

**Programación II (Optativa)**

**Código de materia 25**

**Prof. Titular: Ing. Gastón Matías Weingand**

**Facultad de tecnología informática UNIVERSIDAD ABIERTA INTERAMERICANA**



**Guía de Revisión**

**Introducción a Python**

1. *¿Qué diferencias existen entre un programa y un proyecto?*

El proyecto es un conjunto de elementos que suele estar compuesto por uno o varios programas. El programa es mas, un pequeño desarrollo que no suele contar con Documentación tal como tiene el proyecto, es decir, diagramas de arquitectura de la solución, casos de uso, etc.

1. *¿Qué es el entorno integrado de desarrollo?*

El entorno integrado de desarrollo , o también conocido como IDE, son un conjunto de herramientas en donde escribimos y ejecutamos nuestro código. Gracias a ellas, la tarea de programar se hace muy amena.

1. *Enumere y explique las diferencias entre la programación estructurada, orientada a objetos y funcional.*

* La programación estructurada es llamada así porque sigue una estructura al nivel de la ejecución (runtime) del programa que estemos desarrollando.
* La programación orientada a objetos trabaja con el concepto de clases, objetos instancias, etc, que permiten declarar tipos y llamarlos desde cualquier parte del programa, sin necesidad de seguir un orden claro al momento de codear.
* La programación funcional, como su nombre lo dice esta orientada a funciones, que reciben parámetros y devuelven un resultado.

1. *¿Qué es una abstracción? ¿Qué es un tipo?*

Una abstracción es cuando eliminamos todos los conceptos de algo que no son necesarios ni componen al elemento, es decir, no son sus fundamentales (solo nos quedamos con estos últimos).

1. Defina clases, objetos, herencia y polimorfismo
2. *¿Qué es python? ¿Por qué se dice que es semi interpretado?*

Python es un lenguaje de programación que fue ganando popularidad en los últimos años. Se dice que es semi-interpretado dado que si bien su ejecución se realiza como si fuera interpretado, a su vez tiene una especie de compilación para pasarse al lenguaje de maquina para poder correr.

1. ¿Por qué es importante conocer python? ¿Es adecuado para cualquier tipo de aplicaciones?
2. ¿Cómo es la gestión de tipos de python? ¿Fuerte o débil? ¿Estático o Dinámico?

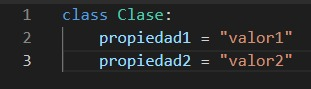
**Clases y Objetos**

* 1. *¿Cuáles son las estructuras condicionales y de repetición que ofrece el lenguaje?*

Las estructura que ofrece el lenguaje son las siguientes:

* For
* IF
* Else / Else IF
* While
  1. *¿Cómo se define una clase?*

Para definir una clase solo basta con escribir la palabra “class” delante de la clase que se quiere definir. Ejemplo:



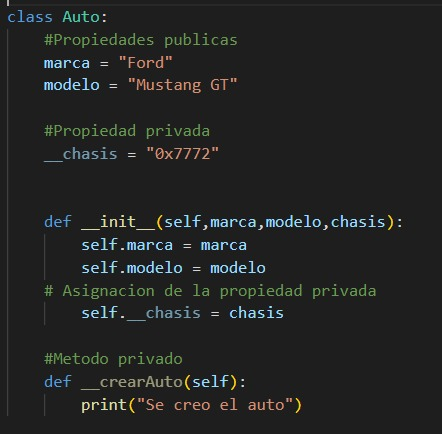
* 1. ¿Qué diferencia a una clase clásica con una de “nuevo estilo”?
  2. *¿Cómo se instancia una clase para crear un objeto?*

Con respecto a esto python da una solución muy simple a la instanciacion de objetos comparado con otros lenguajes OO. Con solo crear un nuevo objeto del tipo de la clase, la instancia se realiza automáticamente. Ejemplo:



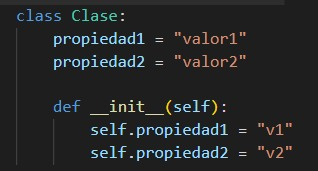
* 1. *¿Cómo se gestiona el encapsulamiento?*

Para gestionar el encapsulamiento tenemos que utilizar las herramientas del lenguaje. En el caso de las propiedades publicas no tenemos que realizar nada en especial, dado que al por ejemplo definir una clase o una propiedad, si no marcamos ningún tipo de “Scope” el programa automáticamente interpreta que estamos hablando de algo publico. Ahora si queremos que esto pase a ser “Privado” vamos a tener que agregar un modificador, el mismo es “\_\_”. Ejemplo en código:

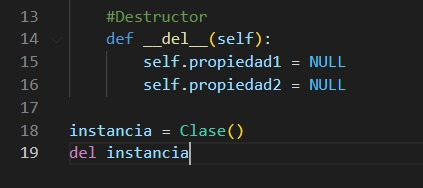


* 1. *¿Cómo y para qué se utilizan el constructor y el destructor?*

El constructor se utiliza para poder darle valores de inicio a una instancia de una clase. Por ejemplo si quiero que todos los autos que se creen sean de color naranja, en el constructor voy a definir por “defecto” el parámetro “color = naranja”.



Con respecto al destructor, se utiliza para limpiar o mejor dicho borrar una instancia de una clase, para que deje de estar en memoria y así no ocupar la misma. No solo se usa con este, fin, pero en aplicaciones con grandes cantidades de objetos es una buena aplicación.



* 1. ¿Para qué se utiliza el método \_\_init\_\_(self, args)?
  2. ¿Cómo se implementa una herencia simple y múltiple?
  3. ¿Qué métodos especiales son los más utilizados?
  4. ¿Cómo se maneja el polimorfismo?

**Programación funcional y excepciones**

* 1. ¿Qué es una función de orden superior?
  2. ¿Qué es una iteración de orden superior?
  3. ¿Para qué se utilizan las funciones map, filter y reduce?
  4. ¿Para que sirve el operador lambda?
  5. ¿Qué son las list comprehensions?
  6. ¿Qué son las expresiones generadoras?
  7. ¿Qué es un decorador?
  8. ¿Cómo se gestionan las excepciones?



**Archivos y expresiones regulares**

* 1. Diferencias entre módulo y paquete.
  2. Entrada y salida estándard. Parámetros de línea de comandos. Ejemplos.
  3. ¿Cómo se gestiona la escritura y lectura de archivos?
  4. Movimiento del puntero de escritura. Ejemplo.
  5. ¿Cómo se tratan los errores en la gestión de archivos?
  6. Uso de expresiones regulares. ¿Qué es el módulo re? Ejemplos.
  7. ¿Qué flags son los más utilizados?
  8. ¿Qué es un RegexObject?

**Librerías e introducción al machine learning**

* 1. ¿Para qué sirven las librerías? Pygame, RE, Collections, NumPy, SQLAlchemy, Request y Pillow
  2. ¿Qué es el machine learning?
  3. ¿Qué es el deep learning?
  4. Defina los conceptos de: Aprendizaje. Perceptrón. Función de activación. Red neuronal. Back propagation.
  5. ¿Para que se utilizan los framework Anaconda y miniconda? ¿Diferencia entre ambos?



**Guía Trabajos Prácticos**

**Introducción a Python**

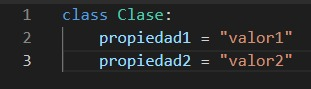
1. Realice un cuadro sinóptico donde se expliquen las ventajas y desventajas en PE,

POO y Funcional.

1. Realice un cuadro sinóptico con las definiciones de clase, objeto, polimorfismo y herencia

**Clases y Objetos**

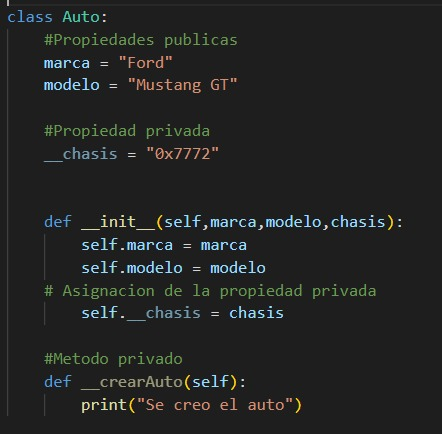
1. Escriba en un mismo código de ejemplo las sentencias condicionales y de iteración para cada estructura conocida del lenguaje: [if, elseif y else] [while y for].
2. *Realice el código necesario para definir una clase*

**

1. *Realice el código necesario para instanciar dicha clase*

**

1. *Realice un ejemplo en código donde se demuestre el encapsulamiento*



1. Realice un ejemplo en código para herencia simple y múltiple
2. Realice un ejemplo en código que permita polimorfismo
3. Realice un ejemplo en código donde se utilicen los métodos especiales init, new, del, str y cmp.

**Programación funcional y excepciones**

1. Realice un ejemplo en código para el uso de diccionarios, cadenas y listas.
2. Realice un ejemplo en código para el uso de una función de orden superior
3. Realice un ejemplo en código para cada función de: map, filter y reduce
4. Realice un ejemplo en código para el uso de una función lambda
5. Realice un ejemplo en código para el uso de una list comprehension
6. Realice un ejemplo en código para el uso de una expresión generadora
7. Realice un ejemplo en código para el uso de una función decoradora
8. Realice un ejemplo en código para el tratamiento de una excepción



**Archivos y expresiones regulares**

1. Genere un código que permita la creación y el uso de un módulo.
2. Agregue el uso de paquetes al código anterior.
3. Genere un código para la creación, apertura, grabación, lectura y cierre de un archivo
   1. de texto
   2. de datos binarios
4. Genere el código para tratar los errores de la gestión de archivos del punto anterior.
5. Genere un ejemplo donde aplique expresiones regulares en la lectura (Línea a línea) de un archivo X (Por ejemplo, encontrar direcciones de email u otro patrón que usted desee), guarde las líneas que contienen su patrón en un archivo y en otro las líneas que no lo contiene.

**Librerías e introducción al machine learning**

1. Genere códigos de ejemplo utilizando cada una de las bibliotecas: Pygame, RE, Collections, NumPy, SQLAlchemy, Request y Pillow.
2. Genere el código para que un perceptrón pueda aprender las compuertas AND, OR, NOT, XOR y XNOR.
3. Diseñe el código para que un vehículo arduino/micro:bit pueda aprender a evitar obstáculos



**Guía de Abordaje**

**INTRODUCCIÓN A PYTHON**

Raúl González Duque - “Python para todos”. 1ª Ed. España. Creative Commons 2015. Capítulos 1 y 2.

Eric Matthes - “Python Crash Course” 2ª Ed. USA. No Starch Press 2019. Introducción y Capítulo 1

**CLASES Y OBJETOS**

Raúl González Duque - “Python para todos”. 1ª Ed. España. Creative Commons 2015. Capítulos 7 y 8

David Beazley, Brian K. Jones - “Python Cookbook” 3 ª Ed. USA. OReilly Media 2013. Capítulo 8.  
Eric Matthes - “Python Crash Course” 2ª Ed. USA. No Starch Press 2019. Introducción y Capítulo 9

**PROGRAMACIÓN FUNCIONAL Y EXCEPCIONES**

Raúl González Duque - “Python para todos”. 1ª Ed. España. Creative Commons 2015. Capítulos 9 y 10

David Beazley, Brian K. Jones - “Python Cookbook” 3 ª Ed. USA. OReilly Media 2013. Capítulo 10 y 14.

Eric Matthes - “Python Crash Course” 2ª Ed. USA. No Starch Press 2019. Introducción y Capítulo 8 y 10

**ARCHIVOS Y EXPRESIONES REGULARES**

Raúl González Duque - “Python para todos”. 1ª Ed. España. Creative Commons 2015. Capítulos 11, 12 y 13.  
David Beazley, Brian K. Jones - “Python Cookbook” 3 ª Ed. USA. OReilly Media 2013. Capítulo 2, 5 y 10.

Eric Matthes - “Python Crash Course” 2ª Ed. USA. No Starch Press 2019. Introducción y Capítulo 10

**LIBRERÍAS E INTRODUCCIÓN AL MACHINE LEARNING**

Aurélien Géron – “Hands-On Machine Learning with Scikit-Learn & TensorFlow”. 1ª Ed USA. OReilly Media 2017. Capítulo 10