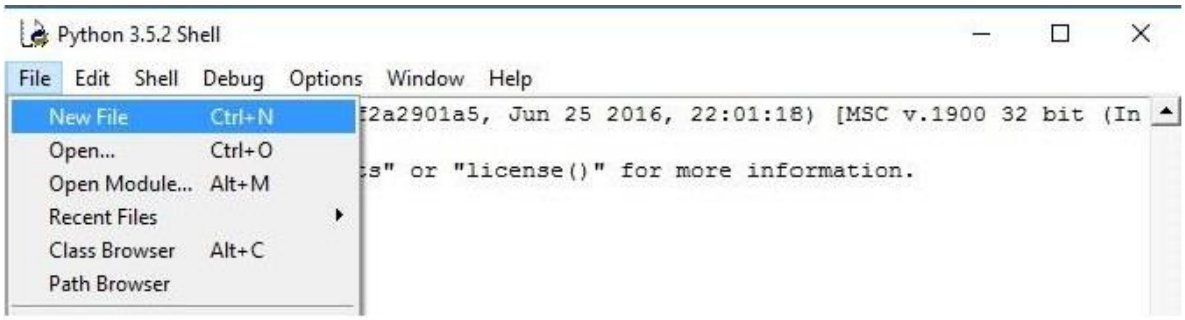
Curso aprendiendo a programar con Python
2016

Trabajo Práctico N°5

Orientación a Objetos

Ejercicio 1 – Clase Mascota

1. El objetivo del programa es crear una instancia de un objeto llamado mascota, que posea como atributos el número de patas y su color.
2. Abrir el programa IDLE (Python 3.5)
3. Observar que aparece el prompt esperando ingreso de datos.
4. Ingresar al menú **File → New File** o **Ctrl+N**



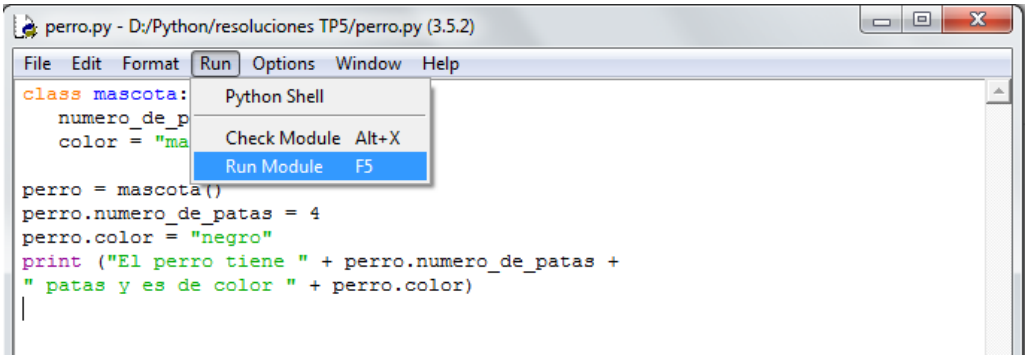
5. Se abrirá una nueva pantalla en blanco

6. Cree un archivo, llamado Mascota.
7. Declare una clase llamada Persona, mediante la siguiente línea:
   ```
   class mascota:
   ```
8. Asígnele los siguientes atributos a esa persona: numero_de_patas, color.
9. Instancie un objeto de tipo mascota llamado perro mediante la siguiente línea:
   ```
   perro=mascota()
   ```
10. Asígnele un número de patas al perro por defecto con la siguiente líena:
    ```
    pero.numero_de_patas=4
    ```

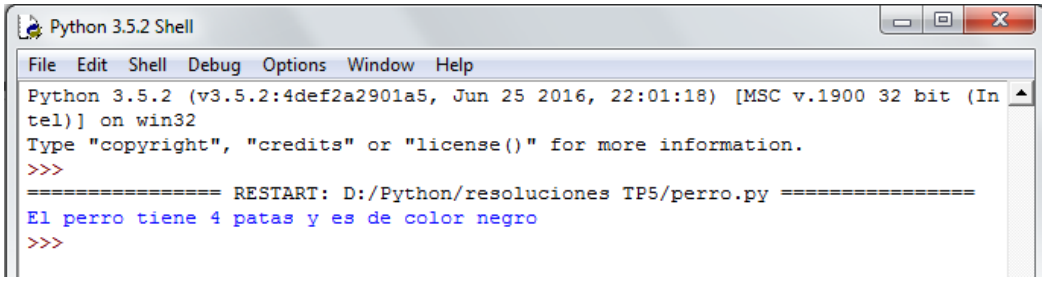
Observe el orden en que ocurrió la línea: primero coloco el nombre de la instancia seguido por un punto, el nombre del atributo que deseo modificar y finalmente el valor que deseo modificar.

11. Realice lo mismo que el punto 10, pero con el color del perro.
12. Muestre las características del objeto creado mediante la siguiente línea:
    ```
    print("El perro tiene “ + str(perro.numero_de_patas) + “ patas y es de color “ + perro.color)
    ```
13. Guarde el código del programa con **File → Save** o **Ctrl+S** con el nombre Perro.py (recuerde que el programa debe tener extensión .py o Tipo “Python Files”, de lo contrario no funcionará).
    No se olvide de seleccionar la carpeta donde desee que se guarde el archivo (**Recomendación:** Modificar la que aparece por defecto).

14. Ejecute el programa con **F5** o con **Run → Run Module**



15. Se abrirá nuevamente la pantalla principal del IDLE (donde figura el prompt >>>).
16. A continuación comenzará a ejecutarse su programa y se visualizará el resultado del mismo de color azul.



Ejercicio 2 – Métodos de clase Mascota

1. El objetivo del programa es crear un método dentro de la clase perro del ejercicio anterior, que muestre que el perro está durmiendo.
2. Utilice el código del ejercicio 1.
3. Agregue dentro de la clase mascota un método llamado dormir que recibe por parámetros un objeto self.
4. Complete el cuerpo de la función dormir para que muestre por pantalla el mensaje “zzz”.
5. Invoque el método dormir del objeto perro mediante la siguiente línea:

```
perro.dormir
```

Ejercicio 3 – Mensaje

1. El objetivo del programa es mostrar por pantalla un mensaje mediante métodos, utilizando un método inicializador.
2. Cree un nuevo archivo llamado Mensaje
3. Defina una nueva clase llamada Mensaje
4. Dentro del cuerpo de la clase declare un método llamado inicializar que reciba 2 parámetros: self y tit
5. Asigne lo recibido por parámetros a la clase mediante la siguiente línea:

```
self.titulo=tit
```

6. Defina otro método llamado mostrar, que reciba como parámetro un objeto self
7. Dentro del método declare una línea que imprima lo contenido en la clase mensaje mediante el siguiente mensaje:

```
print(self.titulo)
```

8. Fuera del cuerpo de la clase instancie un nuevo objeto Mensaje llamado mensaje
9. Indique al objeto mensaje que ejecute la función inicializar y envíe por parámetros el siguiente mensaje: “Estoy aprendiendo orientación a objetos con Python”
10. Indique al objeto mensaje que ejecute el método mostrar.
11. Muestre el resultado en la salida estándar mediante un mensaje.
12. Guarde el archivo.
13. Ejecute

Ejercicio 4 – Hora

1. El objetivo del programa es mostrar por pantalla la hora que el ususario desee.
2. Cree un archivo, llamado Hora.
3. Defina una clase llamada Horas.
4. Dentro de la clase declare una función llamada imprimeHora que reciba como parámetro un objeto de tipo hora.
5. Dentro de la función, complete el cuerpo del mismo para que imprima hora:minutos:seguntos:
6. Dentro del archivo cree la sentencia para leer desde el teclado.
7. Solicite al usuario que ingrese las horas y guárdelo en un atributo del objeto horaActual mediante la siguiente línea:

```
horaActual.horas=input("Ingrese las horas: ")
```

Observe el orden en que se colocan los argumentos: primero el nombre del objeto seguido por un punto, y finalmente el nombre del atributo.

8. Realice lo mismo que en punto 7, pero con los minutos y segundos.
9. Invoque al método imprimeHora declarado dentro de la clase mediante el siguiente mensaje:

```
horaActual.imprimeHora()
```

10. Muestre el resultado en la salida estándar mediante un mensaje.
11. Guarde el archivo.
12. Ejecute.

Ejercicio 5 – Número Complejo

1. El objetivo del programa es mostrar por pantalla un número complejo.

Recuerde que un número complejo está dividido en 2: parte real y parte imaginaria

2. Cree un archivo, llamado Complejo.
3. Defina una clase llamado Complejo.
4. Dentro de la clase declare una función constructora mediante la siguiente línea:

```
def __init__(self, partereal, parteimaginaria):
```

Observe que los parámetros recibidos son 3: el objeto en si mismo, la parte real y la parte imaginaria.

5. Dentro de la función, complete el cuerpo del mismo para que asigne a un atributo llamado r la parte real mediante la siguiente línea:

```
self.r=partereal
```

6. Realice lo mismo que en el punto 5, pero con un atributo llamado i.
7. Concatene a la asignación de la parte imaginaria el caracter i.
8. Dentro del archivo, pero fuera de la clase cree la sentencia para leer desde el teclado.
9. Solicite al usuario que ingrese la parte real y guárdelo en una variable llamada parteReal
10. Realice lo mismo que en punto 9, pero con la parte imaginaria.

11. Instancie un objeto llamado numero pasándole por parámetros al constructor la parte real y la parte imaginaria ingresada previamente.
12. Muestre el resultado en la salida estándar mediante un mensaje mediante la siguiente línea:

```
print(x.r, x.i)
```

13. Guarde el archivo.
14. Ejecute.

Ejercicio 6 – Becario

1. El objetivo del programa es mostrar por pantalla un Becario con su beca asignada.
2. Cree un archivo, llamado Becario.
3. Defina una clase llamado Becario.
4. Dentro de la clase declare una función constructora que asigne el nombre del becario, su legajo y la beca asignada.

Recuerde las becas que actualmente ofrece el Centro de Estudiantes.

5. Dentro de la función, complete el cuerpo del mismo para que asigne a un atributo llamado nombre el nombre del becario:
6. Realice lo mismo que en el punto 5, pero con los otros atributos.
7. Dentro del archivo, pero fuera de la clase cree la sentencia para leer desde el teclado.
8. Solicite al usuario que ingrese el nombre del becario y guárdelo en una variable llamada nombre
9. Realice lo mismo que en punto 9, pero con el legajo y el nombre de la beca.
10. Instancie un objeto llamado becario pasándole por parámetros al constructor los valores ingresados previamente.
11. Muestre el resultado en la salida estándar mediante un mensaje
12. Guarde el archivo.
13. Ejecute.

Ejercicio 6 – Becarios y Becas

1. El objetivo del programa es mostrar por pantalla instancias de Becarios con su beca asignada.
2. Cree un archivo, llamado Becas.
3. Defina una clase llamado Becario.
4. Dentro de la clase declare una función constructora que asigne el nombre del becario, su legajo y la beca asignada.

Recuerde que los métodos constructores utilizan la palabra reservada `self`.
Ejemplo, para inicializar el nombre usamos la siguiente instrucción:

```
self.nombre=nombre
```

5. Dentro del archivo, pero fuera de la clase cree la sentencia para leer desde el teclado.
6. Solicite al usuario que ingrese el nombre del becario y guárdelo en una variable llamada nombre
7. Realice lo mismo que en punto 6, pero con el codigo y el nombre de la beca.
8. Instancie un objeto llamado becario pasándole por parámetros al constructor los valores ingresados previamente.
9. Cree una clase llamada Beca
10. Realice el método constructor de la misma con los atributos código y nombre.
11. Cree una nueva clase llamada Main
12. Solicite al usuario que ingrese la cantidad de becas y becarios que desee.

13. Dentro de dicha clase realice la asociación entre clases mediante el siguiente código:

```
def __call__(self):
    becas = []
    alumnos = []
```

Observe que, al igual que en el Trabajo Práctico Nº3, estamos creando arreglos con la cantidad de becas y alumnos ingresadas por el usuario.

14. Utilice un bucle para recorrer el arreglo de becas
15. Imprima el codigo de la beca seguida de su nombre.
16. Utilice un bucle para recorrer el arreglo de becarios
17. Imprima el legajo del alumno seguido de su nombre.
18. Fuera de las clases declare una instancia del programa main con la siguiente línea:

```
main_instance = Main()
main_instance()
```

19. Muestre el resultado en la salida estándar mediante un mensaje
20. Guarde el archivo.
21. Ejecute.