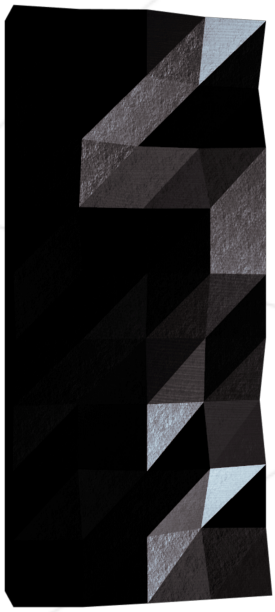

Discovery Piscine

Módulo8 - Python

Resumen: En este módulo, vemos cómo usar métodos y scopes.

Versión: 2.0

Índice general

Capítulo I

Unas palabras sobre esta Discovery Piscine

¡Bienvenido!
Comenzarás un módulo de esta Discovery Piscine en programación informática. Nuestro objetivo es introducirte al código que se esconde detrás del software que usas a diario y sumergirte por completo en el aprendizaje entre pares, el modelo educativo de 42.

Programar trata sobre lógica, no sobre matemáticas. Te brinda bloques de construcción básicos que puedes combinar de innumerables formas. No existe una única solución "correcta" para un problema; tu solución será única, al igual que la de cada uno de tus compañeros.

Rápida o lenta, elegante o desordenada, mientras funcione, ¡eso es lo que importa! Estos bloques de construcción formarán una secuencia de instrucciones (para cálculos, visualizaciones, etc.) que el ordenador ejecutará en el orden que diseñes.

En lugar de ofrecerte un curso donde cada problema tiene una única solución, te colocamos en un entorno de aprendizaje entre pares. Buscarás elementos que te ayuden a afrontar el desafío, los refinarás mediante pruebas y experimentación, y finalmente crearás tu propio programa. Habla con otros, comparte tus perspectivas, genera nuevas ideas en conjunto y prueba todo por ti mismo para asegurarte de que funcione.

La evaluación entre pares es una gran oportunidad para descubrir enfoques alternativos y detectar posibles problemas en tu programa que podrías haber pasado por alto (piensa en lo frustrante que puede ser que un programa se bloquee). Cada revisor analizará tu trabajo de manera diferente, como clientes con expectativas variadas, brindándote nuevas perspectivas. Incluso podrías establecer conexiones para futuras colaboraciones.

Al final de esta Piscine, tu recorrido será único. Habrás enfrentado distintos desafíos, validado diferentes proyectos y elegido caminos distintos a los de los demás, ¡y eso está perfectamente bien! Esta es una experiencia tanto colectiva como individual, y todos sacarán algo valioso de ella.

Buena suerte a todos; esperamos que disfruten este viaje de descubrimiento.

Capítulo II

Introducción

Lo que aprenderás en este módulo:

- Aprenderás a cómo usar métodos y scopes.

Capítulo III

Instrucciones generales

A menos que se indique lo contrario, las siguientes reglas se aplican todos los días de esta Piscine.

- Este documento es la única fuente confiable. No te fíes de rumores.
- Este documento puede actualizarse hasta una hora antes del plazo de entrega.
- Las tareas deben completarse en el orden especificado. No se evaluarán tareas posteriores si las anteriores no están correctamente finalizadas.
- Presta mucha atención a los permisos de acceso de tus archivos y carpetas.
- Tus tareas serán evaluadas por tus compañeros de la Piscine.
- Todos los ejercicios de terminal deben ejecutarse con `/bin/bash`.
- No debes dejar ningún archivo en tu espacio de entrega salvo aquellos explícitamente solicitados en las instrucciones.
- ¿Tienes una duda? Pregunta a tu compañero de la izquierda. Si no, prueba con el de la derecha.
- Toda respuesta técnica que necesites se encuentra en las páginas `man` o en línea.
- Recuerda usar el foro de la Piscine en tu intranet y Slack!
- Lee los ejemplos detenidamente, pueden contener requisitos que no sean evidentes en la descripción del ejercicio.
- ¡Por Thor, por Odín! ¡Usa tu cerebro!

Capítulo IV

Ejercicio 00 Descubriendo Métodos!

	Ejercicio: 00
Descubriendo Métodos!	
Directorio de entrega: *ex*00/	
Archivos a entregar: `hello_all.py`	
Funciones autorizadas: `All`	

- Crea un programa llamado `hello_all.py`.
- Asegúrate de que este programa sea ejecutable.
- Este programa debe:
  - Definir un método llamado `hello` que imprima "Hello, everyone!".
  - Llamar a este método para mostrar el mensaje. Consulta el ejemplo a continuación, excepto que la definición del método ha sido oculta.

```
?> cat hello_all.py
# Your method definition

hello()
?> ./hello_all.py
Hello, everyone!
?>
```

`Busca "method definition in Python".`

Capítulo V

Ejercicio 01: upcase_it

	Ejercicio: 01
¡El Regreso de upcase!	
Directorio de entrega: $ex01/$	
Archivos a entregar: `upcase_it.py`	
Funciones autorizadas: `All`	

- Crea un programa llamado `upcase_it.py` (¡de nuevo!)
- Asegúrate de que este programa sea ejecutable.
- Define un método en el programa llamado `upcase_it`.
- El método `upcase_it` debe recibir una string como argumento y devolver la string en mayúsculas.
- Prueba el método llamándolo en tu programa. En el ejemplo a continuación, lo probamos con "hello":

```
?> cat upcase_it.py
# Your method definition

print(upcase_it("hello"))
?> ./upcase_it.py
HELLO
?>
```

Capítulo VI

Ejercicio 02: downcase_all

	Ejercicio: 02
Demos un Paseo por el Array	
Directorio de entrega: $ex02/$	
Archivos a entregar: `downcase_all.py`	
Funciones autorizadas: `All`	

- Crea un programa llamado `downcase_all.py`.
- Asegúrate de que este programa sea ejecutable.
- Define un método en este programa llamado `downcase_it`.
- El método `downcase_it` debe recibir una string como argumento y devolver la string en minúsculas.
- Aplica este método a cada uno de los parámetros del programa y muestra el valor de retorno para cada uno.
- Si no hay parámetros, muestra `"none"` seguido de un salto de línea.

```
?> ./downcase_all.py
none
?> ./downcase_all.py "HELLO WORLD" "I understood Arrays well!"
hello world
i understood arrays well!
?>
```

Capítulo VII

Ejercicio 03: greetings_for_all

	Ejercicio: 03
¡Vamos a Saludar a Todos!	
Directorio de entrega: $ex03/$	
Archivos a entregar: `greetings_for_all.py`	
Funciones autorizadas: `All`	

- Crea un programa llamado `greetings_for_all.py` que no reciba parámetros.
- Asegúrate de que este programa sea ejecutable.
- Crea un método llamado `greetings` que reciba un nombre como parámetro y muestre un mensaje de bienvenida con ese nombre.
- Si el método es llamado sin un argumento, su parámetro por defecto debe ser "noble stranger".
- Si el método es llamado con un argumento que no sea una string, debe mostrar un mensaje de error en lugar del mensaje de bienvenida.

```
?> cat greetings_for_all.py | cat -e
# your method definition here

greetings('Alexandra')
greetings('Wil')
greetings()
greetings(42)
```

will produce the following output:

```
?> ./greetings_for_all.py | cat -e
Hello, Alexandra.$
Hello, Wil.$
Hello, noble stranger.$
Error! It was not a name.$
?>
```

`Busca default parameter, método isinstance.`

Capítulo VIII

Ejercicio 04: methods_everywhere

	Ejercicio: 04
¡Métodos en Todas Partes!	
Directorio de entrega: $ex04/$	
Archivos a entregar: `methods_everywhere.py`	
Funciones autorizadas: `All`	

- Crea un programa llamado `methods_everywhere.py` que reciba parámetros.
- Asegúrate de que este programa sea ejecutable.
- Debes crear dos métodos diferentes en este programa:
  - El método `shrink`: Debe recibir una string como parámetro y mostrar los primeros ocho caracteres de esa string.
  - El método `enlarge`: Debe recibir una string como parámetro y añadir caracteres 'Z' hasta que la string tenga un total de ocho caracteres. Luego, debe mostrar la string resultante.
- Para cada argumento pasado al programa:
  - Si el argumento tiene más de ocho caracteres, llama al método `shrink` sobre él.
  - Si el argumento tiene menos de ocho caracteres, llama al método `enlarge` sobre él.
  - Si el argumento tiene exactamente ocho caracteres, muéstralo directamente seguido de un salto de línea.

```
?> ./methods_everywhere.py | cat -e
none$
?> ./methods_everywhere.py 'lol' 'physically' 'backpack' | cat -e
lolZZZZZ$
physical$
backpack$
?>
```

método shrink: Usa slicing.

método enlarge: Similar a los arrays, puedes añadir caracteres a una string usando el operador de concatenación.

Capítulo IX

Ejercicio 05: scope_that

	Ejercicio: 05
¡No puedes tocar esto!	
Directorio de entrega: $ex05/$	
Archivos a entregar: `scope_that.py`	
Funciones autorizadas: `All`	

- Crea un programa llamado `scope_that.py` que no reciba parámetros.
- Asegúrate de que este programa sea ejecutable.
- Dentro del programa, crea un método llamado `add_one` que reciba un parámetro y le agregue 1 al parámetro.
- Inicializa una variable en el cuerpo del programa, muéstrala y luego llama al método que agrega 1.
- Muestra tu variable nuevamente en el cuerpo del programa.
- ¿Qué observas?

Capítulo X

Entrega y Evaluación entre Pares

- Debes tener una carpeta llamada `discovery_piscine` en la raíz de tu directorio personal.
- Dentro de `discovery_piscine`, debe existir una carpeta llamada `module8`.
- Dentro de `module8`, debe haber una carpeta para cada ejercicio.
- El Ejercicio 00 debe estar en la carpeta `ex00`, el Ejercicio 01 en `ex01`, y así sucesivamente.
- Cada carpeta de ejercicio debe contener los archivos solicitados en el enunciado.

`Durante tu defensa, todo lo que no esté en la carpeta correspondiente para el día no será revisado.`