

P1

Programación 1

UNIDAD 03
CLASE 04

ESTRUCTURAS REPETITIVAS Y COMENTARIOS



- | Estructuras repetitivas
- | Estructura For
- | Comentarios del código



Al final de esta clase ya podrás:

- | Comprender situaciones problemáticas que para su resolución impliquen el uso de ciclos repetitivos con características de la función "For".
- | Valorar la conveniencia de aplicar comentarios para facilitar la interpretación de cada función.

ISSD

-Des-
Desarrollo de
Software

MÓDULO
DIDÁCTICO
2023

Ciclos repetitivos

¡Bienvenidos a la cuarta clase de la materia Algoritmos y Estructuras de Datos 1!

En esta oportunidad comenzaremos a aprender situaciones problemáticas que para su resolución impliquen el uso de ciclos repetitivos con diferentes características.

Además intentaremos valorar la conveniencia de aplicar contadores y acumuladores para facilitar el diseño de algoritmos eficientes.

Organizaremos la clase en cinco temas:

- | El primero de ellos nos va a ayudar a repasar las **estructuras repetitivas**.
- | En el segundo nos adentraremos en la **estructura “For”**.
- | En tercer y último lugar trabajaremos sobre los **comentarios del código y sus beneficios**,

Al igual que la clase anterior, te propondremos algunos ejercicios comentados y otros para que los resuelvas por tu cuenta y trabajes en tu elaboración de algoritmos. Recordá que para aprender esta materia es necesario la práctica de todos y cada uno de los ejercicios.

¡Que la disfrutes!

Si querés realizar las pruebas del código en tu pc, te recomendamos hacer click en el botón ⬇
Se encuentra ubicado junto a cada código ejemplificado. De esta manera, podrás copiar y pegar el texto puro para que evitar errores en la ejecución.

Si accedes al módulo desde un navegador, te recomendamos abrir los hipervínculos en una nueva pestaña presionando la rueda central del ratón, o bien usando el botón derecho y seleccionando la opción correspondiente.



Antes de comenzar con la clase, leé el siguiente artículo.

Fundó una empresa gaming y hoy lanzó su primer videojuego: “Se puede llegar muy lejos en Argentina”

Luciano Musella tiene 28 años y es uno de los fundadores de Whiteboard Games, una compañía que rubricó el primer convenio con la firma alemana Gameforge, líder en la industria de los videojuegos, en Latinoamérica. La historia de un emprendimiento que nació en tesis universitaria y que hoy se lanza su primera creación

Cuando Luciano Musella tenía tres años, sus padres le regalaron una consola Sega. Durante su infancia y su adolescencia creyeron que habían cometido un grave error: él se la pasaba atado a ese y a los otros aparatos que se acumularon en su casa. Hoy su hijo ya tiene 28 años y dice: “**Mis papás están babeando del orgullo**, se ponen realmente contentos con mis logros y yo me siento eternamente agradecido”. La baba y el orgullo obedece al convenio que

la empresa que fundó Luciano Musella firmó con la distribuidora alemana Gameforge, líder mundial de videojuegos en línea.

Es, según sus propias palabras, la primera alianza en la historia del país entre una compañía de videojuegos alemana y una argentina. Es, según el Ministro del Interior de la Nación, Wado de Pedro, el primer acuerdo en Latinoamérica con la compañía Gameforge: “*Un ejemplo de la potencialidad de nuestra industria del gaming*”. (...)

Este 24 de octubre, la firma lanzó **su primer videojuego: I See Red**.

“Es la culminación de un trabajo de casi tres años, que nos marca especialmente por ser nuestro primer título como Whiteboard Games, un equipo increíble que nació en 2020. También es por esto que el lunes está muy alejado de ser un final: no solo vamos a continuar I See Red y sus próximos lanzamientos en diferentes consolas, sino que este juego es el primero de muchos proyectos y desafíos para Whiteboard. Esto recién empieza y tras este gran primer escalón quedan muchos otros”, dijo Luciano.

La compañía *-y con la compañía, el juego-* tiene su germen en marzo de 2020, cuando estalla la pandemia en el país y se instrumenta el aislamiento social, preventivo y obligatorio por el covid. Él estudiaba en la Facultad Da Vinci. “Para la tesis **pensamos hacer un juego que le gustara a todo el mundo**. Nos tomamos el proyecto muy en serio. En ese momento éramos cuatro personas y comenzamos a trabajar en la producción sin otro interés más allá de sacarnos un 10, recibirnos y listo”. Todo resultó tal lo planeado: una nota altísima, el final de la carrera, el egreso y el vacío. “**¿Y ahora qué hacemos?**”, pensó.

“Ahí empezó una intensa búsqueda de gente que quisiera apoyarnos y brindarnos una plataforma económica para poder forjarnos como empresa -contó Luciano-. Para hacer un juego se necesita mucha gente que haga distintas tareas. Y estábamos enfocados en eso cuando conocí a alguien que sería uno de mis socios fundadores”. Se vinculó, azarosamente, con un inversor de tecnología de Neutrón Project Accelerator, que estaba apadrinando, en ese momento, proyectos gaming.

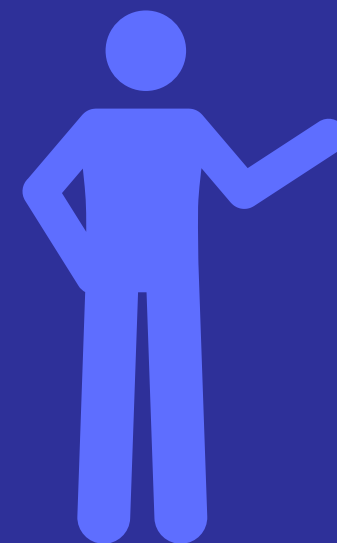
“El inversor nos dijo que en vez de diseñar un juego, por qué no crean una empresa de videojuegos que a su vez tenga muchos vídeo juegos adentro. Y así es como nace Whiteboard. (...)”

Empezaron siendo cuatro fundadores. Se sumó un quinto. Se sumó más gente con ganas de ayudar gratuitamente. Se sumó un inversor y un crecimiento de la potencialidad y de los empleados: 18. Se sumó un nuevo proyecto y con él otros cuatro trabajadores. Whiteboard Games es una compañía argentina incipiente en la industria de los videojuegos con un cuerpo laboral nutrido por 22 personas. “**Somos creadores, diseñadores, desarrolladores, y gamers haciendo juegos que nosotros mismos querríamos jugar**”, describe la página web. (...)”

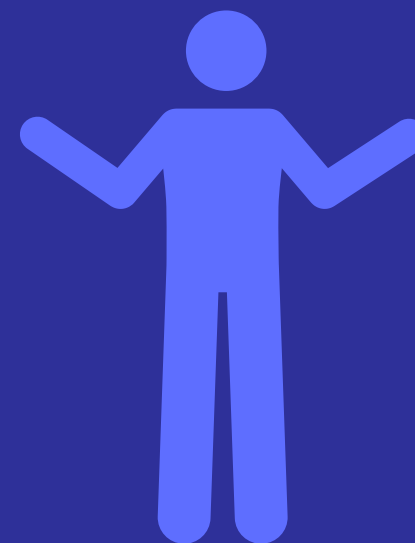
Es contemporáneo, también, a una época que presenta un escenario auspicioso. Latinoamérica es la región con mayor crecimiento de la industria gaming en el globo con un crecimiento interanual que alcanzó picos de hasta un 20% y Argentina lidera la avanzada latinoamericana después de haber facturado 500 millones de dólares en 2016 (...)”

Infobae – Octubre de 2022

La Programación tiene múltiples aristas: no se limita al desarrollo de aplicaciones administrativas, sino que abarca cada vez más áreas consideradas menos importantes, como los videojuegos.



Pasar de ser un usuario que juega con lo que otros pensaron, a ser capaces de imaginar y crear nuestros propios recorridos, personajes y desafíos, es cambiar el rol de consumidores al de productores de software de alto valor.



Con lo que vamos a aprender en la clase de hoy, estarás en condiciones de empezar a imaginar tus propias aplicaciones ¡Adelante!



Repaso de la Estructura Repetitiva

¿Conocés PacMan? Seguro que sí. Supongamos ahora que te han pedido que realices la programación de este juego. [\[1\]](#)

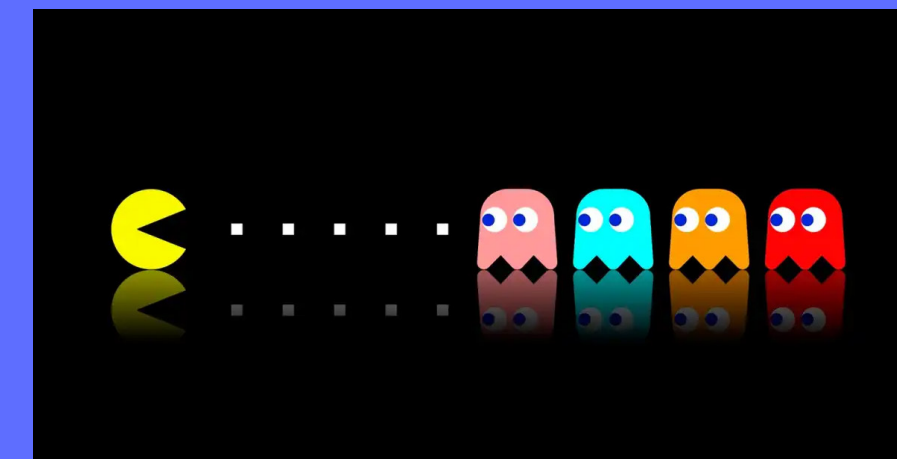
Ya sabés que el personaje se mueve dentro de un laberinto, y que cada vez que es tocado por uno de los fantasmas, pierde una vida. Podrá continuar el juego hasta que se quede sin vidas, en donde aparecerá el fatídico “**Game Over**”.

Si PacMan tiene 3 vidas cuando empieza el juego, con lo que sabés de programación hasta hora, no te queda otra que repetir tres veces el algoritmo del juego.

Pero hay una solución mucho más práctica, que consiste en usar una instrucción que se **encarga** de repetir automáticamente una secuencia de comandos determinada.

Si la condición que determina la ejecución de las instrucciones es siempre verdadera, estamos en presencia de un ciclo repetitivo infinito. Dicha situación es un error de programación, porque nunca finalizaría el programa. En el ejemplo de PacMan, la condición que dará por terminado el ciclo repetitivo, será que la variable **vidas** sea igual a cero, en cuyo caso el juego llegará a su fin.

|1|



Una estructura repetitiva permite ejecutar una instrucción o un conjunto de instrucciones varias veces, dependiendo de una condición que puede ser falsa o verdadera.



Desempeño 25

En las clases anteriores hablamos de lo que significaba “inicializar” una variable. En el ejemplo del PacMan ¿cómo debería inicializarse la variable **vidas**? ¿Y la variable **puntos**?

Repaso de la estructura *While*

Volvamos al ejemplo del PacMan. Suponiendo, como dijimos, que tenemos una variable llamada **vidas** que empieza valiendo 3 y que disminuye en 1 cada vez que un fantasma toca al personaje, **el juego continuará mientras vidas sea mayor a 0**. Cuando vidas contenga el valor 0, significará que las vidas se acabaron, y también el juego.

Por lo tanto, **vidas > 0** será la **condición** que determinará que el ciclo repetitivo continúe. Cuando esta condición sea Falsa (es decir, que **vidas** sea igual o menor que 0), el ciclo dejará de funcionar.

Si te recordás la clase anterior, habrás deducido que podemos resolverlo utilizando la estructura repetitiva *While*, y resolver fácilmente este diagrama.

Suponiendo que tenemos una variable llamada vidas que almacena la cantidad de vidas que nos quedan, y otra llamada puntos, que guarda el puntaje total obtenido a cada momento:
¿cómo variará su contenido?

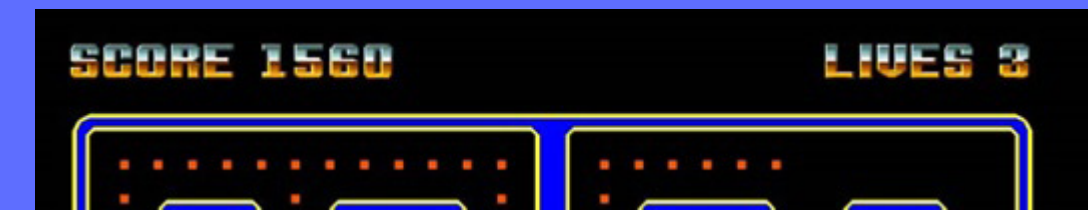
| Primero, veamos el caso de vidas: empezará el juego valiendo 3, y cada vez que un fantasma toque al PacMan, su valor disminuirá en 1. [\[2\]](#)

| **¿Y puntos?** En este caso, cuando empiece el juego contendrá el valor 0 y por cada pelotita que comamos, su contenido se incrementará en 10.

Te invitamos a que elabores el diagrama de flujo y la codificación de este repaso.

[\[2\]](#)

Recordamos que las variables, tales como vidas y puntos, cuyo valor se incrementa o decrementa en una cantidad constante cada vez que se produce un determinado suceso o acción, se denominan **contadores**.



La estructura “*For*”

Cualquier problema que requiera una estructura repetitiva se puede resolver empleando la estructura `while`, pero hay otra estructura repetitiva cuyo planteo es más sencillo en ciertas situaciones donde tenemos que recorrer una lista de datos definidos con anticipación.

En general, la estructura `For` se utiliza en aquellas situaciones en las cuales conocemos de antemano la cantidad de veces que se desea ejecutar un bloque de instrucciones, como por ejemplo cargar 10 números, ingresar 5 notas de alumnos, etc.

Supongamos que nos solicitan diseñar un juego que consista en la ejecución de tiros penales, en el que resulte ganador quien mayor cantidad de goles convierta después de 10 intentos. De acuerdo a esto, será necesario incluir un ciclo repetitivo que habilite realizar 10 disparos a cada participante antes de mostrar al ganador.

A primera vista podríamos resolverlo con una estructura **`while`**, como la que vimos la clase anterior.

Esta estructura incluiría un contador de tiros (**`i`**) que se incrementaría en cada vuelta del programa.

La condición del **`while`** será verdadera en la medida de que dicho contador no supere el décimo intento (**`i > 10`**).

En la vida real, existen muchos problemas que involucran ciclos repetitivos, pero en los cuales, como en el ejemplo, conocemos anticipadamente la cantidad de veces en que deberá reiterarse.

A fin de optimizar el diseño de algoritmos de este tipo, podemos aplicar una estructura llamada **for**, cuya función bien podría ser reemplazada por un **while** más un contador, pero que tiene la ventaja de simplificar el proceso y hacerlo más sencillo de visualizar.

La representación codificada de **for** es la que te mostramos a continuación.

| Programa:

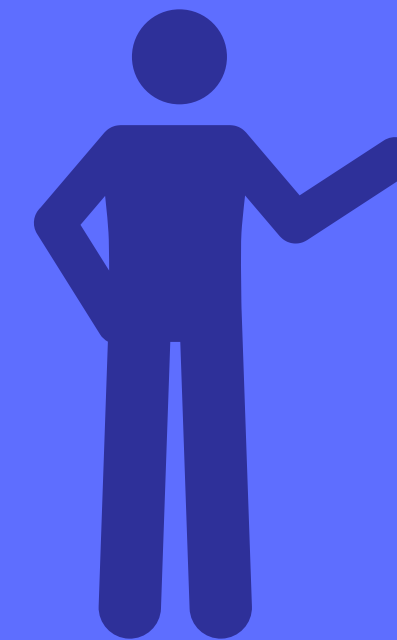
```
for x in range(101):  
    print(x)
```

Tenemos primero la palabra clave **for** y seguidamente **el nombre de la variable** que almacenará en cada vuelta del for el valor entero que retorna la función range.

La función range retorna la primera vez el valor 0 y se almacena en x, luego el 1 y así sucesivamente hasta que retorna el valor que le pasamos a range menos uno (es decir en nuestro ejemplo al final retorna un 100)

Este mismo problema resuelto con la estructura while se vería así:

```
x=0  
while x<101:  
    print(x)  
    x=x+1
```



Ejemplo 15

Programemos un contador desde el 20 al 30

Realizar un programa que imprima en pantalla los números del 20 al 30.

| Programa: ⌵

```
for x in range(20,31):  
    print(x)
```

La función **range** puede tener dos parámetros; el primero indica el valor inicial que tomará la variable x, cada vuelta del for la variable x toma el valor siguiente hasta llegar al valor indicado por el segundo parámetro de la función range menos uno.

Ejemplo 16

1...3...5...7...

Realizar un programa que imprima todos los números impares que hay entre 1 y 100.

| Programa: ⬇

```
for x in range(1,100,2):  
    print(x)
```

La función range puede tener también tres parámetros, el primero indica el valor inicial que tomará la variable x, el segundo parámetro el valor final (que no se incluye) y el **tercer parámetro indica cuanto se incrementa cada vuelta x**.

En nuestro ejemplo anterior la primer vuelta del for x recibe el valor 1, la segunda vuelta toma el valor 3 y así sucesivamente hasta el valor 99.

Ejemplo 17

Una nueva forma de sumar y promediar

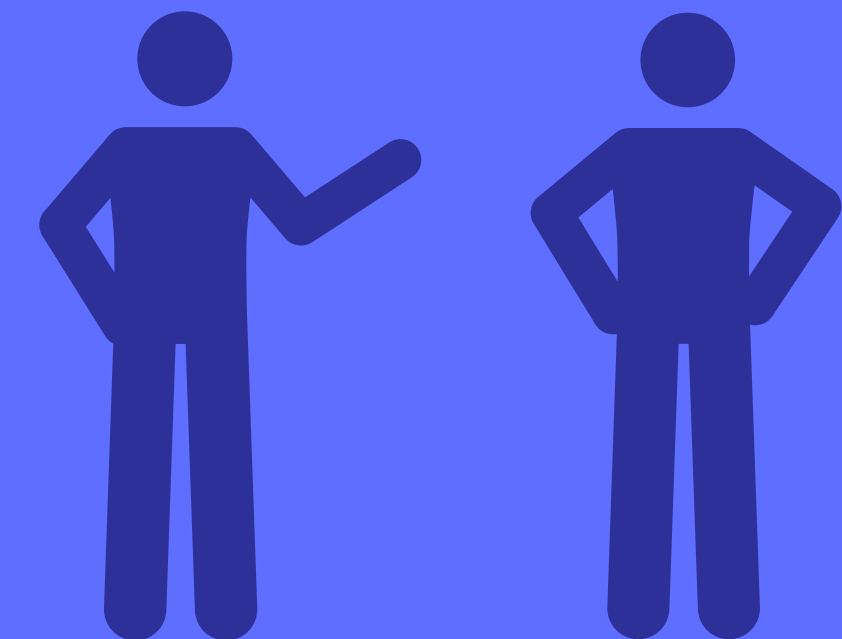
Desarrollá un programa que permita la carga de 10 valores por teclado y, posteriormente nos muestre la suma de los valores ingresados y su promedio.

| Programa: 

```
suma=0
for f in range(10):
    valor=int(input("Ingrese valor:"))
    suma=suma+valor
print("La suma es")
print(suma)
promedio=suma/10
print("El promedio es:")
print(promedio)
```

Como vemos la variable f del for solo sirve para iterar(repetir) las diez veces el bloque contenido en el for. El resultado hubiese sido el mismo si llamamos a la función range con los valores: range(1,11)

Este problema ya lo desarrollamos, lo resolveremos empleando la estructura for para repetir la carga de los diez valores por teclado.



Ejemplo 18

Contador de aprobados y reprobados

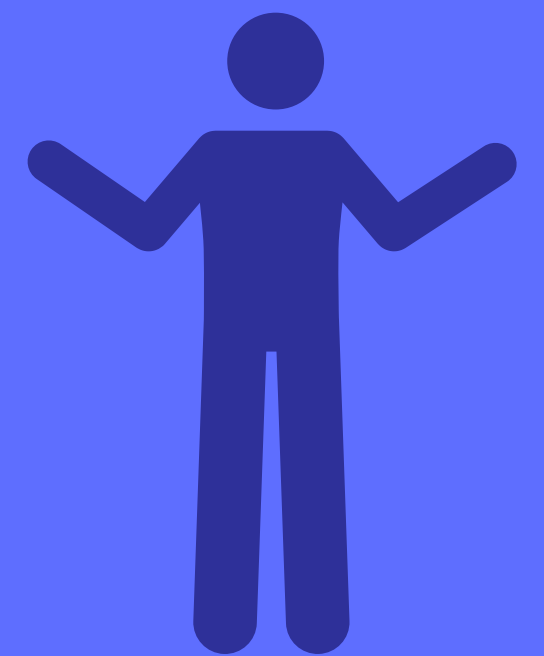
Escribir un programa que solicite por teclado 10 notas de alumnos, y luego nos informe cuántos datos tienen notas mayores o iguales a 7 y cuántos menores.

Nuevamente utilizamos el for ya que sabemos que el ciclo repetitivo debe repetirse 10 veces.

| Programa: 

```
aprobados=0
reprobados=0
for f in range(10):
    nota=int(input("Ingresa la nota:"))
    if nota>=7:
        aprobados=aprobados+1
    else:
        reprobados=reprobados+1
print("Cantidad de aprobados")
print(aprobados)
print("Cantidad de reprobados")
print(reprobados)
```

Recordemos que si usáramos el while deberíamos llevar un contador y recordar de incrementarlo en cada vuelta.



Ejemplo 19

Múltiplos de 3 y de 5

Escribir un programa que lea 10 números enteros y luego muestre cuántos valores ingresados fueron múltiplos de 3 y cuántos de 5.

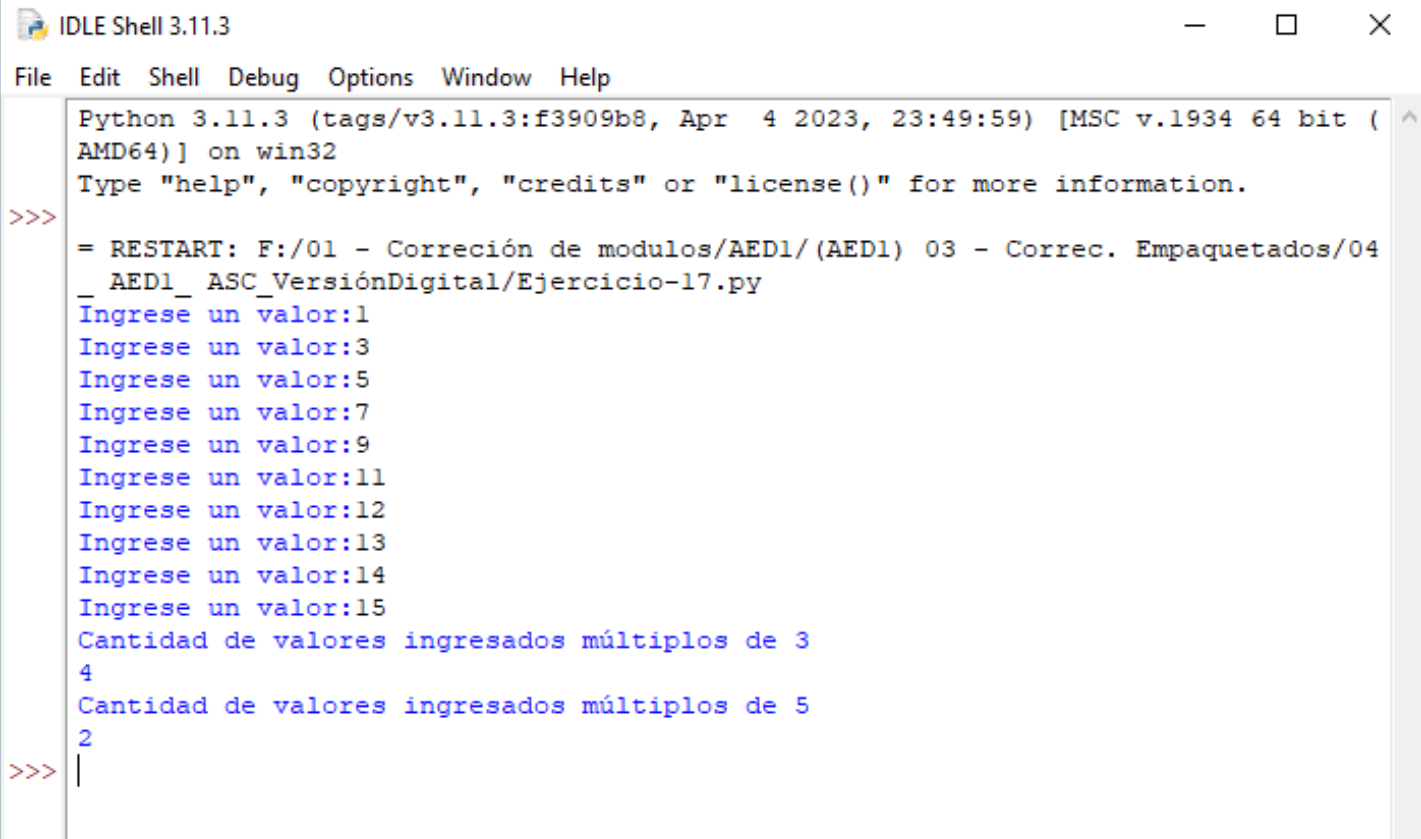
Debemos tener en cuenta que hay números que son múltiplos de 3 y de 5 a la vez.

Programa: 

```
mul3=0
mul5=0
for f in range(10):
    valor=int(input("Ingrese un valor:"))
    if valor%3==0:
        mul3=mul3+1
    if valor%5==0:
        mul5=mul5+1
print("Cantidad de valores ingresados múltiplos de 3")
print(mul3)
print("Cantidad de valores ingresados múltiplos de 5")
print(mul5)
```

Si ejecutamos el programa tenemos una salida similar a esta: |3|

|3|



```
IDLE Shell 3.11.3
File Edit Shell Debug Options Window Help
Python 3.11.3 (tags/v3.11.3:f3909b8, Apr  4 2023, 23:49:59) [MSC v.1934 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> = RESTART: F:/01 - Corrección de modulos/AED1/(AED1) 03 - Correc. Empaquetados/04
_AED1_ASC_VersiónDigital/Ejercicio-17.py
Ingrese un valor:1
Ingrese un valor:3
Ingrese un valor:5
Ingrese un valor:7
Ingrese un valor:9
Ingrese un valor:11
Ingrese un valor:12
Ingrese un valor:13
Ingrese un valor:14
Ingrese un valor:15
Cantidad de valores ingresados múltiplos de 3
4
Cantidad de valores ingresados múltiplos de 5
2
>>> |
```

Ejemplo 20

Detector de N valores mayores a 1000

Codificar un programa que lea n números enteros y calcule la cantidad de valores mayores o iguales a 1000 (n se carga por teclado)

Este tipo de problemas también se puede resolver empleando la estructura repetitiva for. Lo primero que se hace es cargar una variable que indique la cantidad de valores a ingresar. Dicha variable se carga antes de entrar a la estructura repetitiva for.

| Programa: 

```
cantidad=0
n=int(input("Cuantos valores ingresará:"))
for f in range(n):
    valor=int(input("Ingrese el valor:"))
    if valor>=1000:
        cantidad=cantidad+1
print("La cantidad de valores ingresados mayores o iguales a 1000 son")
print(cantidad)
```



Desempeño 26

Confeccionar un programa que lea n pares de datos, cada par de datos corresponde a la medida de la base y la altura de un triángulo. El programa deberá informar:

- a| De cada triángulo la medida de su base, su altura y su superficie.
- b| La cantidad de triángulos cuya superficie es mayor a 12 (la superficie se calcula multiplicando la base por la altura y a este valor se divide en 2)



Desempeño 27

Desarrollar un programa que solicite la carga de 10 números e imprima la suma de los últimos 5 valores ingresados.



Desempeño 28

Desarrollar un programa que muestre la tabla de multiplicar del 5 (del 5 al 50)



Desempeño 29

Confeccionar un programa que permita ingresar un valor del 1 al 10 y nos muestre la tabla de multiplicar del mismo (los primeros 12 términos)
Ejemplo: Si ingreso 3 deberá aparecer en pantalla los valores 3, 6, 9, hasta el 36.



Desempeño 30

Realizar un programa que lea los lados de n triángulos, e informar:

- a| De cada uno de ellos, qué tipo de triángulo es: equilátero (tres lados iguales), isósceles (dos lados iguales), o escaleno (ningún lado igual)
- b| Cantidad de triángulos de cada tipo.



Desempeño 31

Escribir un programa que pida ingresar coordenadas (x,y) que representan puntos en el plano.

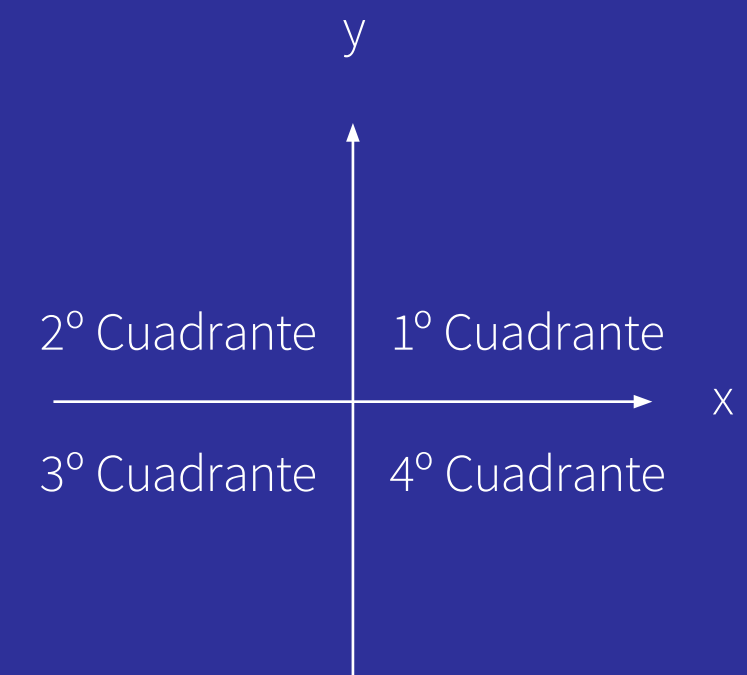
Informar cuántos puntos se han ingresado en el primer, segundo, tercer y cuarto cuadrante. Al comenzar el programa se pide que se ingrese la cantidad de puntos a procesar.

1° Cuadrante: $x > 0, y > 0$

2° Cuadrante: $x < 0, y > 0$

3° Cuadrante: $x < 0, y < 0$

4° Cuadrante: $x > 0, y < 0$





Desempeño 32

Se realiza la carga de 10 valores enteros por teclado. Se desea conocer:

- a| La cantidad de valores ingresados negativos.
- b| La cantidad de valores ingresados positivos.
- c| La cantidad de múltiplos de 15.
- d| El valor acumulado de los números ingresados que son pares.



Desempeño 33

Soluciones



Escribir un programa que utilice los siguientes datos:

Las edades de 5 estudiantes del turno mañana.
Las edades de 6 estudiantes del turno tarde.
Las edades de 11 estudiantes del turno noche.
Las edades de cada estudiante deben ingresarse por teclado.

y los procese para responder a las siguientes consignas:

- a| Obtener el promedio de las edades de cada turno (tres promedios)
- b| Imprimir dichos promedios (promedio de cada turno)
- c| Mostrar por pantalla un mensaje que indique cual de los tres turnos tiene un promedio de edades mayor.

Comentarios en Python

Un programa en Python puede definir además del algoritmo propiamente dicho una serie de comentarios en el código fuente que sirvan para aclarar los objetivos de ciertas partes del programa.

Tengamos en cuenta que **un programa puede requerir mantenimiento en el futuro.**

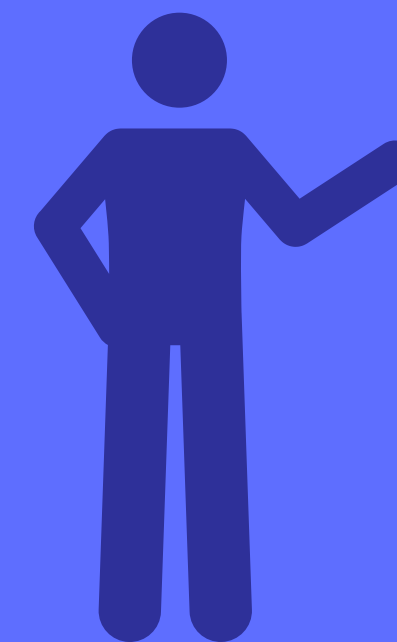
Cuando hay que implementar cambios es de muchísima utilidad encontrar en el programa comentarios sobre el objetivo de las distintas partes del algoritmo, sobre todo si es complejo.

Existen dos formas de definir comentarios en Python:

| **Comentarios de una sola línea, se emplea el carácter #:**

```
# definimos tres contadores  
conta1=0  
conta2=0  
conta3=0
```

Todo lo que disponemos después
del carácter **#** no se ejecuta



Comentarios en Python

| Comentarios de varias líneas, se emplean las comillas dobles(") de la siguiente forma:

```
"""Definimos tres contadores  
que se muestran si son distintos a cero"""  
conta1=0  
conta2=0  
conta3=0
```

Se deben utilizar tres comillas dobles seguidas al principio y al final del comentario.

Ejemplo 21

Aplicando los comentarios en la tabla del 5

Mostrar la tabla de multiplicar del 5 empleando primero el while y seguidamente de nuevo empleando el for.

Programa: 

```
"""
Mostrar la tabla de 5 con las estructuras repetitivas:
    while
    y
    for
"""

#utilizando el while
x=5
while x<=50:
    print("Tabla del 5 empleando el while")
    print(x)
    x=x+5
```



```
#utilizando el for
for x in range(5,51,5):
    print(x)
print("Tabla del 5 empleando el for")
```




Desempeño 34

Soluciones



Confeccionar un programa que solicite la carga de 10 valores flotantes (con parte decimal) por teclado. Mostrar al final su suma.

Definir varias líneas de comentarios indicando el nombre del programa, el programador y la fecha de la última modificación.

Utilizar el caracter # para los comentarios.

Como te habrás dado cuenta, el uso de estructuras condicionales sumado a los ciclos repetitivos, despliegan la verdadera potencia y eficiencia de la programación. Y también, el uso de comentarios prepara nuestro código para ser mantenido en las futuras revisiones.

En la clase que viene, seguimos agregando complejidad, así que no dejes ninguno de los ejercicios y ejemplos sin resolver.

¡Seguimos avanzando!

Créditos

Imágenes

Encabezado: Image by Steve Buissinne from Pixabay

<https://pixabay.com/photos/pawn-chess-pieces-strategy-chess-2430046/>

Tipografía

Para este diseño se utilizó la tipografía *Source Sans Pro* diseñada por Paul D. Hunt.

Extraída de Google Fonts.

Si detectás un error del tipo que fuere (falta un punto, un acento, una palabra mal escrita, un error en código, etc.), por favor comunicate con nosotros a correcciones@issd.edu.ar e indicanos por cada error que detectes la página y el párrafo. Muchas gracias por tu aporte.

Soluciones a los problemas

Te recomendamos utilizar esta sección luego de haber intentado por un largo tiempo la resolución y también para verificar tus soluciones.

Solución al desempeño 26

| Programa: 

```
n=int(input("Cuantos triángulos procesará:"))
cantidad=0
for x in range(n):
    basetri=int(input("Ingrese el valor de la base:"))
    altura=int(input("Ingrese el valor de la altura:"))
    superficie=basetri*altura/2
    print("La superficie es")
    print(superficie)
    if superficie>12:
        cantidad=cantidad+1
print("La cantidad de triángulos con superficie superior a 12 son")
print(cantidad)
```

Solución al desempeño 27

| Programa: ⬇

```
suma=0
for f in range(10):
    valor=int(input("Ingrese un valor:"))
    if f>4:
        suma=suma+valor
print("La suma de los últimos 5 valores es")
print(suma)
```


Solución al desempeño 28

| Programa: ⌵

```
for f in range(5,51,5):  
    print(f)
```

Solución al desempeño 29

| Programa: ⌵

```
valor=int(input("Ingrese un valor entre 1 y 10:"))  
for f in range(1,13):  
    tabla=valor*f  
    print(tabla)
```

Solución al desempeño 30

| Programa: ⌵

```
cant1=0
cant2=0
cant3=0
n=int(input("Ingrese la cantidad de triángulos:"))
for f in range(n):
    lado1=int(input("Ingrese lado 1:"))
    lado2=int(input("Ingrese lado 2:"))
    lado3=int(input("Ingrese lado 3:"))
    if lado1==lado2 and lado1==lado3:
        print("Es un triángulo equilatero.")
        cant1=cant1+1
    else:
        if lado1==lado2 or lado1==lado2 or lado2==lado3:
            print("Es un triángulo isósceles.")
            cant2=cant2+1
        else:
            print("Es un triángulo escaleno.")
            cant3=cant3+1
print("Cantidad de triángulos equilateros:")
print(cant1)
```



```
print("Cantidad de triángulos isósceles:")
print(cant2)
print("Cantidad de triángulos escalenos:")
print(cant3)
```

Solución al desempeño 31

| Programa: ⌵

```
cant1=0
cant2=0
cant3=0
cant4=0
n=int(input("Cantidad de puntos:"))
for f in range(n):
    x=int(input("Ingrese coordenada x:"))
    y=int(input("Ingrese coordenada y:"))
    if x>0 and y>0:
        cant1=cant1+1
    else:
        if x<0 and y>0:
            cant2=cant2+1
```

```
    else:
        if x<0 and y<0:
            cant3=cant3+1
        else:
            if x>0 and y<0:
                cant4=cant4+1
print("Cantidad de puntos en el primer cuadrante:")
print(cant1)
print("Cantidad de puntos en el segundo cuadrante:")
print(cant2)
print("Cantidad de puntos en el tercer cuadrante:")
print(cant3)
print("Cantidad de puntos en el cuarto cuadrante:")
print(cant4)
```

Solución al desempeño 32

| Programa: ⌵


```
negativos=0
positivos=0
mult15=0
sumapares=0
```

```
for f in range(10):
    valor=int(input("Ingrese valor:"))
    if valor<0:
        negativos=negativos+1
    else:
        if valor>0:
            positivos=positivos+1
        if valor%15==0:
            mult15=mult15+1
        if valor%2==0:
            sumapares=sumapares+valor
print("Cantidad de valores negativos:")
print(negativos)
print("Cantidad de valores positivos:")
print(positivos)
print("Cantidad de valores múltiplos de 15:")
print(mult15)
print("Suma de los valores pares:")
print(sumapares)
```


Solución al desempeño 33

Volver a la clase



| Programa: 

```
suma1=0
suma2=0
suma3=0
for f in range(5):
    edad=int(input("Ingrese edad:"))
    suma1=suma1+edad
pro1=suma1/5
print("Promedio de edades del turno mañana:")
print(pro1)
for f in range(6):
    edad=int(input("Ingrese edad:"))
    suma2=suma2+edad
pro2=suma2/6
print("Promedio de edades del turno tarde:")
print(pro2)
for f in range(11):
    edad=int(input("Ingrese edad:"))
    suma3=suma3+edad
pro3=suma3/11
print("Promedio de edades del turno noche:")
```



```
print(pro3)
if pro1<pro2 and pro1<pro3:
    print("El turno mañana tiene un promedio menor de edades.")
else:
    if pro2<pro3:
        print("El turno tarde tiene un promedio menor de edades.")
    else:
        print("El turno noche tiene un promedio menor de edades.")
```

[Volver a la clase](#)



Solución al desempeño 34

| Programa: 

```
#Programa: Carga de 10 Numeros
#Programador: Paz Marcos
#Fecha de última modificación: 28/06/2018

suma=0.0
for x in range(10):
    valor=float(input("Ingrese valor:"))
    suma=suma+valor
print("La suma de los 10 numeros es")
print(suma)
```