Guía de ejercicios resueltos
Algoritmos y programación

Anggelo Urso G.
anggelo.urso@inacapmail.cl

23 de junio de 2020

1 Introducción

El siguiente documento tiene por finalidad generar una guía de ejercicios resueltos para facilitar el aprendizaje de los alumnos de la asignatura de algoritmos y programación.

2 Ejercicios resueltos

A continuación se desarrollarán una serie de ejercicios resueltos de acuerdo a distintas secciones de interés en la materia.

2.1 Entrada y salida de datos

Ejercicio 1. Muestre por pantalla el siguiente texto: ”Hola mundo”.

R. Para desarrollar este ejercicio utilizaremos la función **print**, luego nuestro código quedaría como:

```
1 print("Hola mundo")
```

En este caso, observamos que pasamos como argumento el **string** ”Hola mundo” y luego se muestra por pantalla.

Ejercicio 2. Leer un string desde teclado.

R. Para desarrollar este ejercicio utilizaremos la función **input**. Esta función lee desde el teclado los datos.

```
1 input()
```

Luego si queremos mejorar un poco la entrada de los datos, agregando un texto antes para indicar al usuario que es lo que pedimos, la función **input** recibe como argumento un **string**, el cual se despliega por pantalla y solicita el texto.

```
1 input("Escriba una palabra: ")
```

Finalmente si queremos almacenar el valor recibido por teclado en una variable para posteriormente utilizarlo, usamos:

```
1 texto = input("Escriba una palabra: ")
```

Ejercicio 3. Leer un string desde el teclado y luego mostrarlo por pantalla.

R. Para este ejercicio usaremos **print** e **input**, a través de la segunda capturaremos el **string** desde teclado y con la primera lo mostraremos por pantalla.

```
1 palabra = input("Escriba una palabra: ")
2 print(palabra)
```

Prueba tu código con la palabra *Hola*. Debería mostrarte la palabra por pantalla.

Ejercicio 4. Leer un numero desde teclado.

R. Primero es necesario entender que la función **input**, lo que nos retorna al flujo del programa es un string. Para poder devolver un **int** al flujo, es necesario hacer un *cast* de la variable, es decir. Si queremos un entero, debemos forzar el tipo a la función input.

```
1 # esto retorna un string
2 variable = input("Ingrese un numero: ")
3
4 # Para que sea un número es necesario
5 variable = int(input("Ingrese un numero: "))
```

Importante: si el dato que estamos ingresando por teclado no es un entero (ej. un flotante o un string) esto nos arrojará un error. Esto debido a que estamos forzando el tipo de dato a entero si o si.

Ejercicio 5. Sumar dos números recibidos desde teclado y mostrar el resultado por pantalla.

R. Para esto usaremos la función **input** y asignación de variables. Luego sumaremos el resultado y usaremos la función **print** para mostrar el resultado por pantalla.

```
1 num1 = int(input("Ingrese numero 1: "))
2 num2 = int(input("Ingrese numero 2: "))
3
4 resultado = num1 + num2
5 print(resultado)
```

Prueba tu programa escribiendo 3 y 5, el resultado por pantalla debiese ser 8.

Ejercicio 6. Recibir dos números flotante por teclado y mostrar la suma por pantalla sin usar una variable para la suma.

R. Tal y como vimos en el ejercicio 4 para determinar el tipo de dato de la función **input** debemos forzarlo. Luego nos piden mostrar por teclado sin usar una variable para la suma, esto lo podemos hacer con la función **print** pasando directamente la suma de los números.

```
1 num1 = float(input("Escriba el primer flotante: "))
2 num2 = float(input("Escriba el segundo flotante: "))
3
4 # similar a: suma = num1 + num2 y luego print(suma)
5 print(num1 + num2)
```

Prueba tu programa con 3 y 5, el resultado debiese ser 8.0. Otra opción, probar con 3.0 y 5.0, como resultado también debiese ser 8.0.

Importante: Acá al igual que el ejercicio 4 estamos esperando un tipo de dato flotante. Si pasamos por error un **string** nos arrojará un error. No así si pasamos un entero, dado a que un entero "cabe" en un flotante. Más específicamente, un número flotante es más general que un entero, y el conjunto de los número enteros está contenido en los flotantes.[1]

$$\mathbb{Z} \subset \mathbb{Q}$$

Ejercicio 7. Recibir un nombre desde el teclado y luego mostrar un saludo personalizado a ese nombre.

[1] Esto se lee, los números enteros $\mathbb{Z}$ son un subconjunto de los números racionales (flotantes) $\mathbb{Q}$

R. Guardamos el dato recibido por teclado en una variable y luego mostramos *"Hola "* junto con el nombre recibido por teclado.

Acá tenemos dos alternativas, la primera es pasar el saludo, y separado por una coma, el nombre ingresado por teclado. La segunda es concatenar (sumar los dos **strings**) en la función **print**.

```
1 nombre = input("Ingrese su nombre: ")
2
3 # Alternativa 1
4 print("Hola ", nombre)
5
6 # Alternativa 2
7 print("Hola " + nombre)
```

Si pruebas tu programa con el nombre *Pedro* debiese retornar: **Hola Pedro** dos veces con este código (una vez si solo usaste una de las dos alternativas).

Ejercicio 8. Pedir el nombre al usuario por teclado y luego solicitar el apellido (también por teclado), mostrando el nombre en la entrada de datos. Luego saludar a la persona.

R. Una de las cosas bonitas que tiene la función **input** es que podemos concatenar **strings** recibidos anteriormente en otro **input**. De esta manera podemos hacer un código de las siguiente forma.

```
1 nombre = input("Ingrese su nombre: ")
2 apellido = input("Hola " + nombre + ", cuál es su apellido: ")
3
4 print("Mucho gusto " + nombre + " " + apellido)
```

En este caso si ingresamos como nombre *Pedro* y como apellido *González*, el programa nos debería mostrar: **Hola Pedro, cuál es su apellido**, seguido por: **Mucho gusto Pedro González**

Ejercicio 9. Resolver el ejercicio 8 ahora con texto formateado.

R. Cuando nos piden texto formateado, anteponemos una **f** antes de la cadena y luego, a través de paréntesis de llave, vamos escribiendo las variables.

```python
nombre = input("Ingrese su nombre: ")
apellido = input(f"Hola {nombre}, cuál es su apellido: ")

# la f nos dice que el texto está con formato
print(f"Mucho gusto {nombre} {apellido}")
```

En este caso si ingresamos como nombre *Pedro* y como apellido *González*, el programa nos debería mostrar: **Hola Pedro, cuál es su apellido**, seguido por: **Mucho gusto Pedro González**

A contar de ahora en adelante, si queremos un texto *con formato* usaremos esta forma para representarlo (importante la **f** y la forma de escritura).

Ejercicio 10. Diseñe un programa que a partir del valor del lado de un cuadrado (3 metros), muestre el valor de su perímetro (en metros) y el de su área (en metros cuadrados).

R. Sabemos que el perímetro de un cuadrado es la suma de sus cuatro lados, o similar:

$$perimetro = lado * 4$$

Y que el área es calcular el cuadrado de su lado:

$$area = lado^2$$

Con los datos que nos entregan, el perímetro debe darnos 12 metros y el área 9 metros cuadrados. Luego un programa que calcule el perímetro, con un valor de lado dado, sería algo como esto:

```python
lado = 3

perimetro = lado * 4
area = lado ** 2

print(f"El perimetro es {perimetro} metros")
print(f"El área es {area} metros cuadrados")
```

Ejercicio 11. Diseñe un programa que, a partir del valor de la base y de la altura de un triángulo (3 y 5 metros respectivamente), muestre el valor de su área (en metros cuadrados).

R. Recordamos que el cálculo del área de un triangulo viene dado por:

$$area = \frac{base * altura}{2}$$

Luego con los datos que nos entregan el área del triángulo debiese tener un valor de 7.5 metros cuadrados. Si hacemos un programa en Python que nos haga dicho cálculo, tenemos:

```python
base = 3
altura = 5
area = (base * altura) / 2

print(f"El área es {area} metros cuadrados")
```

Ejercicio 12. Calcule el área de una circunferencia, pidiendo el radio de ésta al usuario para que lo ingrese por teclado.

R. El cálculo del área de una circunferencia está dado por:

$$area = \pi * radio^2$$

Python nos ofrece la librería **math** para obtener desde ella distintos tipos de funciones o constantes (entre ellas el valor de **pi**). Para usarla solo basta importarla y usarla directamente.

```python
# importamos la libreria math
import math

radio = int(input("Ingrese el radio de la circunferencia: "))

# acá llamamos a pi desde la libreria
area = math.pi * radio ** 2

print(f"El área de la circunferencia es: {area}")
```

Podemos probar el programa, ingresando el valor del radio en 3, lo cual nos daría como resultado: **El área de la circunferencia es: 28.274333882308138**

Ejercicio 13. Calcular el volumen de una esfera, solicitando al usuario el ingreso del radio de ésta por teclado.

R. Recordando la fórmula

$$volumen = \frac{4}{3} * \pi * radio^3$$

Luego usamos **input** para capturar el valor del radio desde el teclado y llevamos la formula a nuestro programa, e importamos la librería **math** para obtener el valor de **pi**.

```python
import math

radio = int(input("Ingrese el radio: "))
volumen = 4 / 3 * math.pi * radio ** 3
print(f"Volumen de esfera es: {volumen}")
```

Podemos probar el programa, ingresando el valor del radio en 3, lo cual nos daría como resultado: **Volumen de esfera es: 113.09733552923254**

Ejercicio 14. Diseña un programa que pida el valor de los tres lados de un triángulo y calcule el valor de su área y perímetro.

R. Si nos recordamos, el área de un triángulo en base a sus lados (a, b, c) se calcula como:

$$area = \sqrt{s(s-a)(s-b)(s-c)}$$

Donde:

$$s = \frac{a+b+c}{2}$$

Acá usaremos nuevamente la librería **math** para poder usar la función de raíz cuadrada (**sqrt**). Escribiendo nuestro programa en Python, obtenemos lo siguiente:

```python
import math

a = int(input("Ingrese el lado a: "))
b = int(input("Ingrese el lado b: "))
c = int(input("Ingrese el lado c: "))

# calculamos s
s = (a + b + c) / 2

# calculamos el área
area = math.sqrt(s * (s - a) * (s - b) * (s - c))

# calculamos el perímetro
perimetro = a + b + c

# mostramos resultados por pantalla
print(f"El área es {area}")
print(f"El perímetro es {perimetro}")
```

Si probamos con lados de 3, 5 y 7; el resultado debiese ser: **El área es 6.49519052838329** y **El perímetro es 15**.

Ejercicio 15. Diseñe un programa que calcule el área de un triángulo a partir del valor de dos de sus lados y el ángulo entre ellos (en grados), mostrando por pantalla el resultado.

R. Nuevamente nos vamos a matemáticas:

$$area = \frac{1}{2} a * b * \sin(\theta)$$

Siendo **a** y **b** los lados y θ el ángulo entre ellos (en grados). Un punto importante, usaremos la librería **math** para poder usar la función **sin**. Sin embargo esta función calcula el seno de un ángulo, pero este debe estar en radianes (no en grados). Sin embargo la transformación de un ángulo en grados a radianes es bastante simple, sabiendo que $\pi = 180$, luego aplicamos regla de 3 simple y tenemos:

$$radianes = grados * \frac{\pi}{180}$$

Así nuestro código nos quedaría como:

```python
import math

a = int(input("Ingrese lado a: "))
b = int(input("Ingrese lado b: "))
angulo = int(input("Ingrese el ángulo (en grados): "))

# transformamos el ángulo de grados a radianes
radianes = angulo * math.pi / 180

area = 1 / 2 * a * b * math.sin(radianes)
print(f"El area es {area}")
```

Si probamos nuestro código con **a=1**, **b=2** y **ángulo=30**, el resultado debiese ser: **El área es 0.4999999999999994** o aproximado a **0.5**

Ejercicio 16. Diseñe un programa que pida al usuario una cantidad de pesos, una tasa de interés anual y un número de años y calcule cuanto habrá convertido el capital inicial transcurrido esos años, si a cada año se le calcula la tasa tasa de interés (interés compuesto).

R. Nos vamos a nuestra fórmula de interés compuesto (con tasa de interés en porcentaje):

$$monto = C * (1 + \frac{i}{100})^n$$

Con **C** el capital, **i** la tasa de interés anual y **n** periodo en años. Luego si llevamos esto a nuestro programa en Python tenemos:

```python
cantidad = int(input("Ingrese cantidad: "))
tasa = float(input("Ingrese tasa de interés: "))
periodo = int(input("Ingrese periodo (años): "))

valor = cantidad * (1 + tasa/100)**periodo
print(f"El valor final en {periodo} años es de: {valor}")
```

Si probamos nuestro código con: **capital=10000**, **interés=4.5** y **periodo=20**; debiésemos obtener como resultado: **El valor final en 20 años es de: 24117.14**.

Acá debemos notar que capital y periodo son enteros, por ende forzamos el tipo de la función **input** en entero; y que el valor de interés es un número flotante, así que forzamos el tipo a flotante.

2.2 Ejercicios con tortuga

En los ejercicios de la siguiente sección utilizaremos el módulo **turtle** que se encuentra en Python. Cada ejercicio importará el módulo, así que presten atención a los ejemplos.

Además mencionar que utilizaremos una versión un poco diferente de lo que vimos en clases. En particular utilizaremos y definiremos la pantalla en la cual trabajaremos, así si observan el siguiente código:

```
1 from turtle import Screen, Turtle
2
3 pantalla = Screen()
4 pantalla.setup(425, 225)
5 pantalla.screensize(400, 200)
6
7 pantalla.exitonclick()
```

Observamos lo siguiente:

- Importamos los elementos usando **from turtle import Screen, Turtle**. Esto lo realizamos para importar solo esas funciones de la librería turtle (y no traernos todo). En caso de que necesitemos todo, el ejercicio nos dirá directamente **import turtle**.
- Creamos una pantalla a través de la función **Screen()** y definimos su tamaño utilizando la función **setup** (si necesitamos más grande o más chico acá vamos ajustando) de 425 píxeles de ancho y 225 píxeles de alto.
- En la línea **5** fijamos el tamaño de la superficie de dibujo en 400 píxeles de ancho y 200 píxeles de alto.
- A través de la función de la línea **7** controlamos que la pantalla de la tortuga se cierre una vez que demos clic sobre la interfaz

Entendiendo eso como esencial comenzamos con los ejercicios resueltos[2].

Ejercicio 1. Diseñe un programa que dibuje un cuadrado por la pantalla de lado 100.

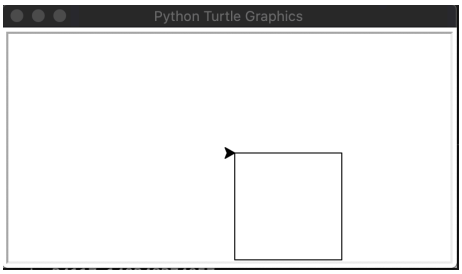
R. Para esto utilizaremos la función **fd** para mover nuestra tortuga 100 pasos hacia adelante y **rt** para girar a la derecha[3].

[2] Es recomendable comenzar a leer documentación del módulo de turtle

[3] **rt** es el nombre corto de **right**, al igual que **fd** es el nombre corto de **forward** o **lt** es de **left**

```python
from turtle import Screen, Turtle

pantalla = Screen()
pantalla.setup(425, 225)
pantalla.screensize(400, 200)

tortuga = Turtle()

# dibujamos el cuadrado
tortuga.fd(100)
tortuga.rt(90)
tortuga.fd(100)
tortuga.rt(90)
tortuga.fd(100)
tortuga.rt(90)
tortuga.fd(100)
tortuga.rt(90)

pantalla.exitonclick()
```

El resultado debiese entregar esta figura

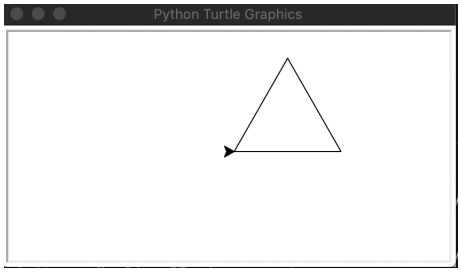


Ejercicio 2. Diseñe un programa que dibuje en la pantalla un triángulo equilátero de lado 100.

R. Para que sea un triángulo equilátero, todos sus lados miden lo mismo y los ángulos interiores son de 60°. Para esto usamos la función **lt** para hacer el triángulo hacia arriba. Además consideramos pasar 120 como valor a la función **lt** (diferencia entre 180 - 60). Así el código nos queda como:

```python
from turtle import Screen, Turtle

pantalla = Screen()
pantalla.setup(425, 225)
pantalla.screensize(400, 200)

t = Turtle()

# Dibujamos el triangulo
t.fd(100)
t.lt(120)
t.fd(100)
t.lt(120)
t.fd(100)
t.lt(120)

pantalla.exitonclick()
```

Luego deberíamos ver por pantalla:



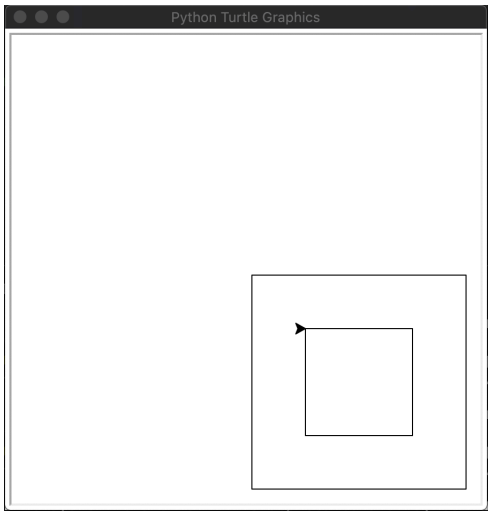
Ejercicio 3. Diseñe un programa que dibuje un cuadrado cuyos lados midan 200 pasos y otro cuadrado cuyos lados midan 100 centrado al interior del cuadrado más grande.

R. Para esto usaremos **up** y **down** que levantan el pincel y lo bajan respectivamente. Con esto podemos mover libremente la tortuga sin pintar sobre el lienzo.

Para posicionar a la tortuga dentro del cuadrado grande, levantaremos el pincel y nos moveremos 50 pasos hacia el frente y 50 pasos hacia abajo. Una vez en esa posición dibujaremos el segundo cuadrado y nos quedará centrado. Así el código quedaría:

```python
from turtle import Screen, Turtle

pantalla = Screen()
pantalla.setup(450, 450)
pantalla.screensize(425, 425)

t = Turtle()

# Primer cuadrado
t.fd(200)
t.rt(90)
t.fd(200)
t.rt(90)
t.fd(200)
t.rt(90)
t.fd(200)
t.rt(90)

# levantamos el pincel y movemos la tortuga al interior
t.up()
t.fd(50)
t.rt(90)
t.fd(50)
t.lt(90)
t.down()

# segundo cuadrado
t.fd(100)
t.rt(90)
t.fd(100)
t.rt(90)
t.fd(100)
t.rt(90)
t.fd(100)
t.rt(90)

pantalla.exitonclick()
```

Y obtendremos como resultado la siguiente figura:

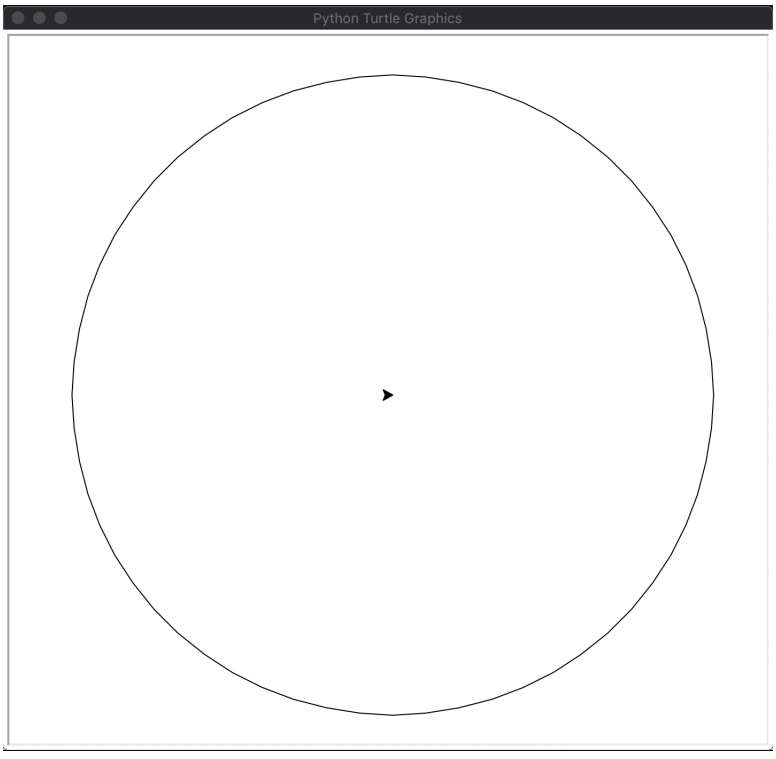


Ejercicio 4. Diseñar un programa que permita visualizar un gráfico de torta a través de la aplicación **turtle** en donde exista la siguiente distribución de porcentajes: suspensos 10%, 20% de aprobados, un 40% de notables y un 30% de sobresalientes.

R. Para desarrollar este programa, lo realizaremos paso a paso. Lo primero que se realizará es el círculo de nuestro gráfico de torta. Para esto definiremos una circunferencia de radio 300. Moveremos el pincel desde la posición *(0,0)* a la posición *(0,-300)* para dejar un gráfico centrado en la ventana.

```python
from turtle import Screen, Turtle

radio = 300

pantalla = Screen()
tortuga = Turtle()
tortuga.speed(0)

tortuga.up()
tortuga.goto(0, -radio)
tortuga.down()
tortuga.circle(radio)
tortuga.up()
tortuga.home()
tortuga.down()

pantalla.exitonclick()
```

Con esto tendremos una imagen de este tipo:



Con la función **tortuga.home()** movemos al pincel a la posición inicial *(0,0)*. Al levantar y bajar el pincel a conveniencia podemos graficar el círculo sin graficar todos los movimientos de éste al moverlo a los puntos de inicio del dibujo de la circunferencia.

Guardamos las calificaciones en variables y agregamos comentarios para mayor claridad de nuestro código, así el código nos quedaría de la siguiente forma:

```
1  from turtle import Screen, Turtle
2
3  # Calificaciones
4  suspensos = 10
5  aprobados = 20
6  notables = 40
7  sobresalientes = 30
8
9  # Radio de circunferencia
10 radio = 300
11
12 # Inicialización
13 pantalla = Screen()
14 tortuga = Turtle()
15 tortuga.speed(0)
16
17 # Dibujo de circulo exterior
18 tortuga.up()
19 tortuga.goto(0, -radio)
20 tortuga.down()
21 tortuga.circle(radio)
22 tortuga.up()
23 tortuga.home()
24 tortuga.down()
25
26 # Salir cuando damos click sobre la ventana
27 pantalla.exitonclick()
```

Luego si queremos dibujar las líneas divisoras de las calificaciones, tendremos que calcular los ángulos necesarios de cada una de las calificaciones. Para esto usaremos la fórmula:

$$ngulo_{calificacin} = \frac{360 * tipo_{nota}}{100}$$

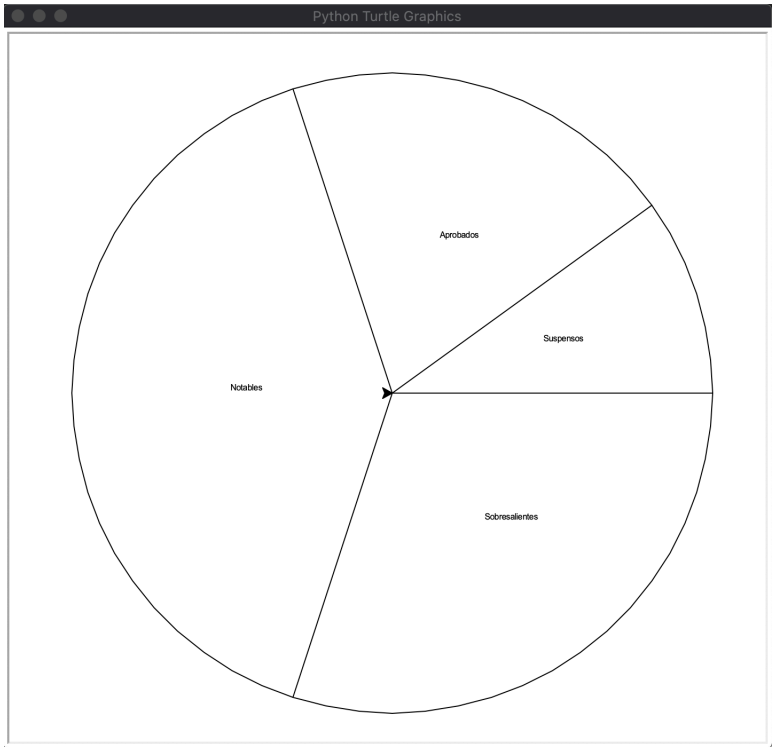
Así haciendo el lineamiento por cada uno de los porcentajes de

notas, tenemos el siguiente código:

```
from turtle import Screen, Turtle

# Calificaciones
suspensos = 10
aprobados = 20
notables = 40
sobresalientes = 30

# Radio de circunferencia
radio = 300

# Inicialización
pantalla = Screen()
tortuga = Turtle()
tortuga.speed(0)

# Dibujo de circulo exterior
tortuga.up()
tortuga.goto(0, -radio)
tortuga.down()
tortuga.circle(radio)
tortuga.up()
tortuga.home()
tortuga.down()

# Dibujo linea de suspensos
angulo = 360 * suspensos / 100
tortuga.lt(angulo)
tortuga.fd(radio)
tortuga.bk(radio)

# Escribir texto para los suspensos
tortuga.up()
tortuga.rt(angulo/2)
tortuga.fd(radio/2)
tortuga.write('Suspensos')
tortuga.bk(radio/2)
tortuga.lt(angulo / 2)
tortuga.down()
```

```
40 # Dibujo de línea para los aprobados
41 angulo = 360 * aprobados / 100
42 tortuga.lt(angulo)
43 tortuga.fd(radio)
44 tortuga.bk(radio)
45
46 # Escribir texto para los aprobados
47 tortuga.up()
48 tortuga.rt(angulo/2)
49 tortuga.fd(radio/2)
50 tortuga.write('Aprobados')
51 tortuga.bk(radio/2)
52 tortuga.lt(angulo/2)
53 tortuga.down()
54
55 # Dibujo de línea para los notables
56 angulo = 360 * notables  / 100
57 tortuga.lt(angulo)
58 tortuga.fd(radio)
59 tortuga.bk(radio)
60
61 # Textos para los notables
62 tortuga.up()
63 tortuga.rt(angulo/2)
64 tortuga.fd(radio/2)
65 tortuga.write('Notables')
66 tortuga.bk(radio/2)
67 tortuga.lt(angulo/2)
68 tortuga.down()
69
70 # Dibujo a los sobresalientes
71 angulo = 360 * sobresalientes / 100
72 tortuga.lt(angulo)
73 tortuga.fd(radio)
74 tortuga.bk(radio)
75
76 # Escribir el texto para los sobresalientes
77 tortuga.up()
78 tortuga.rt(angulo/2)
79 tortuga.fd(radio/2)
80 tortuga.write('Sobresalientes')
81 tortuga.bk(radio/2)
82 tortuga.lt(angulo/2)
83 tortuga.down()
84
85 # Salir cuando damos click sobre la ventana
86 pantalla.exitonclick()
```

Acá observamos también la función **write** que nos permite escribir un texto en el gráfico. Así nos quedaría el gráfico de la siguiente manera:



2.3 Sentecias de control

A continuación realizaremos ejercicios con las sentencias de control de flujo **IF**, **ELSE**, **ELIF**. Iremos construyendo ejercicios en orden de relevancia y avanzando desde ejercicios simples a más complejos.

Ejercicio 1. Diseñe un programa que reciba la edad de una persona y que muestre por pantalla si es mayor de edad.

R. Usamos la sentencia **IF** para evaluar la condición: $edad > 18$. Si esta es verdad (**True**) entonces nos va a mostrar el mensaje, en caso contrario, no va a mostrar nada. El código quedaría como:

```python
edad = int(input("Ingrese la edad: "))

if edad > 18:
    print("Es mayor de edad")
```

Ejercicio 2. Rediseñe el programa del ejercicio 1 para incluir un mensaje que indique que la persona no es mayor de edad.

R. Para esto reformularemos el programa anterior y usaremos la sentencia **ELSE**. Esta sentencia se ejecutará solo cuando la condición $edad > 18$ sea falsa (**False**).

```python
edad = int(input("Ingrese la edad: "))

if edad > 18:
    print("Es mayor de edad")
else:
    print("Es menor de edad")
```

Ejercicio 3. Solicitar la edad de dos personas por teclado e indicar que persona es mayor, menor o si tienen la misma edad.

R. Para esto usaremos la sentencia **IF** y la sentencia **ELSE** para validar quien es mayor. Para esto compararemos $edad_{p1} > edad_{p2}$, si es verdad entonces la persona 1 es mayor que la persona 2, pero si es falsa se nos abren dos posibilidades:

(a) Que ambas personas tengan la misma edad

(b) Que la persona 2 es mayor que la persona 1

Con esto en mente, tenemos que en la sentencia **ELSE** considerar ambos escenarios. Así podemos considerar esto como otra sentencia de control de flujo.

```python
edad_1 = int(input("Ingrese la edad de la persona 1: "))
edad_2 = int(input("Ingrese la edad de la persona 2: "))

if edad_1 > edad_2:
    print("La persona 1 es mayor")
else:
    if edad_1 < edad_2:
        print("La persona 2 es mayor")
    else:
        print("Las personas tienen la misma edad")
```

Ejercicio 4. Diseñe un programa que determine si un número ingresado por el usuario es par o impar y muestre un mensaje según corresponda.

R. Para esto utilizaremos la operación módulo (%). Esta operación nos entrega el resto de una división. Si esta es exacta el resultado de la operación es 0, si la división no es exacta, nos entregará el resto de la división. Para determinar si el número es par o no, simplemente tenemos que calcular la operación módulo de 2 con el número. Si este es par, el resultado será cero, en caso contrario entregará 1. Así nuestro código quedará como:

```python
numero = int(input("Ingrese un número: "))

if numero % 2 == 0:
    print(f"El {numero} es par")
else:
    print(f"El {numero} es impar")
```

Así, si ingresamos el número **8**, el programa nos entregará: **El número 8 es par**; si ahora probamos con el número **15**, el programa nos entregará: **El número 15 es impar**.

Ejercicio 5. Diseñe un programa usando el módulo **turtle**, que en base a 3 opciones, presentadas en un menú, dibuje: un cuadrado, un triángulo o un círculo.

R. Si definimos que nuestras opciones son:

(a) cuadrado

(b) triángulo

(c) circulo

Tenemos que crear el menú y obtener la opción escogida por el usuario desde el teclado, y en base a esa opción dibujar. En este caso podríamos observar como quedaría el código:

```
from turtle import Screen, Turtle

pantalla = Screen()
pantalla.setup(425, 225)
pantalla.screensize(400, 200)

turtle = Turtle()

print("Escoja una opción:")
print("\t 1.- Cuadrado")
print("\t 2.- Triángulo")
print("\t 3.- Círculo")

opcion = int(input("Ingrese opcion: "))

if opcion == 1:
    turtle.fd(100)
    turtle.lt(90)
    turtle.fd(100)
    turtle.lt(90)
    turtle.fd(100)
    turtle.lt(90)
    turtle.fd(100)
    turtle.lt(90)
else:
    if opcion == 2:
        turtle.fd(100)
        turtle.lt(120)
        turtle.fd(100)
        turtle.lt(120)
        turtle.fd(100)
        turtle.lt(120)
    else:
        turtle.circle(40)

pantalla.exitonclick()
```

Si observamos, capturamos la opción del usuario y según la opción

escogida mostramos una figura u otra. Si mejoramos un poco más el código usando **string** multilinea (ahorrándonos los 4 **print**) y por otro lado usamos la sentencia **ELIF** por el sangrado, nos queda un código un poco más limpio:

```
1  from turtle import Screen, Turtle
2
3  pantalla = Screen()
4  pantalla.setup(425, 225)
5  pantalla.screensize(400, 200)
6
7  turtle = Turtle()
8
9  print("""Escoja una opción:
10         \t 1.- Cuadrado
11         \t 2.- Triángulo
12         \t 3.- Círculo""")
13
14 opcion = int(input("Ingrese opcion: "))
15
16 if opcion == 1:
17     turtle.fd(100)
18     turtle.lt(90)
19     turtle.fd(100)
20     turtle.lt(90)
21     turtle.fd(100)
22     turtle.lt(90)
23     turtle.fd(100)
24     turtle.lt(90)
25 elif opcion == 2:
26     turtle.fd(100)
27     turtle.lt(120)
28     turtle.fd(100)
29     turtle.lt(120)
30     turtle.fd(100)
31     turtle.lt(120)
32 elif opcion == 3:
33     turtle.circle(40)
34 else:
35     print("Opción ingresada es incorrecta")
36
37 pantalla.exitonclick()
```

Ejercicio 6. Diseñe un programa que permita calcular el valor de **x** de la siguiente ecuación. Los valores de a y b serán ingresados por teclado.

$$ax + b = 0$$

R. Para resolver esta ecuación, debemos despejar la x. Esto lo hacemos moviendo los términos hacia el otro lado de la ecuación, quedándonos de la siguiente manera:

$$x = \frac{-b}{a}$$

```python
a = int(input("Ingrese el valor de a: "))
b = int(input("Ingrese el valor de b: "))

x = -b / a

print(f"El valor de x es {x}")
```

Luego, acá se nos presenta un problema. Si el usuario ingresa un valor de $a = 0$, esta ecuación nos entrega un error de división por 0. Esto se produce dado a que la ecuación no tiene valores posibles para resolverse.

Otro escenario que es posible apreciar acá es cuando el usuario ingresa un valor de $a = 0$ y $b = 0$. En ese caso, la ecuación tiene infinitas soluciones. Sin embargo esos dos casos no los estamos graficando correctamente en nuestro código, para eso tenemos que considerar ambas restricciones y colocar los respectivos mensajes, quedando nuestro código de la siguiente manera:

```python
a = int(input("Ingrese el valor de a: "))
b = int(input("Ingrese el valor de b: "))

x = -b / a

if a != 0:
    print(f"El valor de x es {x}")
else:
    if b == 0:
        print("La ecuación tiene infinitas soluciones")
    else:
        print("La ecuación no tiene solución")
```

Ejercicio 7. Diseñe un programa que permita calcular los valores que toma x en una ecuación de segundo grado, en donde los valores de a, b y c son ingresados por el usuario por teclado. Recordar que para obtener los valores de x, se debe resolver la siguiente ecuación:

$$x = \frac{-b \pm \sqrt{b^2 - 4 * a * c}}{2 * a}$$

R. Para este ejercicio lo iremos desarrollando por partes.

Lo primero que debemos validar, son las entradas de los valores de a, b y c.

Si observamos la fórmula anterior, observamos que el primer problema que podemos tener es que el usuario ingrese un valor de $a = 0$. Con esto el cálculo de los valores de x se indefine, dado a que me quedaría una división por 0. Por otro lado, si observamos como planteamos una ecuación de segundo grado:

$$ax^2 + bx + c = 0$$

Por ende, si $a = 0$, entonces se nos transforma a una ecuación de primer grado, quedando como:

$$bx + c = 0$$

Despejando x, nos queda:

$$x = \frac{-c}{b}$$

Para este escenario podemos tener una primera versión de código de la siguiente manera:

```python
from math import sqrt

a = int(input("Ingrese el valor de a: "))
b = int(input("Ingrese el valor de b: "))
c = int(input("Ingrese el valor de c: "))

if a != 0:
    x_1 = (-b + sqrt(b**2 - 4*a*c)) / 2 * a
    x_2 = (-b - sqrt(b**2 - 4*a*c)) / 2 * a
    print(f"El resultado es: x_1={x_1}, x_2={x_2}")
else:
    x = -c / b
    print(f"El valor de x es {x}")

```

Bien, salvamos el primer problema (pero no el único). ¿Qué pasa ahora si el usuario ingresa $a = 0$ y $b = 0$?. Nuevamente estamos frente a un potencial error de división por 0. Para esto añadiremos más sentencias de control, tal y como hicimos cuando resolvimos la ecuación de primer grado.

```
1  from math import sqrt
2
3  a = int(input("Ingrese el valor de a: "))
4  b = int(input("Ingrese el valor de b: "))
5  c = int(input("Ingrese el valor de c: "))
6
7  if a != 0:
8      x_1 = (-b + sqrt(b**2 - 4*a*c)) / 2 * a
9      x_2 = (-b - sqrt(b**2 - 4*a*c)) / 2 * a
10     print(f"El resultado es: x_1={x_1}, x_2={x_2}")
11 else:
12     if b != 0:
13         x = -c / b
14         print(f"El valor de x es {x}")
15     else:
16         if c != 0:
17             print("La ecuación no tiene solución")
18         else:
19             print("La ecuación tiene infinitas soluciones")
```

Como los niveles de indentación (sangrado) nos van quedando incómodos, usamos la sentencia **ELIF** para agrupar mejor, quedando un código como:

```python
from math import sqrt

a = int(input("Ingrese el valor de a: "))
b = int(input("Ingrese el valor de b: "))
c = int(input("Ingrese el valor de c: "))

if a != 0:
    x_1 = (-b + sqrt(b**2 - 4*a*c)) / 2 * a
    x_2 = (-b - sqrt(b**2 - 4*a*c)) / 2 * a
    print(f"El resultado es: x_1={x_1}, x_2={x_2}")
elif b != 0:
    x = -c / b
    print(f"El valor de x es {x}")
elif c != 0:
    print("La ecuación no tiene solución")
else:
    print("La ecuación tiene infinitas soluciones")
```

Ok, hasta este punto podemos calcular los valores de x reales, pero seguimos teniendo un problema, ¿qué pasa si el discriminante es negativo?.

Determinamos al discriminante de la ecuación de segundo grado al cálculo que realizamos dentro de la raíz cuadrada:

$$\Delta = b^2 - 4 * a * c$$

El problema es que si este valor es negativo la ecuación no tendrá soluciones reales. Por otro lado, la librería **math**, y en particular la función **sqrt** no permite resolver raíces cuadradas negativas, entonces debemos realizar esta validación.

```python
from math import sqrt

a = int(input("Ingrese el valor de a: "))
b = int(input("Ingrese el valor de b: "))
c = int(input("Ingrese el valor de c: "))

if a != 0:
    discriminante = b**2 - 4 * a * c

    if discriminante > 0:
        x_1 = (-b + sqrt(discriminante)) / 2 * a
        x_2 = (-b - sqrt(discriminante)) / 2 * a
        print(f"El resultado es: x_1={x_1}, x_2={x_2}")
    else:
        print("La ecuación no tiene soluciones reales")
elif b != 0:
    x = -c / b
    print(f"El valor de x es {x}")
elif c != 0:
    print("La ecuación no tiene solución")
else:
    print("La ecuación tiene infinitas soluciones")
```

Hasta este punto, tenemos nuestra ecuación validada y podemos resolver los valores de x siempre que estos sean reales, en caso contrario nos mostrará por pantalla: **La ecuación no tiene soluciones reales**.

Pero nuevamente dejamos por fuera un escenario más. Que sucede si me interesa saber las soluciones *complejas* de la ecuación, al igual que las soluciones reales, ¿cómo resolvemos eso?.

Para resolver el problema, hay un truco sumamente simple y útil que podemos usar:

$$\sqrt{a * b} = \sqrt{a} * \sqrt{b}$$

Luego, si tenemos por ejemplo:

$$\sqrt{-5} = \sqrt{5} * \sqrt{-1}$$

Y por definición de números complejos:

$$j = \sqrt{-1}$$

Luego:

$$\sqrt{-5} = \sqrt{5} * j$$

Por ende si el discriminante es negativo (señalizado acá como Δ), podemos hacer lo siguiente:

$$\sqrt{-\Delta} = \sqrt{\Delta} * j$$

Quedándonos la ecuación como:

$$x = \frac{-b \pm \sqrt{\Delta} j}{2 * a} = \frac{-b}{2 * a} \pm \frac{\sqrt{\Delta}}{2 * a} j$$

Con esto claro, colocamos la última modificación de código:

```python
from math import sqrt

a = int(input("Ingrese el valor de a: "))
b = int(input("Ingrese el valor de b: "))
c = int(input("Ingrese el valor de c: "))

if a != 0:
    discriminante = b**2 - 4 * a * c
    det = 2 * a

    if discriminante > 0:
        x_1 = (-b + sqrt(discriminante)) / det
        x_2 = (-b - sqrt(discriminante)) / det
    else:
        discriminante = -discriminante
        raiz = sqrt(discriminante)
        x_1 = complex(-b/(det), raiz/(det))
        x_2 = complex(-b/(det), -raiz/(det))

    print(f"El resultado es: x_1={x_1}, x_2={x_2}")
elif b != 0:
    x = -c / b
    print(f"El valor de x es {x}")
elif c != 0:
    print("La ecuación no tiene solución")
else:
    print("La ecuación tiene infinitas soluciones")
```

Por simple comodidad de código, hemos definido la variable $det = 2 * a$ para mayor claridad del código.

En la línea 15, lo que hacemos es invertir el valor del discriminante (agregando el menos delante) y así poder calcular la raíz de forma segura. Luego usando la función **complex** creamos el número complejo.

El primer parámetro de esa función es la parte real del número complejo, y el segundo parámetro es la parte compleja.