Ciclo 1: Fundamentos de programación con Python

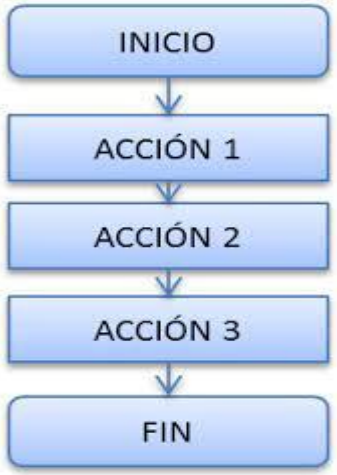
Implementación de estructuras de control de secuencia

Ciclo 1: Fundamentos de programación con Python

Implementación de estructuras de control de secuencia

Estructura secuencial

La estructura secuencial es aquella en la que una acción (instrucción) sigue a otra en secuencia. Las tareas se suceden de tal modo que la salida de una es la entrada de la siguiente y así sucesivamente hasta el fin del proceso.



Ciclo 1: Fundamentos de programación con Python

Implementación de estructuras de control de secuencia

Tipos de asignaciones posibles que se pueden encontrar en un algoritmo:

- Simples: Ejem. (a=15)
- Contador: Ejem. (a=a+1)
- Acumulador: Ejem. (a=a+b)
- De trabajo: Ejem.(a=c+b*2/4)

Ciclo 1: Fundamentos de programación con Python

Implementación de estructuras de control de secuencia

- Suponga que un individuo desea invertir su capital en un banco y desea saber cuanto dinero ganará después de un mes si el banco paga a razón de 15% efectivo anual.

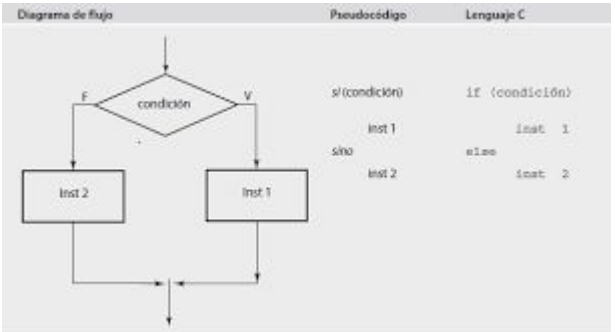
- Un vendedor recibe un sueldo base, más un 10% extra por comisión de sus ventas, el vendedor desea saber cuánto dinero obtendrá por concepto de comisiones por las tres ventas que realiza en el mes y el total que recibirá en el mes tomando en cuenta su sueldo base y comisiones.

Ciclo 1: Fundamentos de programación con Python

Implementación de estructuras de control de secuencia

Estructura condicional

Una instrucción (o estructura) condicional o de selección es aquella que establece qué instrucciones deben de ejecutarse o no, en función del valor de una condición.



Ciclo 1: Fundamentos de programación con Python

Implementación de estructuras de control de secuencia

Tabla de verdad

p	q	$\neg p$	$\neg q$	$p \wedge q$	$p \vee q$	$p \rightarrow q$	$p \leftrightarrow q$
V	V	F	F	V	V	V	V
V	F	F	V	F	V	F	F
F	V	V	F	F	V	V	F
F	F	V	V	F	F	V	V

Video: https://www.youtube.com/watch?v=8gCoQCkx9A0

Ciclo 1: Fundamentos de programación con Python

Implementación de estructuras de control de secuencia

TIPOS DE INSTRUCCIONES CONDICIONALES

- De condición simple.
- Bicondicional o de condición múltiple.

Ciclo 1: Fundamentos de programación con Python

Implementación de estructuras de control de secuencia

TIPOS DE INSTRUCCIONES CONDICIONALES

- De condición simple.

  Un condicional simple es una estructura de control que ejecuta un conjunto de líneas de código si es cierta una expresión booleana. En procesamiento un condicional simple se expresa según este código:

```
if (expresion) {
    Líneas de código que se ejecutan
    si la expresión es cierta.
}
```

Ciclo 1: Fundamentos de programación con Python

Implementación de estructuras de control de secuencia

TIPOS DE INSTRUCCIONES CONDICIONALES

- Bicondicional o de condición múltiple.

  A un condicional simple se le puede añadir la cláusula else para especificar qué líneas de código se quieren ejecutar si la expresión booleana es falsa. En este caso se habla de la estructura if-else, que en procesamiento se expresa según este código:

```
if (expresion) {
    Líneas de código que se ejecutan
    si la expresión es cierta.
} else {
    Líneas de código que se ejecutan
    si la expresión es falsa.
}
```

Ciclo 1: Fundamentos de programación con Python

Implementación de estructuras de control de secuencia

TIPOS DE INSTRUCCIONES CONDICIONALES

- Anidamiento

  Las instrucciones (o estructuras) condicionales pueden aparecer en cualquier bloque del programa. Por ejemplo puede aparecer en el bloque de una función pero también puede aparecer en los bloques que componen una instrucción condicional.

Ciclo 1: Fundamentos de programación con Python

Implementación de estructuras de control de secuencia

```
if (expresion1) {
    ...
} else if (expresion2) {
        ...
    } [else {
        ...
    }]
```

```
if (a > 0) {
    if (a % 2 == 0)
        println("Es un número positivo y par.");
}
```

```
if ((a > 0) && (a % 2 == 0))
    println("Es un número positivo y par.");
}
```

Ciclo 1: Fundamentos de programación con Python

Implementación de estructuras de control de secuencia

Ejercicios:

- Leer 2 números; si son iguales que los multiplique, si el primero es mayor que el segundo que los reste y si no que los sume. (**compararYOperar.py**)

- En una tienda de descuento se efectúa una promoción en la cual se hace un descuento sobre el valor de la compra total según el color de la bolita que el cliente saque al pagar en caja. Si la bolita es de color blanco no se le hará descuento alguno, si es verde se le hará un 10% de descuento, si es amarilla un 25%, si es azul un 50% y si es roja un 100%. Determinar la cantidad final que el cliente deberá pagar por su compra. se sabe que solo hay bolitas de los colores mencionados. (**pagoCompra.py, pagoCompraCase.py**)

- En una fábrica de computadoras se planea ofrecer a los clientes un descuento que dependerá del número de computadoras que compre. Si las computadoras son menos de cinco se les dará un 10% de descuento sobre el total de la compra; si el número de computadoras es mayor o igual a cinco pero menos de diez se le otorga un 20% de descuento; y si son 10 o más se les da un 40% de descuento. El precio de cada computadora es de $3.500.000

Ciclo 1: Fundamentos de programación con Python

Implementación de estructuras de control de secuencia

- Determinar si un año es bisiesto o no. Si el año es bisiesto se debe asignar en la variable **a** el valor '**s**' de lo contrario '**n**'. Un año es bisiesto si es divisible por 4 y no es divisible por 100, excepto los los años múltiplos de 400.

- Elaborar un algoritmo que lea dos números enteros y los almacene en las variables **a** y **b**. El algoritmo debe determinar si uno es múltiplo del otro, en cuyo caso se debe escribir **'s'**. Se deberá escribir **'n'** en caso contrario.

- Diseñar un algoritmo que pida por teclado tres números; si el primero es negativo, debe imprimir el producto de los tres y si no lo es, imprimirá la suma.

- Escribir un algoritmo, que solicite el valor (longitud) de los tres lados de un triángulo, a continuación deberá mostrar si es isósceles, escaleno o equilátero.

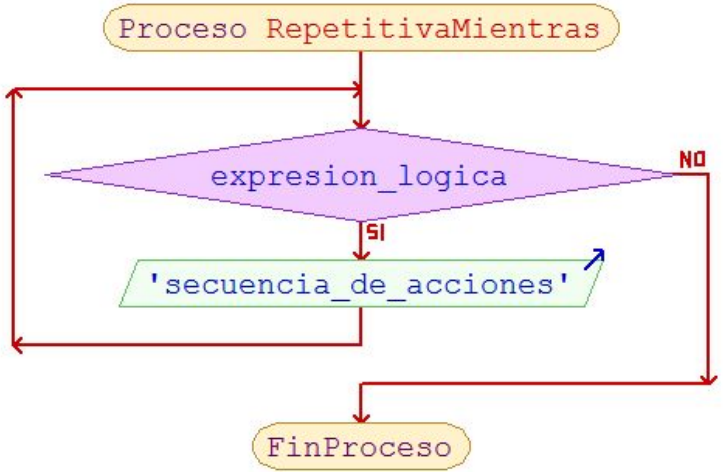
Ciclo 1: Fundamentos de programación con Python

Implementación de estructuras de control de repetición (while)

Ciclo 1: Fundamentos de programación con Python

Implementación de estructuras de control de repetición (while)

La instrucción "**while**", es una estructura de control repetitiva que puede impedir la ejecución de un conjunto de instrucciones, si la evaluación de la expresión relacional y/o lógica es falsa. Esto significa que se convierte en repetitiva únicamente cuando la evaluación de la condición es verdadera.



Ciclo 1: Fundamentos de programación con Python

Implementación de estructuras de control de repetición (while)

La instrucción **break**, **continue**, **else** y manejo de **excepciones** en el **while**.

```
i = 1
while i < 6:
  print(i)
  if i == 3:
    break
  i += 1
```

```
i = 0
while i < 6:
  i += 1
  if i == 3:
    continue
  print(i)
```

```
i = 1
while i < 6:
  print(i)
  i += 1
else:
  print("i is no longer less than 6")
```

Ciclo 1: Fundamentos de programación con Python

Implementación de estructuras de control de repetición (while)

La instrucción **break**, **continue**, **else** y manejo de **excepciones** en el **while**.

```
while True:
    try:
        x = int(input("Please enter a number: "))
        break
    except ValueError:
        print("Oops!  That was no valid number.  Try again...")
```

Ver código: **testExceptionWhile.py**

Ciclo 1: Fundamentos de programación con Python

Implementación de estructuras de control de repetición (while)

Ejercicios:

- Elaborar un algoritmo que calcule la suma de los números de 1 a **n**.
- Elaborar un algoritmo que almacene en la variable **a** la cantidad de números pares y en **b** la cantidad números impares que existan entre 1 y **n**.
- Suponga que en la variable **a** existe un número entre 1 y 999. Elabore un conjunto de instrucciones para invertir la cifra almacenada en **a**. Por ejemplo el dato 834 debe quedar en 438. El dato 17 debe quedar en 71, etc. (**invertirCifra.py**)
- Elaborar un algoritmo que realice (mostrar) las tablas de multiplicar del 1 al número **n**.

Ciclo 1: Fundamentos de programación con Python

Implementación de estructuras de control de repetición (for)

Ciclo 1: Fundamentos de programación con Python

Implementación de estructuras de control de repetición (for)

El bucle **for**

En general, un bucle es una estructura de control que repite un bloque de instrucciones. Un bucle for es un bucle que repite el bloque de instrucciones un número predeterminado de veces. El bloque de instrucciones que se repite se suele llamar cuerpo del bucle y cada repetición se suele llamar iteración.

Los bucles **for** se utilizan tradicionalmente cuando tiene un bloque de código que desea repetir un número fijo de veces. La instrucción **for** de Python itera sobre los miembros de una secuencia en orden, ejecutando el bloque cada vez. Contrasta la instrucción **for** con el ciclo **while**, que se utiliza cuando es necesario comprobar una condición en cada iteración o para repetir un bloque de código para siempre.

La sintaxis de un bucle **for** es la siguiente:

- **for** variable **in** elemento iterable (lista, cadena, range, etc.):
  cuerpo del bucle

Ciclo 1: Fundamentos de programación con Python

Implementación de estructuras de control de repetición (for)

El bucle **for**

```
for x in range(0, 3):
    print("We're on time %d" % (x))
```

for-01.py

```
for x in range(3):
    if x == 1:
        break
```

for-02.py

```
for x in range(1, 11):
    for y in range(1, 11):
        print('%d * %d = %d' % (x, y, x*y))
```

for-03.py

Ciclo 1: Fundamentos de programación con Python

Implementación de estructuras de control de repetición (for)

El bucle **for**

```
for x in range(3):
    if x == 1:
        break
```

for-04.py

```
for x in range(3):
    print(x)
else:
    print('Final x = %d' % (x))
```

for-05.py

```
string = "Hello World"
for x in string:
    print(x)
```

for-06.py

Ciclo 1: Fundamentos de programación con Python

Implementación de estructuras de control de repetición (for)

El bucle **for**

```
collection = ['hey', 5, 'd']
for x in collection:
    print(x)
```

for-07.py

```
list_of_lists = [ [1, 2, 3], [4, 5, 6], [7, 8, 9]]
for list in list_of_lists:
    for x in list:
        print(x)
```

for-08.py

```
for x in my_range(1, 10, 0.5):
    print(x)
```

for-09.py

Ciclo 1: Fundamentos de programación con Python

Implementación de estructuras de control de repetición (while)

Ejercicios (manejando el ciclo **for**):

- Elaborar un algoritmo que calcule la suma de los números de 1 a **n**.
- Elaborar un algoritmo que almacene en la variable **a** la cantidad de números pares y en **b** la cantidad números impares que existan entre 1 y **n**.
- Suponga que en la variable **a** existe un número entre 1 y 999. Elabore un conjunto de instrucciones para invertir la cifra almacenada en **a**. Por ejemplo el dato 834 debe quedar en 438. El dato 17 debe quedar en 71, etc.
- Elaborar un algoritmo que realice (mostrar) las tablas de multiplicar del 1 al número **n**.
- Hallar la cantidad de vocales que están contenidas en una cadena de caracteres. (contarVocales.py)

Ciclo 1: Fundamentos de programación con Python

Uso de String

Ciclo 1: Fundamentos de programación con Python

Uso de String

Métodos principales de la clase string

- Convertir a mayúscula la primera letra
  Método: capitalize()

  Retorna: una copia de la cadena con la primera letra en mayúsculas.

  ```
  >>> cadena = "bienvenido a mi aplicación"
  >>> print cadena.capitalize()
  Bienvenido a mi aplicación
  ```

Ciclo 1: Fundamentos de programación con Python

Uso de String

Métodos principales de la clase string

- Convertir una cadena a minúsculas

  Método: lower()

  Retorna: una copia de la cadena en minúsculas.

```
>>> cadena = "Hola Mundo"
>>> print cadena.lower()
hola mundo
```

Ciclo 1: Fundamentos de programación con Python

Uso de String

Métodos principales de la clase string

- Convertir una cadena a mayúsculas

  Método: upper()

  Retorna: una copia de la cadena en mayúsculas.

```
>>> cadena = "Hola Mundo"
>>> print cadena.upper()
HOLA MUNDO
```

Ciclo 1: Fundamentos de programación con Python

Uso de String

Métodos principales de la clase string

- Convertir mayúsculas a minúsculas y viceversa

  Método: swapcase()

  Retorna: una copia de la cadena convertidas las mayúsculas en minúsculas y viceversa.

  ```
  >>> cadena = "Hola Mundo"
  >>> print cadena.swapcase()
  hOLA mUNDO
  ```

Ciclo 1: Fundamentos de programación con Python

Uso de String

Métodos principales de la clase string

- Convertir una cadena en Formato Título

  Método: title()

  Retorna: una copia de la cadena convertida.

```
>>> cadena = "hola mundo"
>>> print cadena.title()
Hola Mundo
```

Ciclo 1: Fundamentos de programación con Python

Uso de String

Métodos principales de la clase string

- Centrar un texto

  Método: center(longitud[, "caracter de relleno"])

  Retorna: una copia de la cadena centrada.

  ```
  >>> cadena = "bienvenido a mi aplicación".capitalize()
  >>> print cadena.center(50, "=")
  ==========Bienvenido a mi aplicación==========

  >>> print cadena.center(50, " ")
          Bienvenido a mi aplicación
  ```

Ciclo 1: Fundamentos de programación con Python

Uso de String

Métodos principales de la clase string

- Alinear texto a la izquierda

  Método: ljust(longitud[, "caracter de relleno"])

  Retorna: una copia de la cadena alineada a la izquierda.

  ```
  >>> cadena = "bienvenido a mi aplicación".capitalize()
  >>> print cadena.ljust(50, "=")
  Bienvenido a mi aplicación==========================
  ```

Ciclo 1: Fundamentos de programación con Python

Uso de String

Métodos principales de la clase string

Alinear texto a la derecha

Método: rjust(longitud[, "caracter de relleno"])

Retorna: una copia de la cadena alineada a la derecha.

```
>>> cadena = "bienvenido a mi aplicación".capitalize()
>>> print cadena.rjust(50, "=")
==========================Bienvenido a mi aplicación

>>> print cadena.rjust(50, " ")
                         Bienvenido a mi aplicación
```

Ciclo 1: Fundamentos de programación con Python

Uso de String

Métodos principales de la clase string

Alinear texto a la derecha

Método: rjust(longitud[, "caracter de relleno"])

Retorna: una copia de la cadena alineada a la derecha.

```
>>> cadena = "bienvenido a mi aplicación".capitalize()
>>> print cadena.rjust(50, "=")
==========================Bienvenido a mi aplicación

>>> print cadena.rjust(50, " ")
                         Bienvenido a mi aplicación
```

Ciclo 1: Fundamentos de programación con Python

Uso de String

Métodos principales de la clase string

- Rellenar un texto anteponiendo ceros

  Método: zfill(longitud)

  Retorna: una copia de la cadena rellena con ceros a la izquierda hasta alcanzar la longitud final indicada.

```
>>> numero_factura = 1575
>>> print str(numero_factura).zfill(12)
000000001575
```

Ciclo 1: Fundamentos de programación con Python

Uso de String

Métodos principales de la clase string

- Contar cantidad de apariciones de una subcadena

  Método: count("subcadena" [, posicion_inicio, posicion_fin])

  Retorna: un entero representando la cantidad de apariciones de subcadena dentro de cadena.

  ```
  >>> cadena = "bienvenido a mi aplicación".capitalize()
  >>> print cadena.count("a")
  3
  ```

Ciclo 1: Fundamentos de programación con Python

Uso de String

Métodos principales del método string

- Buscar una subcadena dentro de una cadena

  Método: find("subcadena" [, posicion_inicio, posicion_fin])

  Retorna: un entero representando la posición donde inicia la subcadena dentro de cadena. Si no la encuentra, retorna -1.

```
>>> cadena = "bienvenido a mi aplicación".capitalize()
>>> print cadena.find("mi")
13
>>> print cadena.find("mi", 0, 10)
-1
```

Ciclo 1: Fundamentos de programación con Python

Uso de String

Métodos principales de la clase string

- Saber si una cadena comienza con una subcadena determinada

  Método: startswith("subcadena" [, posicion_inicio, posicion_fin])

  Retorna: True o False

```
>>> cadena = "bienvenido a mi aplicación".capitalize()
>>> print cadena.startswith("Bienvenido")
True
>>> print cadena.startswith("aplicación")
False
>>> print cadena.startswith("aplicación", 16)
True
```

Ciclo 1: Fundamentos de programación con Python

Uso de String

Métodos principales de la clase string

- Saber si una cadena es alfanumérica

  Método: isalnum()
  Retorna: True o False

```
>>> cadena = "pepegrillo 75"
>>> print cadena.isalnum()
False
>>> cadena = "pepegrillo"
>>> print cadena.isalnum()
True
>>> cadena = "pepegrillo75"
>>> print cadena.isalnum()
True
```

Ciclo 1: Fundamentos de programación con Python

Uso de String

Métodos principales de la clase string

- Saber si una cadena es alfabética

  Método: isalpha()
  Retorna: True o False

```
>>> cadena = "pepegrillo 75"
>>> print cadena.isalpha()
False
>>> cadena = "pepegrillo"
>>> print cadena.isalpha()
True
>>> cadena = "pepegrillo75"
>>> print cadena.isalpha()
False
```

Ciclo 1: Fundamentos de programación con Python

Uso de String

Métodos principales de la clase string

- Saber si una cadena es numérica

  Método: isdigit()

  Retorna: True o False

```
>>> cadena = "pepegrillo 75"
>>> print cadena.isdigit()
False
>>> cadena = "7584"
>>> print cadena.isdigit()
True
>>> cadena = "75 84"
>>> print cadena.isdigit()
False
>>> cadena = "75.84"
>>> print cadena.isdigit()
False
```

Ciclo 1: Fundamentos de programación con Python

Uso de String

Métodos principales de la clase string

- Saber si una cadena contiene solo minúsculas

  Método: islower()

  Retorna: True o False

```
>>> cadena = "pepe grillo"
>>> print cadena.islower()
True
>>> cadena = "Pepe Grillo"
>>> print cadena.islower()
False
>>> cadena = "Pepegrillo"
>>> print cadena.islower()
False
>>> cadena = "pepegrillo75"
>>> print cadena.islower()
True
```

Ciclo 1: Fundamentos de programación con Python

Uso de String

Métodos principales de la clase string

- Eliminar caracteres a la izquierda y derecha de una cadena

  Método: strip(["caracter"])

  Retorna: la cadena sustituida.

```
>>> cadena = "   www.eugeniabahit.com   "
>>> print cadena.strip()
www.eugeniabahit.com

>>> print cadena.strip(' ')
www.eugeniabahit.com
```

Ciclo 1: Fundamentos de programación con Python

Uso de String

Métodos principales de la clase string

- Eliminar caracteres a la izquierda y derecha de una cadena

  Método: strip(["caracter"])

  Retorna: la cadena sustituida.

  ```
  >>> cadena = "   www.eugeniabahit.com   "
  >>> print cadena.strip()
  www.eugeniabahit.com

  >>> print cadena.strip(' ')
  www.eugeniabahit.com
  ```

Ciclo 1: Fundamentos de programación con Python

Uso de String

Métodos principales de la clase string

- Longitud de una cadena de caracteres

```
mystring = 'python examples'
length = len(mystring)
print('length of the string is:', length)
```

Ciclo 1: Fundamentos de programación con Python

Uso de String

Métodos principales de la clase string

Ejercicio 1

Crear un módulo para validación de nombres de usuarios. Dicho módulo, deberá cumplir con los siguientes criterios de aceptación:

- El nombre de usuario debe contener un mínimo de 6 caracteres y un máximo de 12.
- El nombre de usuario debe ser alfanumérico.
- Nombre de usuario con menos de 6 caracteres, retorna el mensaje "El nombre de usuario debe contener al menos 6 caracteres".
- Nombre de usuario con más de 12 caracteres, retorna el mensaje "El nombre de usuario no puede contener más de 12 caracteres".
- Nombre de usuario con caracteres distintos a los alfanuméricos, retorna el mensaje "El nombre de usuario puede contener solo letras y números".
- Nombre de usuario válido, retorna True

Ciclo 1: Fundamentos de programación con Python

Uso de String

Métodos principales de la clase string

Ejercicio 2

Crear un módulo para validación de contraseñas. Dicho módulo, deberá cumplir con los siguientes criterios de aceptación:

- La contraseña debe contener un mínimo de 8 caracteres.
- Una contraseña debe contener letras minúsculas, mayúsculas, números y al menos 1 carácter no alfanumérico.
- La contraseña no puede contener espacios en blanco.
- Contraseña válida, retorna True.
- Contraseña no válida, retorna el mensaje "La contraseña elegida no es segura".

Ciclo 1: Fundamentos de programación con Python

Uso de String

Fortalecimiento de conceptos vistos en el módulo