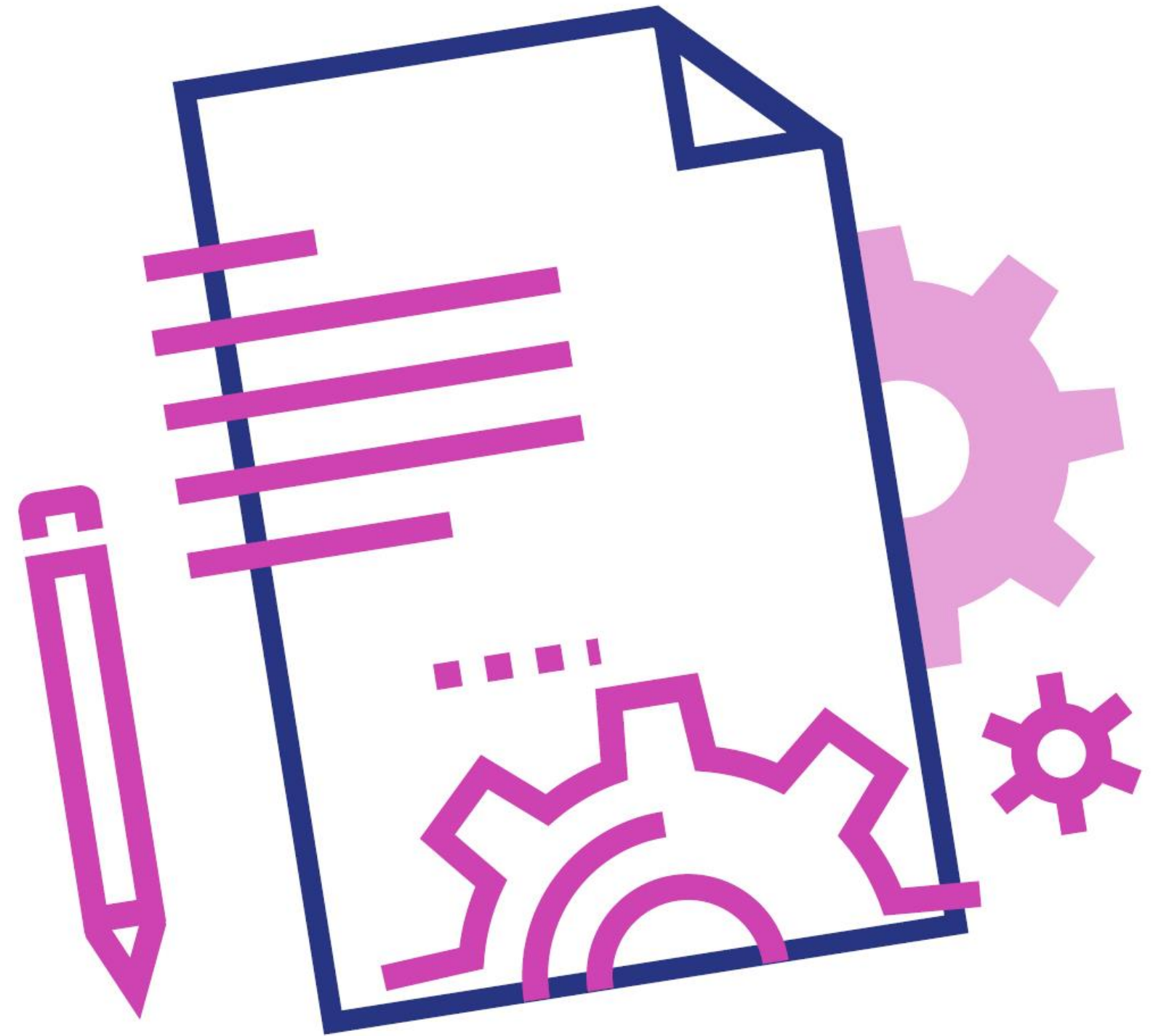


Estructuras de datos y funciones en Python

Módulo 3: Instalación y configuración de redes.



Telecomunicaciones



Perfil de Egreso - Objetivos de Aprendizaje de la Especialidad

Módulo 1	<p>OA1 Leer y utilizar esquemas, proyectos y en general todo el lenguaje simbólico asociado a las operaciones de montaje y mantenimiento de redes de telecomunicaciones.</p>	Módulo 6	<p>OA8 Instalar y configurar una red inalámbrica según tecnologías y protocolos establecidos.</p> <p>OA7 Aplicar la normativa y los implementos de seguridad y protección relativos al montaje y el mantenimiento de las instalaciones de telecomunicaciones y la normativa del medio ambiente.</p>
Módulo 2	<p>OA6 Realizar mantenimiento y reparaciones menores en equipos y sistemas de telecomunicaciones, utilizando herramientas y pautas de mantención establecidas por el fabricante.</p> <p>OA7 Aplicar la normativa y los implementos de seguridad y protección relativos al montaje y el mantenimiento de las instalaciones de telecomunicaciones y la normativa del medio ambiente.</p>	Módulo 7	<p>OA5 Instalar y configurar una red de telefonía (tradicional o IP) en una organización según los parámetros técnicos establecidos.</p>
Módulo 3	<p>OA2 Instalar equipos y sistemas de telecomunicaciones de generación, transmisión, repetición, amplificación, recepción, y distribución de señal de voz, imagen y datos, según solicitud de trabajo y especificaciones técnicas del proyecto.</p> <p>OA10 Determinar los equipos y sistemas de comunicación necesarios para una conectividad efectiva y eficiente, de acuerdo a los requerimientos de los usuarios.</p>	Módulo 8	<p>OA3 Instalar y/o configurar sistemas operativos en computadores o servidores con el fin de incorporarlos a una red LAN, cumpliendo con los estándares de calidad y seguridad establecidos.</p>
Módulo 4	<p>OA9 Detectar y corregir fallas en circuitos de corriente continua de acuerdo a los requerimientos técnicos y de seguridad establecidos.</p>	Módulo 9	<p>OA10 Determinar los equipos y sistemas de comunicación necesarios para una conectividad efectiva y eficiente, de acuerdo, a los requerimientos de los usuarios.</p> <p>OA6 Realizar el mantenimiento y reparaciones menores en equipos y sistemas de telecomunicaciones, utilizando herramientas y pautas de mantención establecidas por el fabricante.</p>
Módulo 5	<p>OA2 Instalar equipos y sistemas de telecomunicaciones de generación, transmisión, repetición, amplificación, recepción y distribución de señal de voz, imagen y datos, según solicitud de trabajo y especificaciones técnicas del proyecto.</p> <p>OA4 Realizar medidas y pruebas de conexión y de continuidad de señal eléctrica, de voz, imagen y datos- en equipos, sistemas y de redes de telecomunicaciones, utilizando instrumentos de medición y certificación de calidad de la señal autorizada por la normativa vigente.</p>	Módulo 10	<p>No está asociado a Objetivos de Aprendizaje de la Especialidad (AOE), sino a genéricos. No obstante, puede asociarse a un OAE como estrategia didáctica.</p>



Perfil de Egreso – Objetivos de Aprendizaje Genéricos

<p>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.</p>	<p>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.</p>	<p>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.</p>
<p>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.</p>	<p>E- Tratar con respeto a subordinados, superiores, colegas, clientes, personas con discapacidades, sin hacer distinciones de género, de clase social, de etnias u otras.</p>	<p>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.</p>
<p>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.</p>	<p>H- Manejar tecnologías de la información y comunicación para obtener y procesar información pertinente al trabajo, así como para comunicar resultados, instrucciones e ideas.</p>	<p>I- Utilizar eficientemente los insumos para los procesos productivos y disponer cuidadosamente los desechos, en una perspectiva de eficiencia energética y cuidado ambiental.</p>
<p>J- Emprender iniciativas útiles en los lugares de trabajo y/o proyectos propios, aplicando principios básicos de gestión financiera y administración para generarles viabilidad.</p>	<p>K- Prevenir situaciones de riesgo y enfermedades ocupacionales, evaluando las condiciones del entorno del trabajo y utilizando los elementos de protección personal según la normativa correspondiente.</p>	<p>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.</p>



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 función.

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 acuerdo 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. Evalúa 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 específicos de su área y de las tendencias de desarrollo para el desempeño de sus actividades y funciones.



Metodología seleccionada

Demostración guiada

- Esta presentación te servirá para avanzar paso a paso en el desarrollo de la actividad propuesta..

Aprendizaje Esperado

- **AE4:** Diseñar programas de mediana complejidad, que involucren sentencias, estructuras y programación modular en Python para la solución de problemas, de acuerdo a los requerimientos de su especialidad y contexto laboral.



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

- **Diseñar** un programa en Python, utilizando estructuras de datos tipo lista y funciones en el desarrollo de aplicaciones cotidianas.



Contenidos

01 LISTAS <<

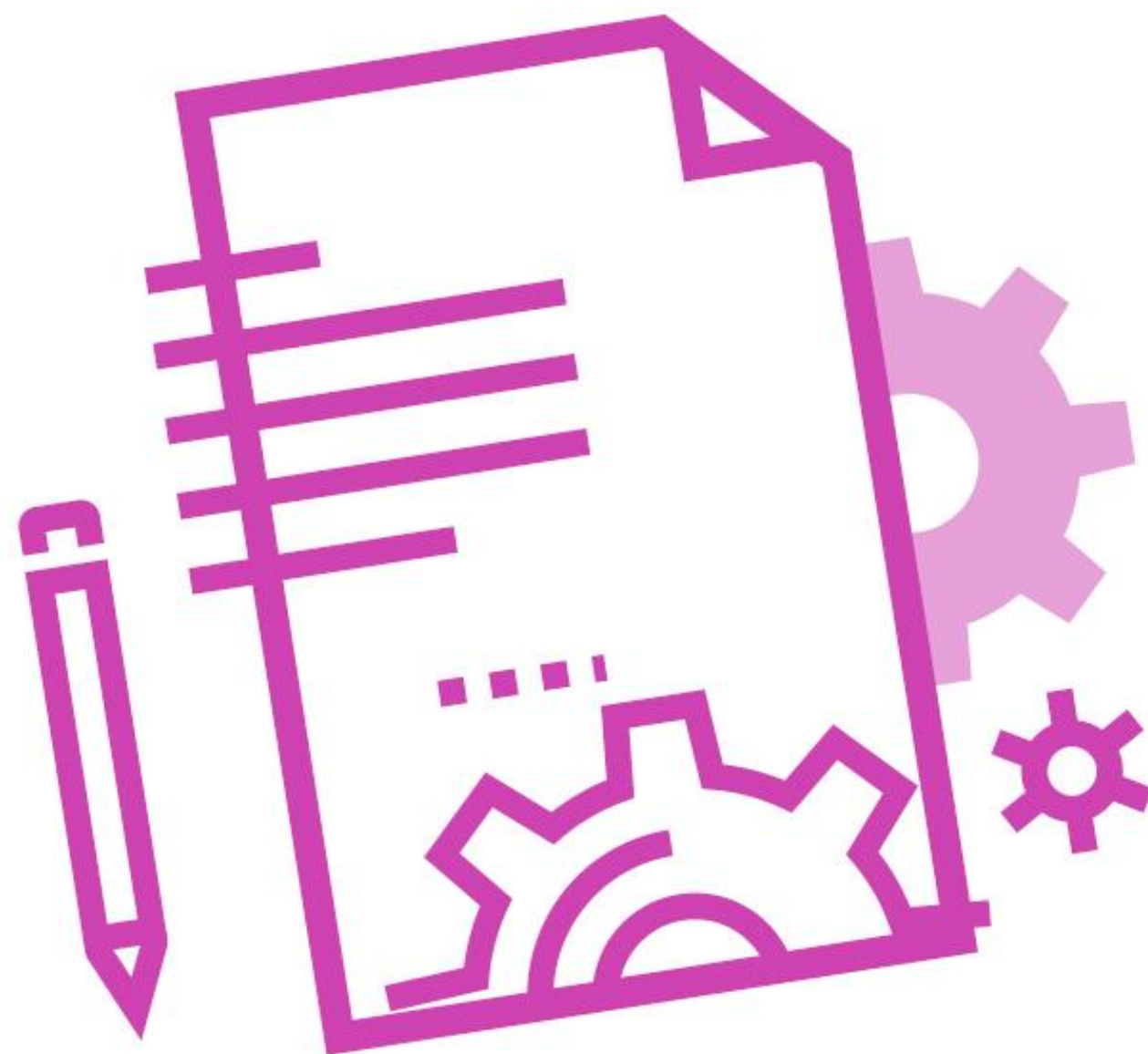
- ¿Qué son las estructuras de datos?
- Listas.
- Estructura de una lista.
- Creación de una lista.
- Elementos de una lista.
- Recorriendo una lista.
- Métodos de lista.

02 FUNCIONES

- Tipos de Funciones.
- Definición de Función.
- Ventajas de las funciones.
- Consideraciones generales.
- Sintaxis de definición.
- Llamada a una función.
- Parámetros y Argumentos.
- Tipos de argumentos.
- Las sentencia return.



LISTAS



Describe la imagen que estás viendo...

- ¿Para qué podría servir este elemento?

1°A



¿Qué son las estructuras de datos?

- Una “estructura de datos” es una **colección de valores**, los cuales están relacionados y se pueden realizar operaciones sobre ellos.
- En otras palabras se refiere organizar los datos y cómo administrarlos.
- Una estructura de datos implica el **formato** en que los valores van a ser almacenados, cómo van a ser accedidos y modificados.

- En Python existen varias estructuras de datos, cada una con sus características propias. Estas son las Listas, las Tuplas y los Diccionarios.
-



Listas

- Una de las estructuras de datos más utilizadas en Python son las LISTAS.
- Las “listas” son una estructura de datos que nos permite almacenar valores de distintos tipos, como números enteros, números reales, caracteres, cadenas, etc.
- Por ejemplo: se pueden almacenar para un alumno los siguientes datos en una lista: nombre, curso, edad, sexo, promedio, etc.

1°A

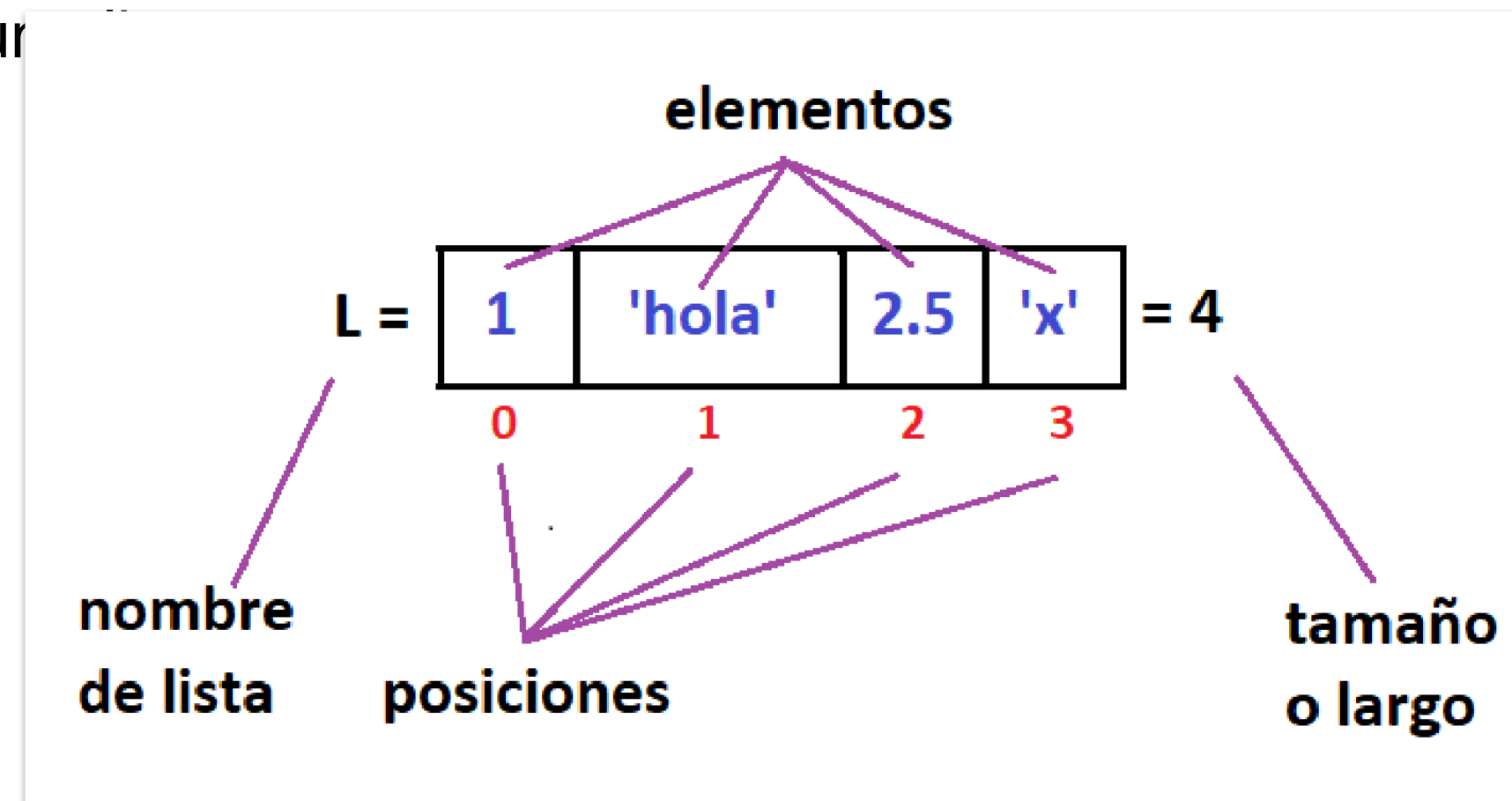


Estructura de una lista

- La idea de estructura de una lista, gráficamente se ve mas o menos así:



- Podemos identificar los elementos que componen una lista de la siguiente manera:



Estructura de una lista

● Donde,

- **NOMBRE DE LISTA:** es el nombre asignado a la lista por el cual la identificaremos en el programa.
- **ELEMENTOS:** son los valores almacenados en la lista, los cuales pueden ser de diferentes o del mismo tipo. Para referirse a un elemento de la lista se debe escribir el nombre de la lista seguido de la posición del elemento entre corchetes, ejemplo: L[1].
- **TAMAÑO O LARGO:** es la cantidad de elementos que están almacenados en la lista.
- **POSICIONES:** también llamados índices, se refiere a la numeración asignada a cada casillero de la lista. Esta numeración comienza en la posición 0 y termina en la posición LARGO-1, dependiendo de la cantidad de elementos que tenga. Por ejemplo: si la lista tiene 10 elementos, las posiciones irán desde 0 hasta 9.



Creando una lista

- Para utilizar una lista, ésta se debe definir. Una lista se define con un nombre y se le asignan corchetes vacíos, de la siguiente manera:

$$L = []$$

- Esto significa que hemos definido una lista llamada L que no tiene ningún elemento aún.

Para agregar elementos a la lista podemos asignarlos de la siguiente manera:

$$L = [1, 'hola', 2.5, 'x']$$


Elementos de una lista

- Para acceder a los elementos de una lista debemos hacerlo a través de su posición.
- Si tenemos la lista: $L = [1, 'hola', 2.5, 'x']$ podemos señalar que:
 - El nombre de la lista es: L.
 - La cantidad de elementos de la lista es: 4.
 - Las posiciones de los elementos son: 0, 1, 2, 3.
 - Los elementos son: L[0], L[1], L[2] y L[3].
- Usualmente para recorrer los elementos de una lista se utiliza un contador y un ciclo.



Recorriendo una lista

- El ciclo mas adecuado para recorrer una lista es el FOR, ya que no requiere ninguna condición más que un rango de números, que en este caso será el LARGO de la lista, o sea 4.
- También utilizaremos como contador la variable 'i' que tomará los valores desde 0 hasta el LARGO-1, o sea 3. Estos elementos en su conjunto se escriben así:

```
L=[1, 'hola', 2.5, 'x']  
for i in range(4):  
    print(L[i])
```

```
1  
hola  
2.5  
x
```



Recorriendo una lista

- En el ejemplo anterior se observa que al ejecutar la sentencia se muestran cada uno de los elementos de una lista, gracias al contador, que la recorre por completo.
- Existen elementos fundamentales para manipular o gestionar las listas. Estos elementos son llamados MÉTODOS, los cuales son funcionalidades asociadas a las listas y que permiten realizar una serie de acciones con ellas.

A continuación, conoceremos algunos de ellos.



Métodos de lista

- Las acciones que pueden realizarse con listas se llaman MÉTODOS de lista.
- Al generar una nueva lista y asignarla a una variable, adquiere una serie de atributos y métodos asociados.
- La forma general para llamar a un método que pertenece a un objeto es:

`objeto.metodo(argumentos)`

- Donde,
 - objeto: corresponde al nombre de la lista.
 - método: corresponde al nombre del método a usar.
 - argumentos: dependen del método que los llama.



Métodos de lista

- 01 APPEND:** Permite agregar elementos a una lista, los cuales serán ingresados por el usuario. Usualmente con esta sentencia se crea una lista.

```
L=[]  
for i in range(5):  
    L.append(input("Ingrese elemento: "))  
print("Lista --> ",L)
```

```
Ingrese elemento: 1  
Ingrese elemento: 2  
Ingrese elemento: 3  
Ingrese elemento: 4  
Ingrese elemento: 5  
Lista --> ['1', '2', '3', '4', '5']
```



Métodos de lista

02 LEN: Determina el largo de una lista, es decir, cuantos elementos tiene.

```
L=[1,2,3,4,5]  
print("Largo de la lista --> ",len(L))
```

```
Largo de la lista --> 5
```



Métodos de lista

03 NT: Determina la cantidad de veces que aparece un elemento en la lista.

```
L=[1,2,3,2,4,2]  
print("Cantidad de veces que aparece el 2 en la lista: ",L.count(2))
```

```
Cantidad de veces que aparece el 2 en la lista: 3
```



Métodos de lista

04 INSERT: Permite insertar un elemento en una posición específica de la lista. Los argumentos de la función son dos: la posición y el elemento a insertar.

```
L=[1,2,3,4]  
L.insert(2,100)  
print("L = ",L)
```

```
L = [1, 2, 100, 3, 4]
```



Métodos de lista

05 CLEAR: Permite vaciar una lista, es decir, elimina todos sus elementos.

```
L=[1,2,3,4]  
L.clear()  
print("L = ",L)
```

```
L = []
```



Métodos de lista

06 POP: Permite eliminar un elemento de la lista por posición, para lo cual se le debe indicar la posición del elemento a eliminar.

```
L=[1,2,3,4]  
L.pop(3)  
print("L = ",L)
```

```
L = [1, 2, 3]
```



Métodos de lista

07 REMOVE: Permite eliminar un elemento de la lista por su valor, para lo cual se le debe indicar el elemento que se quiere eliminar.

```
L=[1,2,3,4]  
L.remove(4)  
print("L = ",L)
```

```
L = [1, 2, 3]
```



Métodos de lista

08 SORT: Permite ordenar una lista ascendentemente (de menor a mayor). El requisito para ordenar es que los elementos de la lista sean todos del mismo tipo (números o texto).

```
L=[100,2,33,-4]
L.sort()
print("L = ",L)
```

```
L = [-4, 2, 33, 100]
```

Para ordenar la lista descendientemente (de mayor a menor) se utiliza el argumento "reverse=True", como en el ejemplo:

```
L=[100,2,33,-4]
L.sort(reverse=True)
print("L = ",L)
```

```
L = [100, 33, 2, -4]
```



Con tu compañero o compañera de puesto...

**Conversen, en pocas
palabras, ¿qué son las listas
y para qué sirven?**



Funciones

f



Tipos de funciones

- En Python, las funciones provienen de por lo menos tres lugares:
 - De Python mismo: varias funciones, como `print()` o `input()`, son una parte integral de Python, y siempre están disponibles. Se les llama funciones integradas.
 - De los módulos preinstalados de Python: muchas de las funciones están disponibles en módulos instalados juntamente con Python. Para poder utilizar estas funciones el programador debe realizar algunos pasos adicionales, como por ejemplo el uso de `import`.
 - Directamente del código: el programador también puede escribir sus funciones propias y colocarlas dentro del código, para usarlas libremente dentro del programa.



Definición de una función

- Una función es un bloque de código escrito en Python, al cual se le asigna un nombre.

Algunas de sus características son:

- Realiza una tarea específica.
- Puede recibir cero o más argumentos como entrada, que serán necesarios para realizar algún procedimiento.
- Puede o no devolver un valor al programa.
- Puede ser llamada desde el programa las veces que se necesite ejecutar.



Ventajas del uso de funciones

- El uso de funciones es un componente muy importante en la programación estructurada o modular, y tiene varias ventajas:
 - Modularización: permite segmentar/dividir un programa complejo en una serie de partes o módulos más simples, facilitando así la programación y el depurado.
 - Reutilización: permite utilizar una misma función varias veces en el mismo programa, o en distintos programas.



Consideraciones generales

- Usualmente las funciones se definen al inicio del programa.
- Una función no puede ser utilizada antes de definirse.
- La definición de una función es una sentencia ejecutable donde se describe una acción a realizar.
- La función no se ejecuta solo con crearla, para poder ejecutarla necesita ser llamada o invocada desde alguna parte del programa.
- El nombre de una función no debe ser igual al nombre de una variable o de una palabra reservada de Python (por ejemplo: int), ya que ello causaría un conflicto de ambigüedad.



Sintaxis de definición

```
def NOMBRE (lista_de_parámetros):  
    SENTENCIAS...  
    RETURN [expresion]
```

Donde:

- DEF: permite definir la función.
- NOMBRE: es el nombre de la función con el cual se le llamará.
- LISTA_DE_PARAMETROS: es la lista de valores que puede recibir una función.
- SENTENCIAS: es el bloque de sentencias en lenguaje Python que realizará cierta operación dada.
- RETURN [EXPRESION]: es la sentencia que permite devolver una expresión o variable al programa.



Ejemplos

```
def saludo():  
    print("Hola!")
```

```
def resta(a, b):  
    return a - b
```

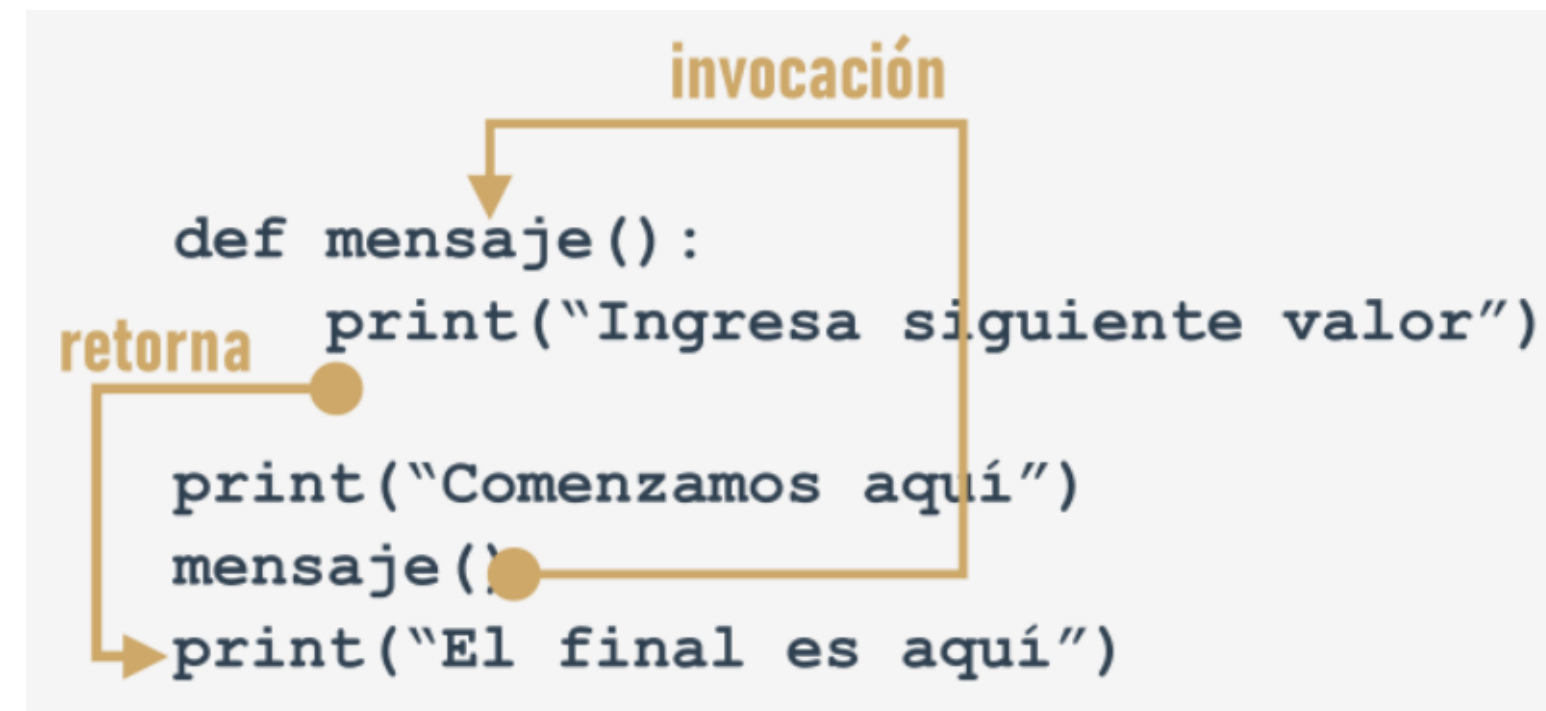
```
def SUMA_LISTA(L):  
    suma=0  
    for i in range(len(L)):  
        suma+=L[i]  
    return(suma)
```



Llamada o invocación a una función

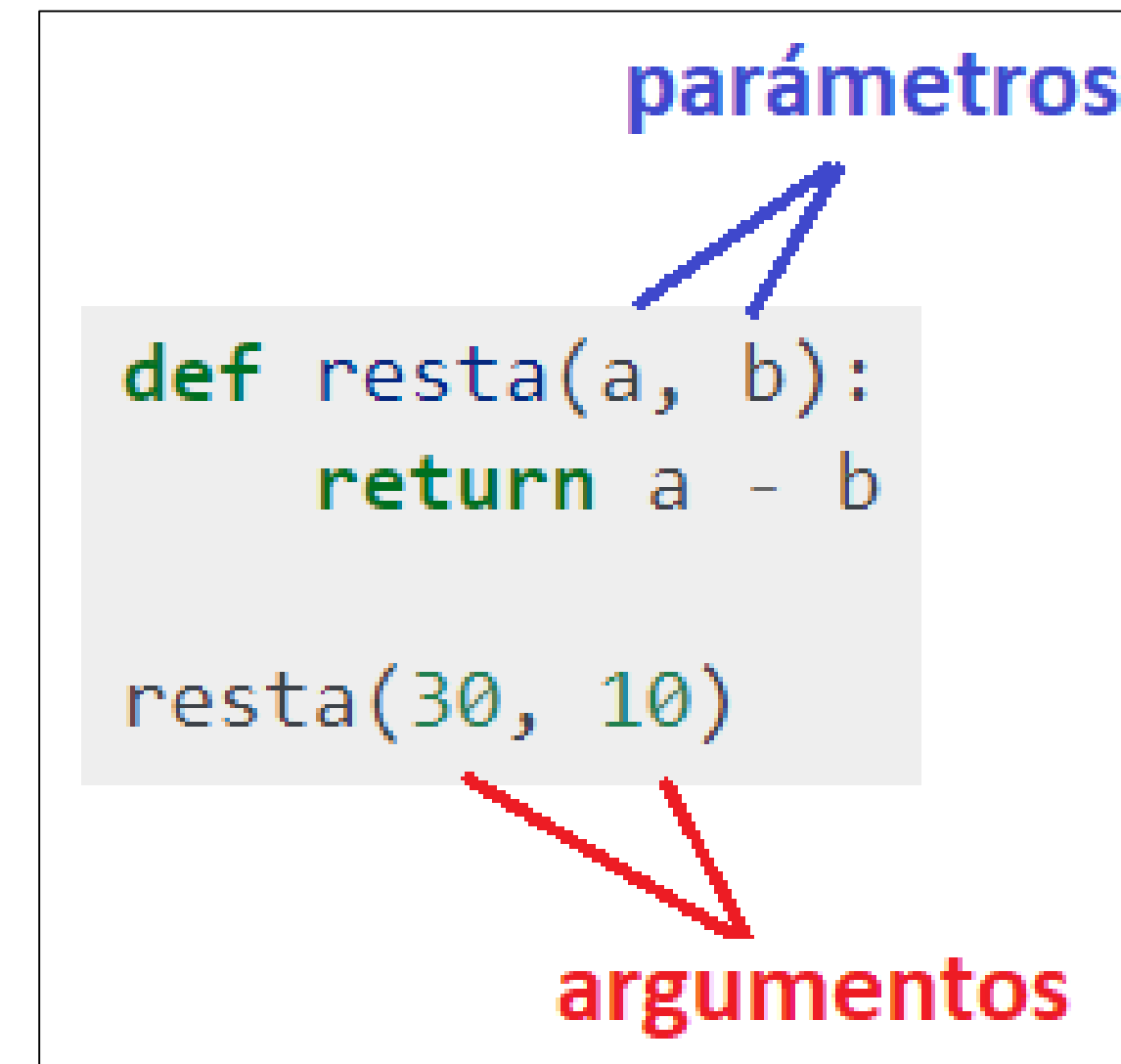
● La invocación de la función sucede así:

- 1 Cuando se invoca una función, Python recuerda el lugar donde esto ocurre y salta hacia dentro de la función invocada.
- 2 El cuerpo de la función es entonces ejecutado.
- 3 Al llegar al final de la función, Python regresa al lugar inmediato después de donde ocurrió la invocación.



Parámetros y argumentos

- Al definir una función, los valores que se reciben se denominan parámetros, pero durante la llamada, los valores que se envían se denominan argumentos.



Parámetros y argumentos

- Los parámetros solo existen dentro de las funciones en donde han sido definidos.
- El único lugar donde un parámetro puede ser definido es entre los paréntesis después del nombre de la función.
- Los argumentos existen fuera de las funciones.
- Los argumentos son los que traspasan los valores a los parámetros correspondientes.
- La asignación de un valor a un parámetro de una función se hace en el momento en que la función es llamada o invocada, esta llamada debe contener la misma cantidad de argumentos que de parámetros definidos en la función.



Tipos de argumentos

● POR POSICIÓN

Cuando se envían argumentos a una función, estos se reciben por orden en los parámetros definidos.

```
def resta(a, b):  
    return a - b  
  
resta(30, 10)
```

● POR ORDEN

Es posible evadir el orden de los parámetros si se indica durante la llamada qué valor tiene cada parámetro a partir de su nombre.

```
def resta(a, b):  
    return a - b  
  
resta(b=30, a=10)
```



La sentencia return

- Las funciones pueden comunicarse con el exterior al programa principal, usando la sentencia return.
- El proceso de comunicación con el exterior se hace devolviendo valores desde la función al programa.

```
def resta(a, b):  
    return a - b  
  
a=int(input("Ingrese primer numero: "))  
b=int(input("Ingrese segundo numero: "))  
R=resta(a,b)  
print("La resta de los nuemros es: ",R)
```

```
Ingrese primer numero: 3  
Ingrese segundo numero: 5  
La resta de los nuemros es: -2
```



Ejemplo de uso

- El siguiente programa utiliza una función para calcular y devolver el valor del área de un rectángulo.

```
def area_rectangulo(base, altura):  
    return base*altura  
  
base=int(input("Ingrese base del rectangulo: "))  
altura=int(input("Ingrese altura del rectangulo: "))  
print("El area del rectangulo es: ",area_rectangulo(base,altura))
```

```
Ingrese base del rectangulo: 3  
Ingrese altura del rectangulo: 2  
El area del rectangulo es: 6
```



Con tu compañero o compañera de puesto...

**Conversen, en pocas palabras,
¿qué son las funciones y para
qué sirven?**



¿Tienes preguntas de lo trabajado hasta aquí?



Referencias

- ***Las imágenes son de autoría personal, excepto la de la diapositiva 8***



Ticket de salida

01

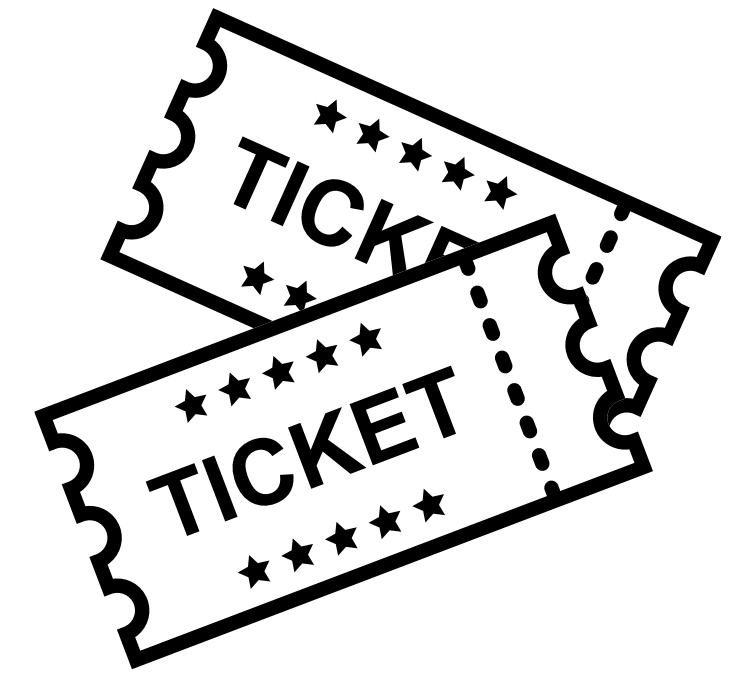
¿Cómo le explicarías a una persona, que no tiene conocimientos técnicos, qué es una lista?

02

¿Para qué sirven los métodos en listas?

03

Explica con tus propias palabras el concepto de función.



Ticket de salida

04

¿Para qué se definen funciones en un programa?

05

¿Cuál fue el contenido más difícil de entender en estas actividades?

¿Qué harías para superar esta dificultad?

