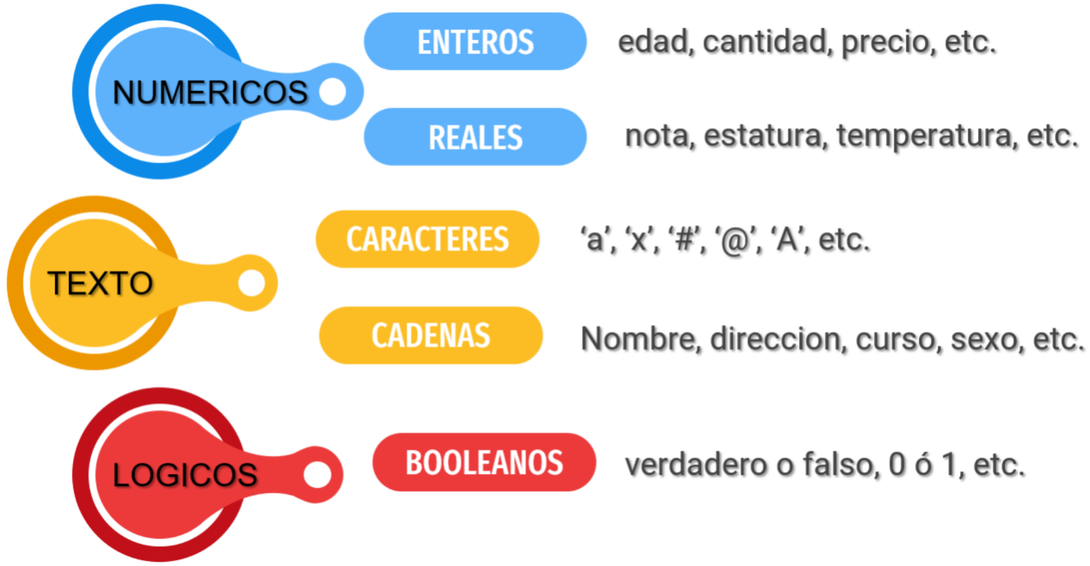
- Por ejemplo, si la variable guardar un número se recomienda que se llame num, si va a guardar un nombre se podría llamar nom, si va a guardar una edad se puede llamar ed o edad, etc.
- Una variable asume el **tipo de dato** del contenido que se le asigna.
- ¿Cuáles son estos tipos de datos posibles?



Tipos de datos

- En lo cotidiano, todos los datos que usamos pueden clasificarse en algún tipo.

- El siguiente esquema representa los posibles **tipos de datos** a asignar en una variable:



Tipos de datos

Los tipos de datos en Python se identifican de la siguiente manera:

- **INT** para números enteros.
- **FLOAT** para números reales.
- **STR** para caracteres y cadenas.
- **BOOL** para lógicos o booleanos.

Mostrar un mensaje por pantalla

Una de las sentencias más básicas de todo lenguaje de programación se relaciona con el hecho de enviar mensajes adecuados al usuario por pantalla.

- La sentencia “**print**” nos permite mostrar un mensaje por pantalla que será leído por el usuario.
- Su forma más simple es: **print** (“mensaje”).
- Ejemplos:

```
print("*** Bienvenido al programa ***")
print("Hola!")
print("Ingrese un numero: ")
```

```
*** Bienvenido al programa ***
Hola!
Ingrese un numero:
```

SALIDA POR PANTALLA

Mostrar un mensaje y variables

- Para asignar un valor predeterminado a una variable se utiliza el símbolo “=“, el cual se conoce como **operador de asignación**, y se usa de la siguiente manera:

  Nom = “Juan”.
  Edad = 16.
  Curso = “3A”.

- Si se quiere mostrar un mensaje acompañado de una variable, se escribirá así:

  print (“mensaje”, **variable)**

Ejemplos:

Print (“Mi nombre es: “, **nom**).
Print (“Edad = “, **edad**).
Print (“Especialidad **??** “, **curso**).

```
nom="Juan"
edad=16
curso="3A"

print("Mi nombre es: ",nom)
print("Edad = ",edad)
print("Curso -->",curso)
```

```
Mi nombre es:  Juan
Edad =  16
Curso --> 3A
```

¿Cómo ingresar datos al programa?

- Como vimos anteriormente, se le puede asignar un dato a una variable así: x = 0

  (esto significa que a la variable 'x' se le asigna el valor 0)

- Pero la forma correcta de ingresar datos o valores a las variables del programa es a través de la sentencia "**input**", de la siguiente manera:

  variable = **input( )**

DATO=input()

¿Cómo ingresar datos al programa?

- De esta manera es el usuario el que puede ingresar un valor cualquiera por teclado.

- Hay que dejar claro que la sentencia input permite el ingreso de una variable que será siempre de tipo texto o cadena.

DATO=input()

¿Cómo ingresar datos al programa?

- Usualmente el ingreso de un dato o valor va acompañado de un mensaje que lo solicita. Este mensaje puede ir dentro del mismo input, o antes del ingreso, utilizando print.

- Veamos algunos ejemplos…

```
num=input("Ingrese un numero: ")
```

```
Ingrese un numero: 3
```

```
print("Ingrese un numero: ")
num=input()
```

```
3

Ingrese un numero:
```

¿Cómo ingresar datos al programa?

- Habíamos dicho que los datos ingresados con input son todos de tipo cadena o texto. Pero, ¿qué sucede si quiero ingresar números enteros o reales para efectuar cálculos por ejemplo?

- Esto se soluciona anteponiendo el tipo de dato requerido a la sentencia input, de la siguiente manera:

variable= **tipo(**input( )**)**

```
print("Ingrese numero entero: ")
num=int(input())
```

```
Ingrese numero entero:
3
```

```
print("Ingrese numero real: ")
num=float(input())
```

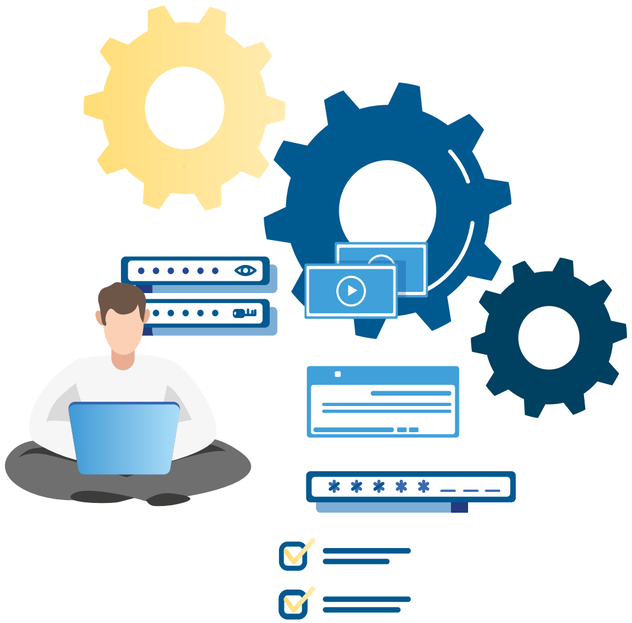
```
Ingrese numero real:
2.5
```

¿Qué podemos hacer con los datos en un programa?

- Dependiendo del enunciado del problema a resolver, sabremos qué tenemos que hacer con los datos.

- En esta parte ya nos referimos al **Proceso** que se hará con los datos para obtener la(s) respuesta(s) requerida(s).

¿Qué podemos hacer con los datos en un programa?



- Usualmente los procesos son operatorias que debemos realizar con los datos. Esto quiere decir que tendremos que “**operar**” los datos entre sí de alguna manera. Por ejemplo: sumar, multiplicar, elevar al cuadrado, etc.

- Para ello necesitaremos “**operadores**” que nos permitan realizar estos cálculos u operaciones con los datos.

Los operadores

- En Python existen muchos operadores, los cuales están clasificados por el tipo de operatoria que realizan.

  Estos son los siguientes:

- **ARITMÉTICOS:** permiten operar matemáticamente los datos contenidos en las variables. Usualmente la respuesta obtenida es de tipo numérica.

OPERADORES ARITMETICOS	
+	SUMA
-	RESTA
*	MULTIPLICACION
**	POTENCIA
/	DIVISION
//	DIVISION ENTERA
%	RESTO DE UNA DIVISION

Los operadores

- **RELACIONALES:** permiten relacionar los datos o, de cierta forma, comprar los valores de las variables. Usualmente la respuesta obtenida es Verdadero o Falso (True/False).

OPERADORES RELACIONALES	
<	MENOR
<=	MENOR O IGUAL
>	MAYOR
>=	MAYOR O IGUAL
==	IGUAL
!=	DISTINTO
IS	ES
IN	ESTA

Los operadores

- **LÓGICOS:** permiten establecer más de una operación relacional entre las variables a través de un conector lógico. Al igual que en el caso anterior, usualmente se obtiene una respuesta del tipo True o False.

OPERADORES LOGICOS	
AND	'Y' LOGICO
OR	'O' LOGICO
NOT	NEGACION

Veamos algunos ejemplos

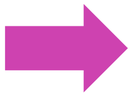
- **EJEMPLO 1:** “Calcular el área de un rectángulo”.

  SOLUCIÓN: Para resolver el problema necesitamos conocer el valor de largo y el ancho del rectángulo, para ello solicitaremos estos valores al usuario, luego debemos calcular el área del rectángulo utilizando el operador aritmético adecuado y, finalmente, debemos mostrar el resultado del cálculo a través de un mensaje.

- Utilizando las sentencias del lenguaje de programación lo escribiremos así:

```
print("Ingrese largo del rectangulo: ")
largo=float(input())
print("Ingrese ancho del rectangulo: ")
ancho=float(input())
area=largo*ancho
print("El area del rectangulo es: ", area)
```

SALIDA POR PANTALLA



```
Ingrese largo del rectangulo:
5
Ingrese ancho del rectangulo:
2.5
El area del rectangulo es:  12.5
```

Veamos algunos ejemplos

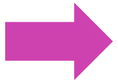
- **EJEMPLO 2:** “Determinar el promedio de 3 números enteros”.

  SOLUCIÓN: Para resolver el problema necesitamos conocer el valor de los tres números enteros, para ello solicitaremos estos valores al usuario, luego debemos calcular el promedio de los números utilizando las operaciones matemáticas adecuadas y, finalmente, debemos mostrar el resultado del promedio a través de un mensaje.

- Utilizando las sentencias del lenguaje de programación lo escribiremos así:

```
print("Ingrese primer numero: ")
num1=int(input())
print("Ingrese segundo numero: ")
num2=int(input())
print("Ingrese tercer numero: ")
num3=int(input())
promedio=(num1+num2+num3)/3
print("El promedio de los numeros es: ", promedio)
```

SALIDA POR PANTALLA



```
Ingrese primer numero:
3
Ingrese segundo numero:
5
Ingrese tercer numero:
7
El promedio de los numeros es:  5.0
```

Veamos algunos ejemplos

EJEMPLO 3: “Determinar el sueldo líquido de un empleado, conociendo el sueldo bruto que gana y considerando la cantidad de horas extras que trabajó, donde cada hora extra vale el 5% del sueldo bruto. Considere además los descuentos legales de AFP e Isapre que corresponden al 20% aproximadamente”.

SOLUCIÓN: Para resolver el problema necesitamos conocer el valor del sueldo bruto y la cantidad de horas extra que trabajó el empleado, los cuales solicitaremos al usuario. Luego debemos calcular el sueldo líquido considerando los descuentos y el valor de la hora extra. Finalmente, debemos mostrar el resultado del cálculo.

Veamos algunos ejemplos

- Utilizando las sentencias del lenguaje de programación lo escribiremos así:

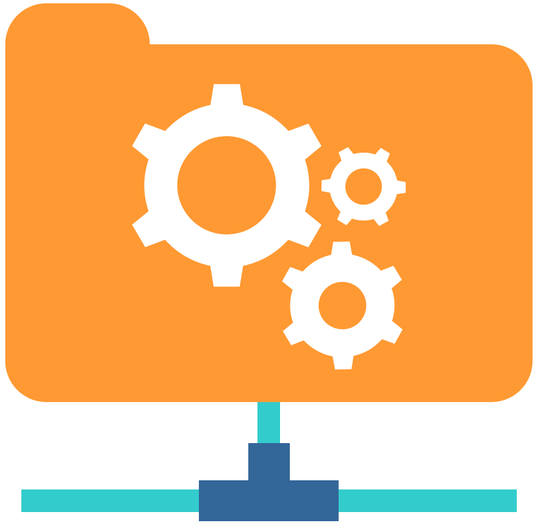
```
print("Ingrese Sueldo Bruto: $")
sb=int(input())
print("Ingrese cantidad de Horas Extra trabajadas: ")
hx=int(input())
AFP_ISA=sb*20/100
extra=sb*5/100*hx
liquido=sb-AFP_ISA+extra
print("Sueldo Liquido = $", liquido)
```

SALIDA POR PANTALLA

```
Ingrese Sueldo Bruto: $
1000000
Ingrese cantidad de Horas Extra trabajadas:
10
Sueldo Liquido = $ 1300000.0
```

En dos palabras….
¿Cómo escribimos un programa en Python?

Sentencia Condicional



Actividad de motivación

- ***(Para que suceda algo que planeamos a veces, deben darse ciertas condiciones…).***

- “Suponga que está organizando una salida con sus amigos al cine, desean ver cierta película, *¿qué harían en el caso de que no queden entradas para esa película?”.*

Las condiciones

EJEMPLO: Supongamos que queremos comprar un producto en una tienda, eso será posible sólo si el dinero que se tiene es mayor o igual que el precio del producto, de lo contrario no se podrá realizar la compra.

Las condiciones

Como se observa en el ejemplo, para poder efectuar la compra de un producto se necesita que se cumpla una **CONDICIÓN** (que la cantidad de dinero sea mayor o igual al precio del producto). Esta condición puede ser **VERDADERA** o **FALSA**. En caso de que se cumpla la condición se efectúa la compra, de lo contrario, no será posible realizar la compra.

COMPRA EXITOSA

COMPRA FALLIDA

Las condiciones

- En programación, una condición es toda sentencia de la cual se puede determinar su **verdad** (TRUE) o **falsedad** (FALSE).

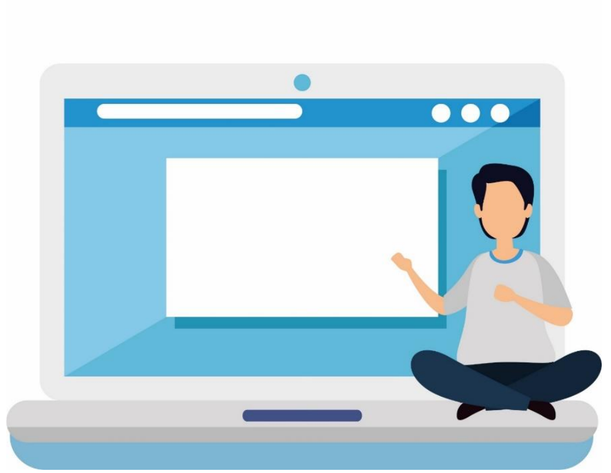
  En su gran mayoría son comparaciones. Por ejemplo, **4 > 5**. Esta sentencia es una condición porque tiene resultado verdadero o falso, en este caso falso, porque 4 no es mayor a 5.

Las condiciones

Hay que considerar que toda condición puede ser solo verdadera o solo falsa, no ambas a la vez, por lo tanto, solo se ejecutarán las instrucciones según la verdad o falsedad de la condición.

Según el ejemplo anterior, tengo o no tengo el dinero para comprar, pero no ambas cosas a la vez.

Existe una sentencia condicional en Python, que permite validar la verdad o falsedad de una o más condiciones.



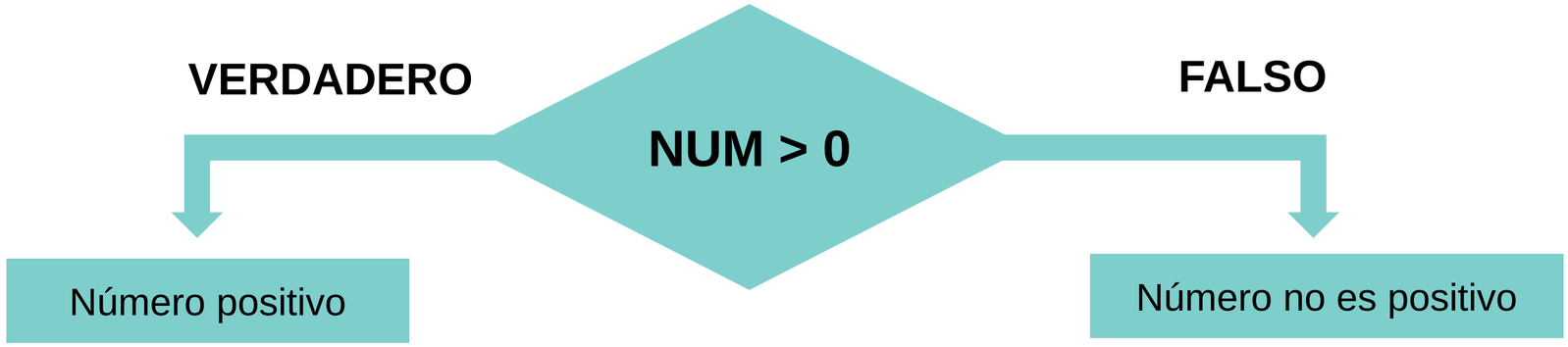
La sentencia “IF” (simple)

SINTAXIS:

```
if condición:
    instrucciones
else:
    instrucciones
```

EJEMPLO:

```
if num>0:
    print(“Numero Positivo”)
else:
    print(“Numero no  es Positivo”)
```



Consideraciones

- Todas las instrucciones que están dentro de “**IF**” sucederán sólo si la condición es **verdadera**, mientras que las instrucciones que están dentro de “**ELSE**” sucederán sólo si la condición es **falsa**.

- Puede existir un IF **sin** un ELSE (en su forma más simple), es decir, que solo se hará algo si la condición se cumple.

Consideraciones

- La sentencia IF utiliza **:** (**dos puntos**) al final de su declaración.
- El código dentro del IF o ELSE debe estar **indentado** (tabulado por **4 espacios** hacia la derecha), esto para poder reconocer de forma sencilla qué instrucciones se ejecutan dentro de la sentencia.

Ejemplos

Solucionemos los siguientes problemas y desarrollémoslos en Python:

EJERCICIO 1: Crear un programa donde el usuario deba ingresar 2 números y determine cuál de ellos es el mayor, o si son iguales.

SOLUCIÓN: Primero debemos conocer los dos números, luego compararlos y mostrar el mayor de los dos, o si son iguales.

```
print("Ingrese primer numero: ")
num1=int(input())
print("Ingrese segundo numero: ")
num2=int(input())
if num1>num2:
    print("El mayor es ", num1)
else:
    if(num2>num1):
        print("El mayor es ",num2)
    else:
        print("Los numeros son iguales")
```

Si observamos la solución del problema, nos damos cuenta que primero se pregunta si num1 es mayor que num2, si es así, mostrará un mensaje, de lo contrario, se debe preguntar si num2 es mayor que num1, y si es así, se mostrará otro mensaje. Finalmente, si ninguno de los dos es mayor, se asume que los números son iguales, ya que es la ultima opción que queda.

Ejemplos

EJERCICIO 2: Crear un programa donde el usuario debe ingresar un usuario y clave, y validar que el usuario sea “jperez” y la clave 12345. En caso contrario, enviar un mensaje de error.

SOLUCIÓN: Primero debemos solicitar al usuario que ingrese nombre y clave de usuario, luego comparar los valores ingresados con los valores aceptados, si los datos son correctos se debe enviar un mensaje, de lo contrario se debe enviar el mensaje de error.

```
print("Ingrese nombre de usuario: ")
user=input()
print("Ingrese clave: ")
clave=int(input())
if user=="jperez" and clave==12345:
    print("Usuario Aceptado")
else:
    print("Usuario No Aceptado")
```

Si observamos la solución del problema, nos damos cuenta que hay dos condiciones que se deben cumplir a la vez, que el usuario y la clave sean los correctos. Para preguntar por más de una opción en un IF se utilizan los operadores lógicos. En este caso el “and” o “y” lógico, que significa que se deben cumplir la primera y también la segunda condición para que el resultado de la sentencia sea verdadera.

La sentencia “IF” (compuesta)

- Consideremos el siguiente caso. Si una persona está en un rango de edad, no puede estar en ninguno de los otros rangos. Esto quiere decir que si es menor de edad, no puede ser adulto ni adulto mayor, por ejemplo.

La sentencia “IF” (compuesta)

- La sentencia condicional IF compuesta admite más de una condición, las cuales son independientes entre sí, esto quiere decir que solo una de ellas será verdadera. Para escribir varias sentencias en un IF se utiliza la sentencia **ELIF** como se muestra en la sintaxis y ejemplo.

```
edad = int(input("¿Qué edad tienes? "))
if edad>0 and edad<18:
    print("Eres menor de edad")
elif edad>=18 and edad<60:
    print("Eres adulto")
elif edad>=60 and edad<=100:
    print("Eres adulto mayor")
else:
    print("Edad fuera de rango")
```

SINTAXIS:

```
If condición:
    instrucciones

elif condicion:
    instrucciones

Elif condicion:
    instrucciones

else:
    instrucciones
```

Resuma, en una frase, lo que son las condiciones

Sentencias Iterativas

Actividad de motivación

01 ¿Cómo mostrarían la palabra “Hola” por pantalla **1** vez?

02 ¿Cómo mostrarían la palabra “Hola” por pantalla **10** veces?

03 ¿Cómo mostrarían la palabra “Hola” por pantalla **100** veces?

Los bucles o ciclos

Para solucionar el problema anterior existen los llamados “**bucles**”.

Un **bucle** nos permitirá realizar una serie de instrucciones tantas veces como queramos, dependiendo de una **condición** que se cumple, el bucle **se detiene** cuando la condición deja de cumplirse.

Los bucles o ciclos

- La condición para que el bucle se detenga es necesaria, de lo contrario, el ciclo sería infinito y no terminaría nunca.
- Los bucles también son llamados “**ciclos**”.
- En Python existen dos ciclos: **while** y **for**

Ciclo “While”

- El ciclo “**while**” significa “**mientras**”, es decir, que mientras se cumpla una condición, se realizarán las instrucciones.

- La sentencia while tiene la siguiente sintaxis, es decir, en su forma general se escribe así:

while *condición*:
sentencias

- *Veamos un ejemplo:* Deseamos escribir un programa que muestre la palabra “Hola” por pantalla, 10 veces.

Ciclo “While”

Para escribir **10 veces** la palabra, necesitamos contar. Para ello utilizaremos una variable que nos ayudará a contar de 1 hasta 10. A esta variable especial le llamaremos **contador**.

01 Necesitamos colocar el valor inicial de la variable en 1 (**inicializar** la variable).

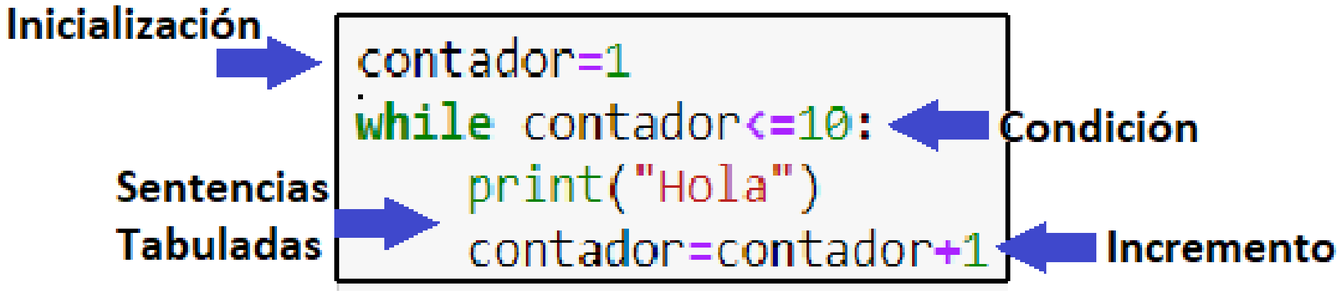
02 Luego tenemos que señalar que la condición del ciclo será que los valores sean menores o iguales a 10 (**contador<=10**).

Ciclo “While”

03 Luego se imprime la palabra “Hola” por pantalla, teniendo en cuenta que dentro del ciclo las instrucciones se escriben con 4 espacios hacia la derecha (**tabular**).

04 Finalmente aumentamos el valor de la variable en 1 unidad (**incrementar**), de tal manera que vaya avanzando de 1 en 1 hasta llegar a 10, para que el ciclo se detenga en algún momento.

Ciclo “While”



- Según la solución planteada, al ejecutar el programa se observa la siguiente salida por pantalla:

```
Hola
Hola
Hola
Hola
Hola
Hola
Hola
Hola
Hola
Hola
Hola
```

Ciclo “FOR”

- El ciclo “**for**” significa “**para**”, es decir, que para ciertos valores del **contador** en un **rango**, se realizarán las instrucciones.

- La sentencia **for** tiene la siguiente sintaxis, es decir, en su forma general se escribe así:

  for *variable* **in** *range(**n**)*:
  sentencias

- *<u>Retomemos el ejemplo anterior:</u>* Deseamos escribir un programa que muestre la palabra “Hola” por pantalla, 10 veces.

Ciclo “FOR”

Para escribir **10 veces** la palabra, necesitamos contar. Para ello utilizaremos una variable que haga las veces de contador y cuente valores en el **rango** especificado en el ciclo.

01 No necesitamos colocar el valor inicial de la variable, en este caso el mismo ciclo la **inicializa en 0.**

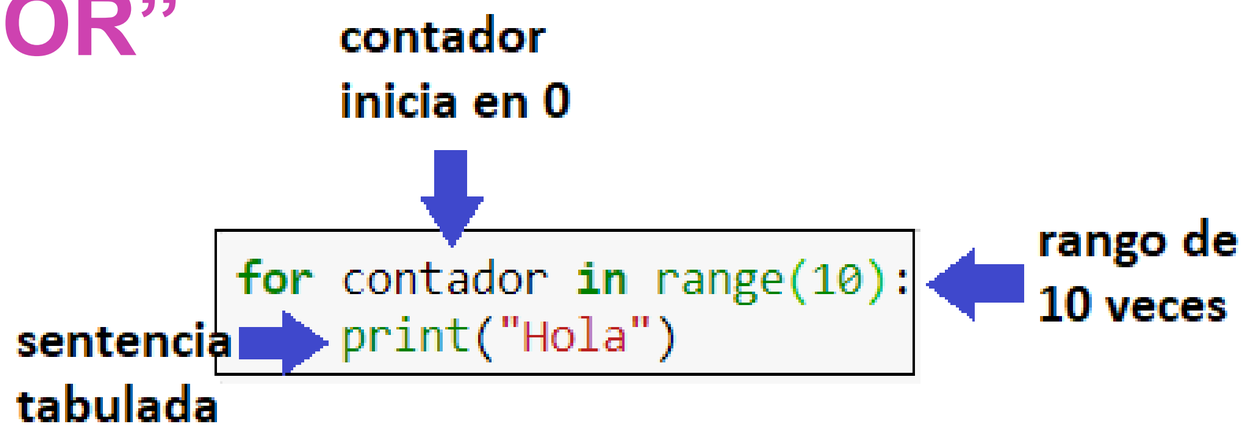
02 Luego tenemos que señalar el rango de valores que queremos que cuente la variable, en este caso es 10, por lo cual se utiliza **range (10),** entonces la variable contará de 0 hasta 9, completando así las 10 veces, al incluir el 0.

Ciclo “For”

03 Luego se imprime la palabra “Hola” por pantalla, teniendo en cuenta que dentro del ciclo las instrucciones se **tabulan con 4 espacios** hacia la derecha.

04 En este caso la variable se incrementa de 1 en 1 **automáticamente**, así que no es necesario escribir la instrucción, y el ciclo finalizará cuando complete la cantidad de veces requerida por el rango.

Ciclo “FOR”



- Según la solución planteada, al ejecutar el programa se observa la siguiente salida por pantalla:

```
Hola
Hola
Hola
Hola
Hola
Hola
Hola
Hola
Hola
Hola
```

Ejemplos con bucles

EJERCICIO 1: Escriba un programa en Python que permita ingresar 5 números, calcular y mostrar el doble de cada número.

UTILIZANDO CICLO “FOR”

```
for contador in range(5):
    num=int(input("Ingrese un numero: "))
    doble=2*num
    print("El doble de ",num," es: ",doble)
```

```
contador=1
while contador<=5:
    num=int(input("Ingrese un numero: "))
    doble=2*num
    print("El doble de ",num," es: ",doble)
    contador=contador+1
```

UTILIZANDO CICLO “WHILE”

Ejemplos con bucles

EJERCICIO 2: Escriba un programa en Python que permita ingresar 6 números, calcular y mostrar la suma de los números.

SOLUCIÓN: En este caso utilizaremos algo parecido a un contador, variable a la cual llamaremos **acumulador**, ya que nos permitirá acumular la suma de cada uno de los 6 números ingresados.

Se escribe igual que un contador, solo que en vez de contar de uno en uno, sumará los números ingresados.

Ejemplos con bucles

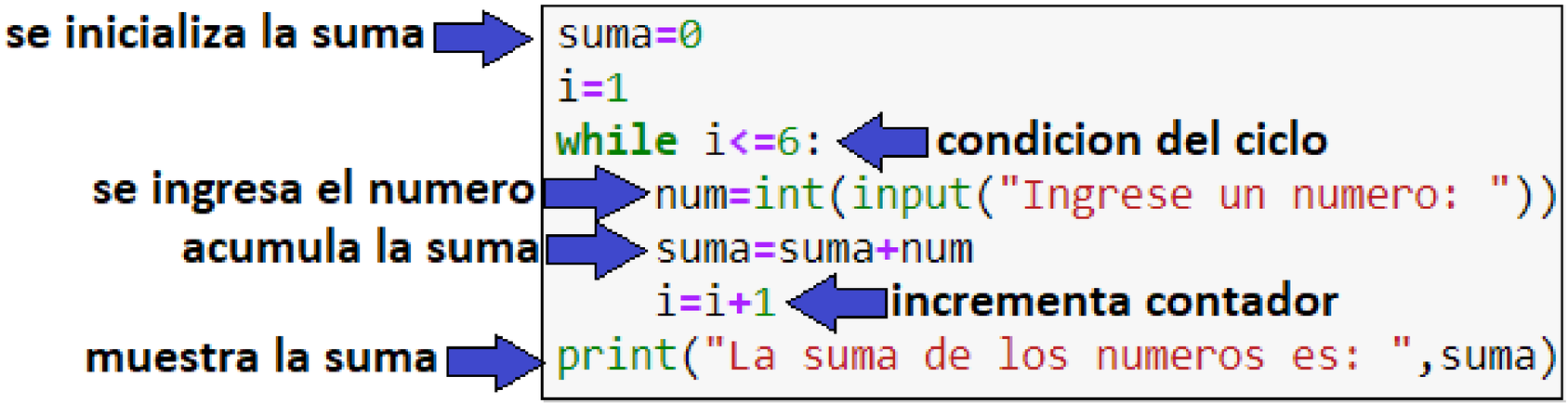
- **EJERCICIO 2:** Escriba un programa en Python que permita ingresar 6 números, calcular y mostrar la suma de los números.

SOLUCIÓN: En este caso utilizaremos algo parecido a un contador, variable a la cual llamaremos **acumulador,** ya que nos permitirá acumular la suma de cada uno de los 6 números ingresados.

Se escribe igual que un contador, solo que en vez de contar de uno en uno, sumará los números ingresados.

Veamos la solución...

Ejemplos con bucles

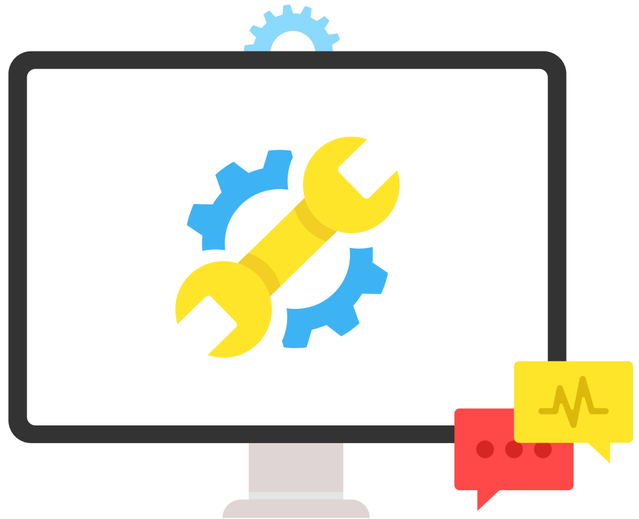


Consideremos algunas cosas importantes...

Ejemplos con bucles

En primer lugar, para comenzar a sumar debemos inicializar el acumulador en 0, para que solo se sumen los valores ingresados y no otro que haya sido asignado en el acumulador (usamos el 0 como neutro aditivo).

En segundo lugar, inicializamos el contador y utilizamos el ciclo para ingresar solo 6 números.



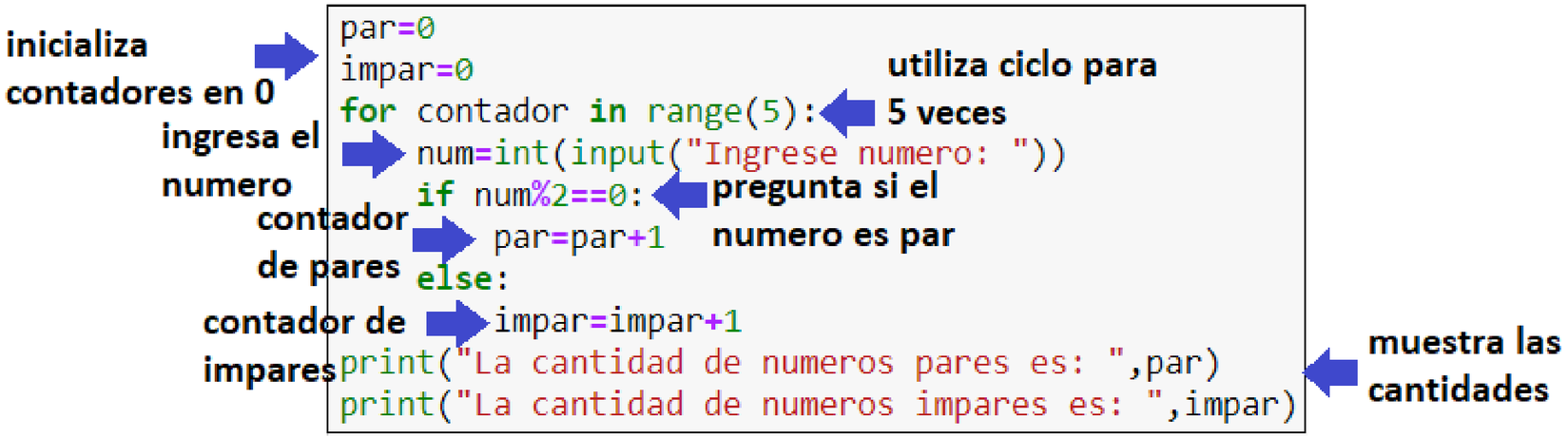
Ejemplos con bucles

- En tercer lugar, de la misma manera que lo hacíamos con el contador, una vez que se ingresa el numero, éste se suma en el acumulador.

- Una vez que termina el ciclo, se muestra el resultado de la suma.

Ejemplos con bucles

- **EJERCICIO 3:** Escriba un programa en Python que permita ingresar 5 números, calcular y mostrar la cantidad de números pares y la cantidad de números impares.



Consideraciones generales

- Las sentencias IF, WHILE y FOR utilizan **dos puntos** (**:**) al final de su declaración.

- Las instrucciones que están dentro de una sentencia se escriben con **tabulación** a la derecha, la cual, equivale a **4 espacios del teclado**.

Consideraciones generales

- Se pueden utilizar sentencias condicionales e iterativas en conjunto.

- Los contadores y acumuladores utilizan la **misma sintaxis** general, solo que el contador acumula un **valor constante** y el acumulador acumula el **valor de una variable**:

***contador = contador +* valor**

***acumulador = acumulador +* variable**

Resuma, en una frase, lo que son las sentencias iterativas.

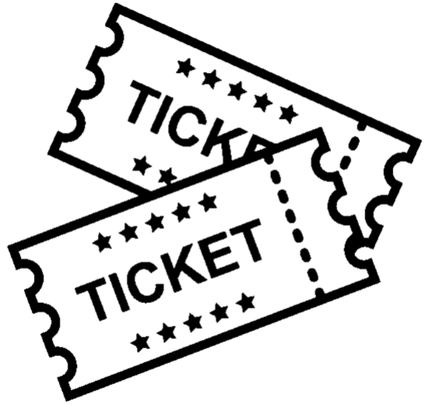
¿Tienes preguntas de lo trabajado hasta aquí?

Referencias de contenido:

Curso Cisco Python “Fundamentos Python Profesores MT RM3 - PCAP - Programming Essentials in Python”. Programa técnicos para Chile, Ministerio de Educación.

Curso Academia Desafío LATAM “Introducción a Python”.

Ticket de salida



01 Menciona 2 sentencias básicas de Python.

02 Menciona 3 operadores de Python.

03 ¿Cómo podrías explicar lo que es una sentencia condicional a alguien que tiene poco conocimiento del tema?

04 Escribe 1 ejemplo cotidiano de sentencia condicional.

05 ¿Cómo le explicarías a un cliente, para qué se utilizan las sentencias iterativas en un programa?

06 ¿Qué debilidades percibiste en tu desempeño durante el desarrollo de la actividad? ¿Cómo puedes trabajarlas para convertirlas en fortalezas?