



Provincia de Tierra del Fuego,
Antártida e Islas del Atlántico Sur.
República Argentina
Ministerio de Educación, Cultura, Ciencia y Tecnología
Centro Educativo Técnico de Nivel Superior "Malvinas Argentinas"



CENTRO EDUCATIVO DE NIVEL SUPERIOR "MALVINAS ARGENTINAS"

Capacitación Laboral

Programación I

Año: 2023

Docente: Noriega Agostina Gisel

Bienvenidos a la Capacitación Laboral Programación I, la misma forma parte de la certificación de nivel III de Programador.

Programación I tiene como propósito general iniciarlos en la construcción de capacidades técnicas en torno a la lógica de programación. Se parte conceptualmente de analizar problemas de base computacional con el fin que adquieran los conceptos y las técnicas básicas de programación estructurada. Tiene como propósito general la construcción de habilidades y conocimientos para resolver problemas e implementar soluciones en un lenguaje de programación, logrando código legible y mantenible.

Durante las próximas semanas, explorarán los fundamentos de la programación, desde la lógica y la estructura de datos hasta la sintaxis y la solución de problemas de programación.

Este módulo está diseñado para ayudarlos a comprender los conceptos básicos de la programación y a sentar las bases para futuros estudios en este campo. No te preocupes si no tienes experiencia previa en programación, ¡esta capacitación es para principiantes! Trabajaremos juntos para desarrollar su habilidad para escribir código y diseñar programas.

Contenidos del módulo :

Definición y análisis de problemas del campo informático. Datos de entrada y salida, relación entre ellos. Variables y constantes. Operadores relacionales y lógicos. Operadores aritméticos. Concepto de algoritmo. Desarrollo de algoritmos: Técnicas de diseño de algoritmos: Estructurada. Herramientas para diseño de algoritmos: Diagramas. Estructuras de programación: Metodología estructurada: Estructura secuencial: Características. Definición de variables. Estructura alternativa. Características. Expresiones lógicas. Estructura repetitiva o iterativa Características. Fases de un programa iterativo. Pruebas de escritorio y depuración: Concepto de codificación. Lenguajes de programación (estructurado o gráfico)

Uso de la sala de informática

Buenas prácticas:

1. Velar por el buen uso de todo el mobiliario de la sala, no rayar las mesas ni sillas, tratar con cuidado todos los dispositivos componentes de la computadora, teclado, mouse, etc.
2. Mantener el área de trabajo ordenada, no colocar peso sobre el teclado, tener los elementos necesarios sobre la mesa.
3. No desconectar ningún cable de la pc.
4. Se prohíbe ingerir alimentos y/o bebidas dentro de la sala.
5. Al finalizar la clase cerrar los programas y apagar el equipo correctamente.
6. Los trabajos realizados deben ser guardados en la carpeta mis documentos y no en el escritorio, para que no sean eliminados.
7. Cualquier pérdida y desperfecto en los elementos y dispositivos de la sala por mala utilización deberá ser repuesto por los responsables, en caso de no identificar los responsables, el grupo completo debe responder por los daños ocasionados.

El buen uso aumentará la vida útil de las mismas, lo cual nos permitirá seguir capacitando a los distintos niveles en beneficio de nuestra comunidad ya que es una herramienta vital para el desarrollo de capacidades, imprescindible para alcanzar los conocimientos necesarios en las nuevas tecnologías.

Introducción a la Programación

La **programación** es el proceso utilizado para idear y ordenar las acciones necesarias para realizar un fin determinado, dar instrucciones a algún dispositivo o crear programas para su uso en computadoras.

Hoy en día asociamos el término principalmente con la creación de software y aplicaciones de informática.

Software vs Hardware

El **software** es el conjunto de componentes lógicos (código) que hacen posible la realización de tareas específicas, en contraposición de los componentes físicos que son llamados **hardware**.

Cuando mencionamos al software de sistema, no podemos dejar de referirnos al que tal vez sea el más importante: el sistema operativo.

Es este quien gestiona todos los recursos del hardware y provee servicios a los programas de aplicación de software.



Actividad 1

- a) Escribe tres componentes físicos y clasifica si son de entrada y/o salida de información .
- b) Investiga el Sistema Operativo de la pc y completa con los siguientes datos:
 - Nombre del Sistema Operativo:
 - Fabricante del SO:
 - Procesador:
 - Memoria RAM:

La computadora una máquina algorítmica

Los **algoritmos** son un conjunto de instrucciones o reglas definidas, no-ambiguas, ordenadas y finitas que permiten solucionar un problema, realizar un cómputo, procesar algún dato o llevar a cabo una tarea específica.

En la vida cotidiana utilizamos todo el tiempo algoritmos para resolver problemas. Cada vez que realizamos una secuencia predefinida de pasos para un fin estamos ejecutando un algoritmo.

Esto puede ser algo tan sencillo como una rutina o una receta, o algo tan complejo como una sucesión de instrucciones técnicas en alguna tarea determinada.



Actividad 2

En nuestra vida, de manera cotidiana recurrimos de manera constante a algoritmos para solucionar problemas y así realizar cosas, por ejemplo:

Cuando un profesor proporciona un conjunto de instrucciones para llevar a cabo un experimento, está especificando un algoritmo, que es seguido por los estudiantes y así obtienen datos para su análisis y aprendizaje. Piensa y escribe dos ejemplos.

Lenguajes de Programación. Niveles

Un **lenguaje de programación** es un lenguaje formal (es decir con reglas gramaticales y semánticas bien establecidas) que le proporciona a un individuo la posibilidad de determinar una serie de instrucciones o secuencias de órdenes en forma de algoritmos para poder controlar el comportamiento físico y lógico de un sistema informático.



Actividad 3

Lee atentamente cada ítems y clasifica en Lenguaje de Bajo (LB) y Alto nivel (LA)

	Usan instrucciones que se vinculan directamente con un tipo de computadora.
	Usan instrucciones que parecen a los lenguajes escritos, se ejecutan en una variedad de tipos de computadoras.
	Lenguaje de máquina y ensambladores.
	Usan características especiales de un tipo de computadora en particular y se ejecutan en el nivel más rápido posible.
	Java, Visual Basic, Python, C, C++



Actividad 4

Se requiere calcular la suma de dos números. Se pide generar la siguiente Salida Impresa:

- El resultado de la suma de los dos números. Para ello Ud. dispone de las siguientes Entradas:
- Número 1 (n1): identifica el primer número.
- Número 2 (n2): identifica el segundo número.

Para resolver este problema debemos:

- 1) Identificación de los componentes

ENTRADA	PROCESO	SALIDA
n1, n2	suma=n1+n2	suma



Actividad 5

Se requiere calcular el producto de dos números. Se pide generar la siguiente Salida Impresa:

- El resultado del producto de los dos números.

Para ello Ud. dispone de las siguientes Entradas:

- Número 1 (n1): identifica el primer número.
- Número 2 (n2): identifica el segundo número.

1) Identificación de los componentes

ENTRADA	PROCESO	SALIDA
n1, n2		producto



Actividad 6:

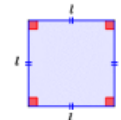
Se necesita averiguar el perímetro de un cuadrado. Se pide generar la siguiente Salida Impresa:

- El resultado del perímetro del cuadrado.

Para ello Ud. dispone de la siguiente Entrada:

- Lado1 (l1): identifica el lado, expresado en centímetros.

1) Identificación de los componentes



ENTRADA	PROCESO	SALIDA
l1		per



Actividad 7:

Desarrollar un algoritmo que declare dos variables enteras, le asigne valores arbitrarios y luego muestre su suma, diferencia y producto.

1) Identificación de los componentes.

ENTRADA	PROCESO	SALIDA
a=5 b=3		suma diferencia producto



Actividad 8:

Un mini mercado de nuestra ciudad necesita obtener información relacionada con el stock de 3 artículos de los productos que comercializa. Se pide generar las siguientes Salidas Impresas:

- El importe total en concepto de ventas de cada artículo.
- El importe total de los tres artículos considerados.



Para ello Ud. dispone de las siguientes Entradas:

- Cantidad Vendida Art1 (cant1): representa la cantidad vendida del artículo 1
- Precio Venta Art1 (pre1): representa el precio de venta del artículo 1.
- Cantidad Vendida Art2 (cant2): representa la cantidad vendida del artículo 2
- Precio Venta Art2 (pre 2): representa el precio de venta del artículo 2.
- Cantidad Vendida Art 3 (cant 3): representa la cantidad vendida del artículo 3.
- Precio Venta Art3 (pre3): representa el precio de venta del artículo 3.

1) Identificación de los componentes



Actividad 9:

Una persona necesita obtener información relacionada con el desempeño de su automóvil. Se pide generar las siguientes Salidas Impresas:

- La cantidad de litros consumidos.
- El importe gastado en combustible.

Para ello Ud. dispone de las siguientes Entradas:

- Kilómetros (km): representa los Km recorridos por el vehículo.
- Precio (pr): representa el precio del combustible por litro.
- Kilómetros Litros (kml): representa los km recorridos por cada litro.

1) Identificación de los componentes



Actividad 10:

El observatorio meteorológico necesita obtener información relacionada con la variación de temperaturas en distintos momentos del día. Se pide generar la siguiente Salida Impresa:

- La temperatura promedio del día.

Para ello Ud. dispone de las siguientes Entradas:

- Temperatura1 (t1): representa la temperatura tomada en horas de la mañana.
- Temperatura2 (t2): representa la temperatura tomada en horas de la tarde.
- Temperatura3 (t3): representa la temperatura tomada en horas de la noche.



Actividad 11:

Ingresar los votos obtenidos por dos candidatos en una elección e informar el porcentaje obtenido por cada uno.



Variables. Identificadores



Actividad 12:

Resalta con color los nombres de variables que son válidos.

GradARad	voltios\$	sumarNum	suma total	HallarMax
area+triang	3num	pendiente	cociente/rest	print



Actividad 13:

En cada ítems determinar el nombre para la variable resultante o de cálculo:

- Encontrar el promedio de un conjunto de números.
- Encontrar el área de un rectángulo.
- Encontrar el valor mínimo en un conjunto de números.
- Encontrar la densidad de una puerta de acero.
- Clasificar un conjunto de números de menor a mayor.
- Mostrar el Nombre y Apellido del operario.
- Mostrar correo electrónico de los empleados.



Actividad 14:

Determine los tipos de datos apropiados para los siguientes datos:

- a. el promedio de cuatro calificaciones
- b. el número de días en un mes
- c. los números en una lotería estatal
- d. dirección de los empleados de una fábrica
- e. perro o gato

Operadores Aritméticos



Actividad 15:

Determine el valor de las siguientes expresiones enteras:

- a. $3 + 4 * 6$
- b. $3 * 4 / 6 + 6$
- c. $2 * 3 / 12 * 8 / 4$
- d. $10 * (1 + 7 * 3)$
- e. $20 - 2 / 6 + 3$
- f. $20 - 2 / (6 + 3)$
- g. $(20 - 2) / 6 + 3$
- h. $(20 - 2) / (6 + 3)$
- i. $50 \% 20$
- j. $(10 + 3) \% 4$



Actividad 16:

Determine el valor de las siguientes expresiones de punto flotante:

- a. $3.0 + 4.0 * 6.0$
- b. $3.0 * 4.0 / 6.0 + 6.0$
- c. $2.0 * 3.0 / 12.0 * 8.0 / 4.0$
- d. $10.0 * (1.0 + 7.0 * 3.0)$
- e. $20.0 - 2.0 / 6.0 + 3.0$
- f. $20.0 - 2.0 / (6.0 + 3.0)$
- g. $(20.0 - 2.0) / 6.0 + 3.0$
- h. $(20.0 - 2.0) / (6.0 + 3.0)$



Actividad 17:

Suponga que num almacena el valor entero 1, m almacena el valor entero 50, n almacena el valor entero 10 y p almacena el valor entero 5. Evalúe las siguientes expresiones:

- a. $n/p+3$
- b. $m/p+n-10* \text{ num}$
- c. $m-3*n+4*\text{num}$
- d. $\text{núm}/5$
- e. $18/p$
- f. $-p*n$
- g. $-m/20$
- h. $(m+n)/(p+\text{num})$
- i. $m+n/p+\text{num}$



Actividad 18:

Determine la salida de los siguientes programas:

- a)

```
print('Respuesta1 es el entero',9/4)
print('Respuesta1 es el entero',17/3)
```
- b)

```
print('El residuo de 9 dividido entre 4 es:',9%4)
print('El residuo de 17 dividido entre 3:',17%3)
```

Precedencia de Operadores



Actividad 19:

Resuelve los ejercicios en papel y luego escribe un programa que imprima por consola el resultado. ¿Obtuviste el mismo resultado?

- a,b=10,7
- c=a + 10 - 5 * b + 4
- d=3 * a + 1 - b // 4 + a % 2
- e=2 * a + b - 1 + a ** 2

Uso de paréntesis para el planteo en Python de distintas expresiones aritméticas comunes.



Actividad 20:

FÓRMULA GENERAL	EXPRESIONES EN PYTHON
$p = \frac{c1+c2+c3}{3}$	
$s = \frac{n+(n+1)}{2}$	
$h = \sqrt{a^2 + b^2}$	
$p = \frac{y2 - y1}{x2 - x1}$	

Ejercicios integradores



Actividad 22:

Escribe un programa generador de cuentas de correo electrónico para estudiantes de la provincia de Tierra del Fuego, usted debe pedir al usuario que ingrese su apellido y su nombre, se mostrará por pantalla el siguiente formato: apellido_nombre@tdf.edu.ar.

Tener en cuenta que la cuenta debe ser en minúscula.

Ayudita: El método `lower()` devuelve una cadena donde todos los caracteres están en minúsculas. Sintaxis: **`print(txt.lower())`**



Actividad 23:

Un vehículo parte de la ciudad de Córdoba y se dirige a Rosario, por autopista. La distancia aproximada entre ambas ciudades es de 400 km, el vehículo se desplaza a una velocidad promedio de 122 km/h.

Desarrolle un programa que calcule el tiempo total en horas que demorara ese vehículo en llegar a Rosario, no es necesario convertir a horas, minutos y segundos, exprese el resultado como un número real.



Actividad 24:

Nos solicitan un programa donde el usuario pueda inscribirse a una capacitación laboral, el mismo debe solicitar: nombre y apellido, dni, edad, teléfono y ciudad..

La salida por consola será la siguiente:

Bienvenido APELLIDO, Nombre al Curso de Diseño Web!

Tus datos son:

DNI:

EDAD:

TELÉFONO:

CIUDAD:

Tener en cuenta que la ciudad debe tener la primera letra en mayúscula cuando salga por consola.

Ayudita: El método `capitalize()` devuelve una cadena donde el primer carácter está en mayúsculas y el resto en minúsculas.

Sintaxis: **`print(txt.capitalize())`**







Diagrama de flujo

Es una de las **técnicas de representación de algoritmos** más antigua y a la vez la más utilizada. Un **diagrama de flujo** es un diagrama que utiliza los símbolos (cajas) estándar y que los pasos del algoritmo son escritos en el interior, donde son unidos mediante flechas, denominadas líneas de flujo, que indican la secuencia en que se deben ejecutar. Los símbolos estándar normalizados son muy variados, sin embargo, los símbolos más utilizados representan:

- ♦ Proceso

- ♦ Decisión
- ♦ Fin
- ♦ Entrada / Salida
- ♦ Dirección del flujo

La siguiente tabla presenta los símbolos más utilizados en la confección de los diagramas de flujo.

Símbolo	Función
	Terminal representa el comienzo y el final de un programa
	Entrada/Salida , cualquier tipo de introducción de datos en la memoria desde los periféricos
	Proceso , cualquier tipo de operación que pueda originar cambio de valor, operaciones aritméticas, transferencia
	Decisión , indica operaciones lógicas o de comparación entre datos y en función del resultado de la misma determina cuál de los distintos caminos alternativos del programa se debe seguir; normalmente tiene dos salidas
flechas	Indicador de dirección o línea de flujo , indica el sentido de ejecución de las operaciones
	Impresora , se utiliza en ocasiones en lugar del símbolo de E/S
	Subrutina , indica comienzo de una subrutina o función

Reglas Básicas para la construcción del Diagrama de Flujo:

1. Todo diagrama debe tener un inicio y un fin
2. Las líneas de conexión o de flujo deben ser siempre rectas, si es posible verticales y horizontales nunca cruzadas o inclinadas; para conseguir lo anterior es necesario apoyarse en conectores.
3. Las líneas que enlazan los símbolos entre sí deben estar todas conectadas.
4. Se deben dibujar todos los símbolos de modo que se pueda seguir el proceso visualmente de arriba hacia abajo (diseño de top-down) y de izquierda a derecha.
5. Evitar la terminología de un lenguaje de programación o máquina.
6. Se deben inicializar las variables que se utilicen o permitir la asignación de valores mediante consulta al usuario.
7. Todo el Diagrama debe ser claro, ordenado y fácil de recorrer;

8. El Diagrama se debe probar recorriéndolo con datos iniciales simples (prueba de escritorio).

Diseño Top Down

Significa descomponer un programa en términos de recursos abstractos, es decir consiste en descomponer una determinada acción compleja en términos de un número de acciones más simples capaz de ejecutarlas o constituyan instrucciones de computadora disponible.

Un programa estructurado cumple las siguientes características:

- ◊ Posee un solo punto de entrada y uno de salida o Fin para control del programa.
- ◊ Existen caminos desde la entrada hasta la salida que se pueden seguir y que pasan por todas partes del programa.
- ◊ Todas las instrucciones son ejecutables y no existen lazos o bucles infinitos (sin fin); son Finitas.

Los Algoritmos son independientes tanto del lenguaje de programación en que se expresan como de la computadora que los ejecuta.

En cada problema el algoritmo se puede expresar en un lenguaje diferente de programación y ejecutarse en una computadora distinta; sin embargo, el algoritmo será siempre el mismo.

Así por ejemplo, en una analogía con la vida diaria, una receta de un plato de cocina se puede expresar en español, inglés o francés, pero cualquiera que sea el lenguaje, los pasos para la elaboración del plato se realizan sin importar el cocinero.



Actividad 25:

Completa los siguientes diagrama de flujo:

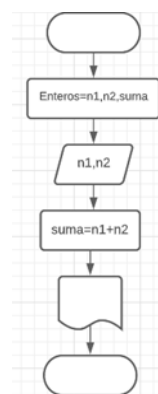
a- Se requiere calcular la suma de dos números.

Se pide generar la siguiente Salida Impresa:

- El resultado de la suma de los dos números.

Para ello Ud. dispone de las siguientes Entradas:

- Número 1 (n1): identifica el primer número.
- Número 2 (n2): identifica el segundo número.



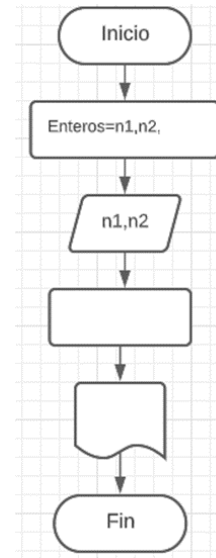
b-Se requiere calcular el producto de dos números.

Se pide generar la siguiente Salida Impresa:

- El resultado del producto de los dos números.

Para ello Ud. dispone de las siguientes Entradas:

- Número 1 (n1): identifica el primer número,
- Número 2 (n2): identifica el segundo número.



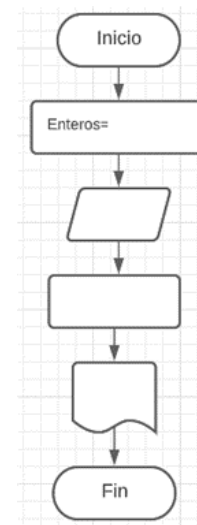
c-Se necesita averiguar el perímetro de un cuadrado.

Se pide generar las siguiente Salida Impresa:

- El resultado del perímetro del cuadrado.

Para ello Ud. dispone de la siguiente Entrada:

- Lado1 (l1): identifica el lado, expresado en centímetros.



d-Una persona necesita obtener información relacionada con el desempeño de su automóvil. Se pide generar las siguientes

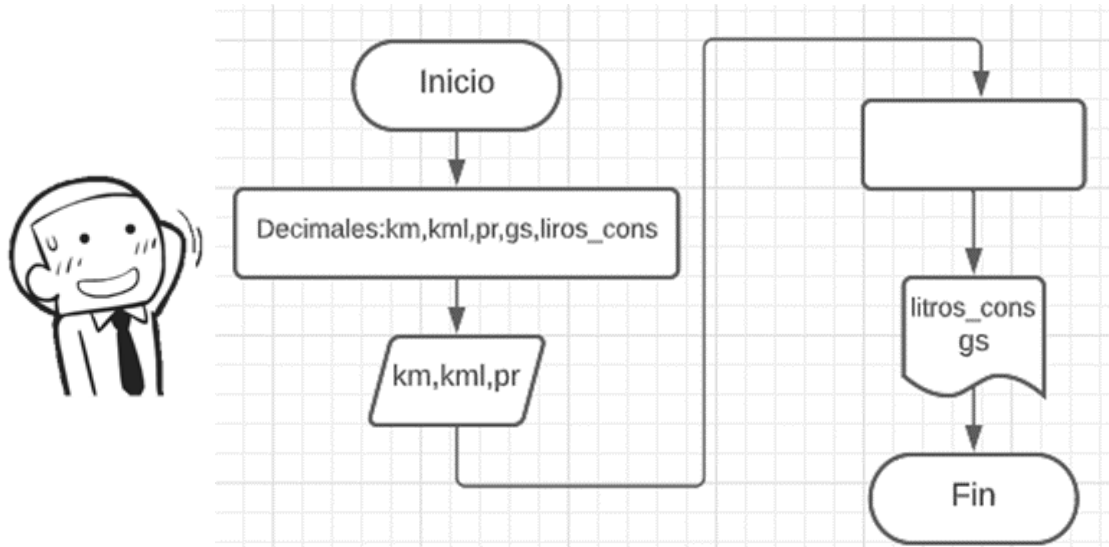
Salidas Impresas:

- La cantidad de litros consumidos.
- El importe gastado en combustible.

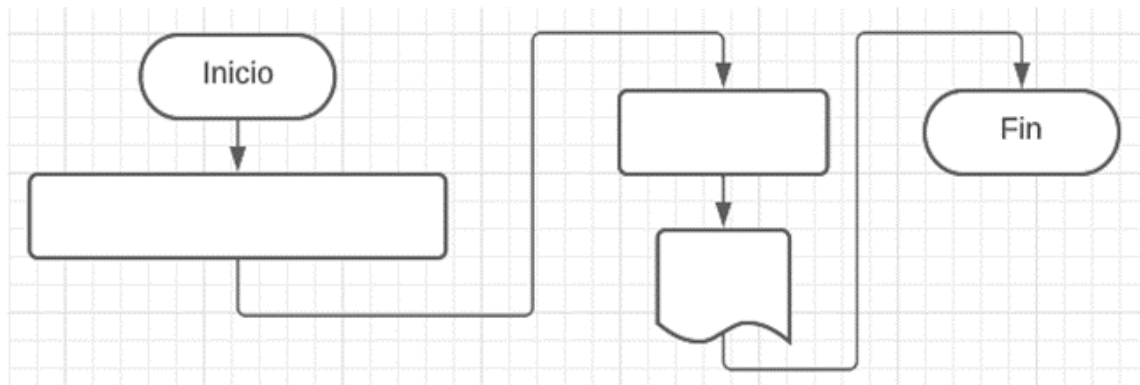
Para ello Ud. dispone de las siguientes Entradas:

- Kilómetros (km): representa los Km recorridos por el vehículo.
- Precio (pr): representa el precio del combustible por litro.
- Kilómetros Litros (kmL): representa los km recorridos por cada litro.





e-Desarrollar un algoritmo que declare dos variables enteras, le asigne valores arbitrarios y luego muestre su suma, diferencia y producto.



Programación Estructurada

El paradigma de la programación estructurada plantea que todo problema puede resolverse con la combinación de tres estructuras básicas:

☺ **Estructura secuencial:** En esta estructura todas las instrucciones son ejecutadas en forma secuencial y obligatoria, sin posibilidad de repetir o de seleccionar la ejecución o no de cada instrucción.

☺ **Estructura condicional:** Esta estructura permite identificar mediante una condición si una instrucción o conjunto de instrucciones deben ser

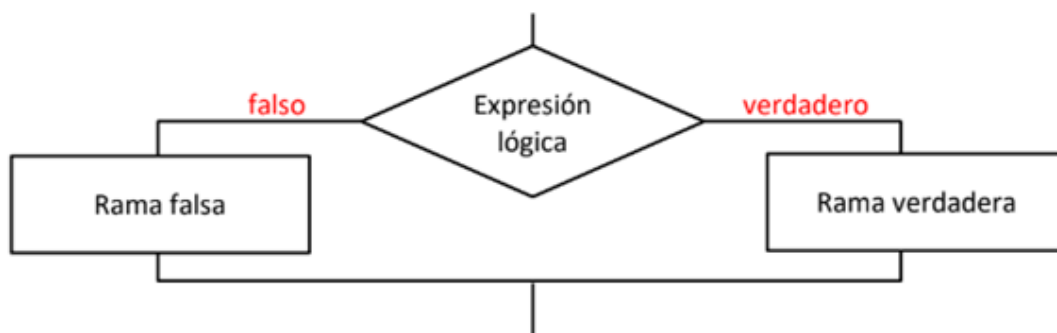
ejecutadas. Por ejemplo, si se requiere calcular el cociente entre dos números, es necesario verificar que el denominador sea distinto de 0 y únicamente en ese caso realizar el cálculo.

☺ **Estructura repetitiva:** Las estructuras repetitivas permiten repetir la ejecución de una instrucción o conjunto de instrucciones, estableciendo un ciclo o bucle que se ve interrumpido al alcanzar una cierta cantidad de repeticiones o frente a una condición que indique que deben finalizarse las repeticiones.

Estructura condicional

Una instrucción condicional contiene una **expresión lógica** que puede ser **evaluada** por **verdadera** o por **falsa**, y dos bloques de instrucciones adicionales designados en general como la salida o rama verdadera y la salida o rama falsa. Si un programa alcanza una instrucción condicional y en ese momento la expresión lógica es verdadera, el programa ejecutará las instrucciones de la rama verdadera (y sólo esas). Pero si la expresión es falsa, el programa ejecutará las instrucciones de la rama falsa (y sólo esas).

Diagrama general de una instrucción condicional típica.



Sintaxis:

if (expresión lógica):
 instrucciones de la rama verdadera
else:
 instrucciones de la rama falsa



Actividad 26:

Cargar por teclado dos números enteros. Mostrarlos ordenados de menor a mayor.

- a) Identifica los componentes.
- b) Realiza el diagrama de flujo.
- c) Codifica la solución.



Actividad 27:

Se necesita desarrollar un programa que permita calcular la suma de tres números. Si el resultado es mayor a 10 dividir por 2 (mostrar el resultado sin números decimales), caso contrario elevar el resultado al cubo.

- a) Identifica los componentes.
- b) Realiza el diagrama de flujo.
- c) Codifica la solución.



Actividad 28:

Se necesita desarrollar un programa para el área de recursos humanos de una empresa que permita informar el diario de un determinado operario. Usted deberá cargar por teclado el código de turno que el operario trabajó ese día (1 representa Diurno y 2 representa Nocturno) y la cantidad de horas trabajadas.

La política de trabajo en la empresa es que los operarios de la misma pueden trabajar en el turno diurno o nocturno. Si un operario trabaja en el turno nocturno el pago es de 8057.75 la hora, si lo hace en el turno diurno cobra 5234 pesos la hora.

- d) Identifica los componentes.
- e) Realiza el diagrama de flujo.
- f) Codifica la solución.

Expresiones lógicas, operadores relacionales y conectores lógicos

Una expresión lógica es una expresión cuyo resultado esperado es un valor de verdad (True o False).

Las instrucciones condicionales se basan en chequear el valor de una expresión lógica para determinar el camino que seguirá el programa en su ejecución.

Para el planteo de expresiones lógicas, todo lenguaje de programación provee operadores que implican la obtención de un valor de verdad como resultado.

Operador	Significado	Ejemplo	Observaciones
==	igual que	<code>a == b</code>	retorna <i>True</i> si <i>a</i> es igual que <i>b</i> , o <i>False</i> en caso contrario
!=	distinto de	<code>a != b</code>	retorna <i>True</i> si <i>a</i> es distinto de <i>b</i> , o <i>False</i> en caso contrario
>	mayor que	<code>a > b</code>	retorna <i>True</i> si <i>a</i> es mayor que <i>b</i> , o <i>False</i> en caso contrario
<	menor que	<code>a < b</code>	retorna <i>True</i> si <i>a</i> es menor que <i>b</i> , o <i>False</i> en caso contrario
>=	mayor o igual que	<code>a >= b</code>	retorna <i>True</i> si <i>a</i> es mayor o igual que <i>b</i> , o <i>False</i> en caso contrario
<=	menor o igual que	<code>a <= b</code>	retorna <i>True</i> si <i>a</i> es menor o igual que <i>b</i> , o <i>False</i> en caso contrario

Ejemplos:

```

cad1 = 'Hola'
cad2 = 'Mundo'
if cad1 == cad2:
    print('Son iguales')
else:
    print('No son iguales')
if cad1 != 'Hello':
    print('No es la palabra Hello...')
else:
    print('Es la palabra Hello...')
  
```



Actividad 29:

Se ingresan las medidas de frente y fondo de un terreno. Determinar si es cuadrado o rectangular y calcular su perímetro y superficie.

- a) Identificación de los componentes
- b) Diagrama de flujo
- c) Codifica la solución

Conectores Lógicos

Para realizar comprobaciones múltiples se debe aplicar los conectores lógicos.

Tabla de conectores lógicos en Python.

Operador	Significado	Ejemplo
and	conjunción lógica (y)	<code>a == b and y != x</code>
or	disyunción lógica (o)	<code>n == 1 or n == 2</code>
not	negación lógica (no)	<code>not x > 7</code>

Un conector lógico u operador booleano es un operador que permite encadenar la comprobación de dos o más expresiones lógicas y obtener un resultado único.

Tablas de verdad

Tabla de verdad del conector and		
p	q	p and q
True	True	True
True	False	False
False	True	False
False	False	False

```
if sueldo > 15000 and antigüedad >= 10:
    print('Crédito concedido')
else:
    print('Crédito rechazado')
```

Tabla de verdad del conector or		
p	q	p or q
True	True	True
True	False	True
False	True	True
False	False	False

```
if opcion == 1 or opcion == 3 or opcion == 5:
    print('Opción correcta')
else:
    print('Opción incorrecta')
```

Tabla de verdad del conector not	
p	not p
True	False
False	True

```
if not edad < 18:
    print('Puede acceder al permiso')
else:
    print('No puede acceder al permiso')
```

```
if edad >= 18:
    print('Puede acceder al permiso')
else:
    print('No puede acceder al permiso')
```

Precedencia de ejecución de los operadores relacionales y de los conectores

precedencia(conectores) < precedencia(relacionales) < precedencia(matemáticos)

Los operadores relacionales o de comparación tienen la misma prioridad todos ellos, y serán aplicados de izquierda a derecha.

if a > b > c: es lo mismo if a > b and b > c:

Los conectores and y or actúan de forma cortocircuitada, lo cual quiere decir que dependiendo del valor de la primera proposición evaluada, la segunda (o las restantes a partir de ella) podrían no llegar a ser evaluadas.

Conector	Ejemplo	Efecto del cortocircuito
and	if a > b and a > c:	Si la primera proposición es <i>False</i> (a > b en este caso) la segunda (a > c) <i>no se chequea</i> y la salida también es <i>False</i> .
or	if n < 0 or n > 9:	Si la primera proposición es <i>True</i> (n < 0 en este caso), la segunda (n > 9) <i>no se chequea</i> y la salida también es <i>True</i> .



Actividad 30:

Cargar por teclado tres números enteros que se supone representan las edades de tres personas. Determinar si alguno de los valores cargados es negativo, en cuyo caso informe en pantalla con un mensaje tal como: "Alguna es incorrecta: negativo". Si todos los valores son positivos o cero, informe que todos son correctos.

- Identificación de componentes
- Diagrama de flujo
- Codifica la solución

Tipos estructurados (o compuestos) básicos en Python

Aquellos tipos de datos que solo admiten que una variable pueda contener un único valor de ese tipo, se llaman tipos simples.

Se representa como un recipiente con único compartimiento para almacenar un único elemento:



Cuando en una misma variable, contiene varios valores al mismo tiempo, se denomina como tipos compuestos, estructurados o como una estructura de datos:



Una secuencia de datos inmutable es simplemente un conjunto finito de datos cuyos valores y orden de aparición no pueden modificarse una vez asignados.

Las secuencias de datos mutables son aquellas secuencias en las que los valores individuales pueden modificarse luego de ser asignados.



Tupla

Una tupla es una secuencia de datos que pueden ser de tipos diferentes.

Para definir una tupla "t" que contenga ninguno, uno o varios valores iniciales:

- a) Usando un par de paréntesis vacíos para denotar una tupla vacía:

$t = ()$

- b) Usando una coma al final, para denotar una tupla con un solo valor:

$t = (a,)$ $t = a,$

- c) Separando los ítems con comas:

$t = a,b,c$ $t = (a,b,c)$

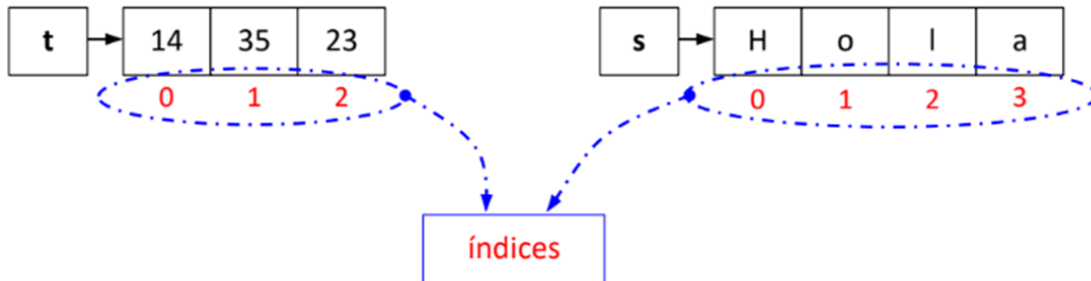
- d) Usando la función predefinida tuple()

$t = \text{tuple}(\text{iterable})$ $t = \text{tuple}(\text{'Hola Mundo'})$

Donde iterable es cualquier secuencia que pueda recorrer con un ciclo for iterator, como una cadena, otra tupla, etc

Secuencia de índices

La secuencia de índices identificar a los casilleros de una secuencia de datos que comienza desde 0(Cero) y sigue luego en forma correlativa.



Para acceder a un elemento individual de una secuencia, basta con escribir el identificador de la misma y luego agregar entre corchetes el índice del casillero a acceder:

```
t = 14, 35, 23  
print(t[1])  
x = t[2]  
print(x)
```



Actividad 31:

La dirección de la carrera de ingeniería en sistemas de la UTN de Córdoba necesita un programa que permite cargar el nombre del estudiante de quinto año, el nombre del profesor responsable de la práctica profesional supervisada de ese estudiante, y el promedio general (con decimales) del estudiante en su carrera. Una vez cargados los datos, se pide simplemente mostrarlos en la consola de salida a modo de verificación, pero de forma que el nombre del practicante vaya precedido de la cadena "est." y el nombre del profesor supervisor se proceda con "prof.". El promedio del alumno debe mostrarse redondeado, sin decimales, en formato entero.

- Identificación de componentes
- Diagrama de flujo
- Codifica la solución



Funciones comunes de la librería estándar de Python

Función	Descripción	Ejemplo de uso
abs(x)	Retorna el valor absoluto del parámetro x .	<code>x = -23</code> <code>y = abs(x)</code> <code>print(y)</code> # muestra: 23
bin(x)	Obtiene la conversión a binario (base 2) del valor x , en formato de cadena cuyos dos primeros caracteres son '0b', indicando que lo que sigue es una secuencia en binario.	<code>x = 8</code> <code>s = bin(x)</code> <code>print(s)</code> # muestra: 0b1000
chr(i)	Retorna el caracter (en formato de string) que corresponde al valor Unicode número i .	<code>i = 65</code> <code>c = chr(i)</code> <code>print(c)</code> # muestra: A
divmod(a, b)	Retorna el cociente y el resto de la división entera entre los parámetros a y b .	<code>a, b = 14, 4</code> <code>c, r = divmod(a,b)</code> <code>print(c, r)</code> # muestra: 3 2
float(x)	Convierte la cadena s a un número en coma flotante. Hemos usado esta función combinada con la función <code>input()</code> para cargar números flotantes desde teclado.	<code>s = '23.45'</code> <code>y = float(s)</code> <code>print(y)</code> # muestra: 23.45
hex(x)	Obtiene la conversión a hexadecimal (base 16) del valor x , en formato de cadena cuyos dos primeros caracteres son '0x', indicando que lo que sigue es una secuencia en hexadecimal.	<code>x = 124</code> <code>s = hex(x)</code> <code>print(s)</code> # muestra: 0x7c
input(p)	Obtiene una cadena de caracteres desde la entrada estándar y la retorna. La cadena p es visualizada a modo de "prompt" mientras se espera la carga.	
int(x)	Convierte la cadena s a un número entero. Hemos usado esta función combinada con la función <code>input()</code> para cargar números enteros desde teclado.	<code>s = '1362'</code> <code>y = int(s)</code> <code>print(y)</code> # muestra: 1362
len(s)	Retorna la longitud del objeto t tomado como parámetro (un string, una lista, una tupla o un diccionario). La longitud es la <i>cantidad de elementos</i> que tiene el objeto t .	<code>t = 12, 45, 73</code> <code>n = len(t)</code> <code>print(n)</code> # muestra: 3
max(a, b, *args)	Retorna el mayor de los parámetros tomados. Notar que admite una <i>cantidad arbitraria</i> de parámetros: max(a,b) retorna el mayor entre a y b , pero max(a,b,c,d) retorna el mayor entre a, b, c y d .	<code>a, b = 6, 3</code> <code>my = max(a, b)</code> <code>print(my)</code> # muestra: 6
min(a, b, *args)	Retorna el menor de los parámetros tomados. Notar que admite una <i>cantidad arbitraria</i> de parámetros: min(a,b) retorna el menor entre a y b , pero min(a,b,c,d) retorna el menor entre a, b, c y d .	<code>a, b = 6, 3</code> <code>mn = min(a, b)</code> <code>print(mn)</code> # muestra: 3
min(a, b, *args)	Retorna el menor de los parámetros tomados. Notar que admite una <i>cantidad arbitraria</i> de parámetros: min(a,b) retorna el menor entre a y b , pero min(a,b,c,d) retorna el menor entre a, b, c y d .	<code>a, b = 6, 3</code> <code>mn = min(a, b)</code> <code>print(mn)</code> # muestra: 3
oct(x)	Obtiene la conversión a octal (base 8) del valor x , en formato de cadena cuyos dos primeros caracteres son '0o', indicando que lo que sigue es una secuencia en octal.	<code>x = 124</code> <code>s = oct(x)</code> <code>print(s)</code> # muestra: 0o174
ord(c)	Retorna el entero Unicode que representa al caracter c tomado como parámetro.	<code>c = 'A'</code> <code>i = ord(c)</code> <code>print(i)</code> # muestra: 65

pow(x, y)	Retorna el valor de x elevado a la y . Note que lo mismo puede hacerse con el operador "doble asterisco" (**). En Python: r = pow(x, y) es lo mismo que r = x**y .	x, y = 2, 3 r = pow(x, y) print(r) # muestra: 8
round(x, n)	Retorna el número flotante x , pero redondeado a n dígitos a la derecha del punto decimal. Si n se omite retorna la parte entera de x (como int). Si n es cero, retorna la parte entera de x (pero como float): round(3.1415) retornará 3 (int), pero round(3.1415, 0) retornará 3.0 (float).	x = 4.1485 r = round(x, 2) print(r) # muestra: 4.15
str(x)	Retorna una versión en forma de cadena de caracteres del objeto t (es decir, retorna la <i>conversión a string</i> de t).	t = 2, 4, 1 s = str(t) print(s) # muestra: (2, 4, 1)



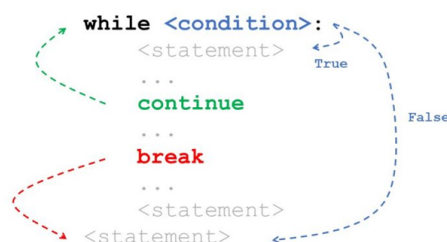
Actividad 32:

Realiza un programa que calcule la mayor edad entre dos usuarios. Utilizando las funciones de Python.

- Identificación de los componentes
- Diagrama de flujo
- Codifica la solución

Ciclo While

Un ciclo o bucle while es una estructura de control de flujo en la programación que se utiliza para repetir un bloque de código, mientras que se cumpla una condición determinada.



En esta estructura, la condición es una expresión booleana que se evalúa en cada iteración del ciclo.

Si la condición es verdadera, el bloque de código dentro del ciclo se ejecuta una y otra vez. Si es falsa, el ciclo se detiene y la ejecución continúa después del ciclo.

Es importante tener en cuenta que si la condición nunca llega a ser falsa, el ciclo continuará ejecutándose en un bucle infinito, lo que puede provocar problemas.

Por lo tanto, es esencial asegurarse de que la condición eventualmente se vuelve falsa para evitar esta situación.



Actividad 33:

Crear una variable e inicializarla con el valor 0, imprimirla siempre que la variable sea menor que 6.



Actividad 34:

Imprimir por pantalla los números desde el 1 hasta el 10



Actividad 35:

Escribir un programa que pregunte una y otra vez si desea continuar con el programa, siempre que se conteste exactamente sí (en minúscula y con tilde).



Actividad 36:

Escriba un programa que solicite una contraseña (el texto de la contraseña no es importante) y la vuelva solicitar hasta que las dos contraseñas coincidan.



Actividad 37:

Leer números enteros del teclado, hasta que el usuario ingrese 0. Finalmente, mostrar la sumatoria de todos los números ingresados.



Actividad 38:

Leer números enteros de teclado, hasta que el usuario ingrese el 0. Finalmente mostrar la sumatoria de todos los números positivos ingresados.

Ciclo For

Permite ejecutar en forma repetida un bloque de acciones durante un número determinado de veces.

```
1 nombres=('Alejandro','Augusto', 'Hugo')
2 for nom in nombres:
3     print (nom)
4
```

La operación de recorrer la estructura y procesar de a uno de sus elementos se llama iterar la estructura, la variable que se use (nom) va tomando cada uno de los valores a medida que se itera sobre ella.

```
ciudad='Córdoba'
for c in ciudad:
    print (c)
```

El for usa la variable iteradora c para ir almacenando una copia de los caracteres de la cadena, a razón de uno por vuelta del ciclo y en el orden que aparecen.



Actividad 39:

Suma de números: Escribe un programa que calcule la suma de todos los números del 1 al 10 e imprima el resultado.



Actividad 40:

Imprimir elementos de una lista: Carga una tupla con 5 herramientas y luego imprime una debajo de la otra.



Actividad 41:

Escribir un programa que cuente y muestre la cantidad de vocales (a,e,i,o,u) en una palabra dada por el usuario.



Actividad 42:

imprimir los números entre el 5 y el 20, saltando de tres en tres.



Actividad 43:

Requerir al usuario que ingrese un número entero positivo e imprimir todos los números correlativos entre el ingresado por el usuario y uno menos del doble del mismo.

Funciones

Una función es una secuencia de instrucciones con nombre, que lleva a cabo la operación deseada. Esta operación se especifica en una definición de función.

Sintaxis:

```
def nombre_funcion (lista de parametros):  
    instrucciones
```

Crear una nueva función le da la oportunidad de dar nombres a un grupo de sentencias. Las funciones simplifican su programa al ocultar cálculos complejos detrás de órdenes sencillas, y usar palabras de su propia lengua en vez de código.

Crear una nueva función hace que el programa sea más pequeño, al eliminar código repetitivo.

Una función, no es ejecutada hasta que no sea invocada. Para invocar una función, simplemente se la llama por su nombre.

```
def mensaje ():  
    print('hola Mundo')  
  
mensaje()
```

Cuando una función, haga un retorno de datos, pueden ser asignados a una variable.

```
def mensajeRetorno():  
    return 'Mensaje de la funcion'  
  
imprimir= mensajeRetorno()  
print(imprimir)
```

Parámetros

Un parámetro es un valor que la función espera recibir cuando sea llamada (invocada), a fin de ejecutar acciones en base al mismo. Una función puede esperar uno o más parámetros (que irán separados por una coma) o ninguno.

```
def nom_ape (nombre, apellido):  
    nombre_completo= nombre + ' ' + apellido  
    print(nombre_completo)  
nom_ape('Agos', 'Noriega')
```

Al llamar a una función, siempre se le deben pasar sus argumentos en el mismo orden en el que se espera.

Variables Locales

Los argumentos son los valores que controlan como la función lleva a cabo su tarea.

```
def suma(n1,n2):  
    resultado=n1+n2  
    print('El resultado de la suma es: ', resultado)  
  
suma (6,2)
```

Flujo de ejecución:

La ejecución comienza siempre por la primera sentencia del programa.

Las sentencias se ejecutan a razón de una cada vez, en orden, hasta que se alcanza una llamada a una función.

Las definiciones de funciones no alteran el flujo de ejecución del programa, pero recuerde que las sentencias que hay dentro de la función no se ejecutan hasta que se hace la llamada a la misma.

Aunque no es habitual, puede definir una función dentro de otra. En este caso, la definición de función interior no se ejecuta hasta que no se llama a la función exterior.