

Ciclo 1: Fundamentos de programación con Python

Contenido semana 2



1. Introducción al pensamiento computacional
2. Principios de pensamiento computacional
3. Introducción al lenguaje de programación Python
4. Realización de operaciones aritméticas en programación
5. Metodología para la solución de algoritmos
6. Guía de estilos para código Python
7. Implementación de estructuras de control de secuencia (if-elif)
8. Implementación de estructuras de control de repetición (while)
9. Implementación de estructuras de control de repetición (for)

Ciclo 1: Fundamentos de programación con Python

Introducción al pensamiento computacional



Ciclo 1: Fundamentos de programación con Python

Introducción al pensamiento computacional



- El pensamiento computacional implica resolver problemas, diseñar sistemas y comprender el comportamiento humano, haciendo uso de los conceptos fundamentales de la informática.
- El pensamiento computacional es el proceso que permite formular problemas de forma que sus soluciones pueden ser representadas como secuencias de instrucciones y algoritmos.
- El pensamiento computacional es el proceso de reconocimiento de aspectos de la informática en el mundo que nos rodea, y aplicar herramientas y técnicas de la informática para comprender y razonar sobre los sistemas y procesos tanto naturales como artificiales.

Ciclo 1: Fundamentos de programación con Python

Introducción al pensamiento computacional



- El pensamiento computacional es un proceso de resolución de problemas que incluye las siguientes características:
 - Formular problemas de forma que se permita el uso de un ordenador y otras herramientas para ayudar a resolverlos.
 - Organizar y analizar lógicamente la información.
 - Representar la información a través de abstracciones como los modelos y las simulaciones.
 - Automatizar soluciones haciendo uso del pensamiento algorítmico (estableciendo una serie de pasos ordenados para llegar a la solución).
 - Identificar, analizar e implementar posibles soluciones con el objetivo de lograr la combinación más efectiva y eficiente de pasos y recursos.

Ciclo 1: Fundamentos de programación con Python

Introducción al pensamiento computacional



- El pensamiento computacional es un proceso de resolución de problemas que incluye las siguientes características (cont.):
 - Generalizar y transferir este proceso de resolución de problemas para ser capaz de resolver una gran variedad de familias de problemas.

Ciclo 1: Fundamentos de programación con Python

Introducción al pensamiento computacional



- Introducción al pensamiento computacional
 - El pensamiento computacional consiste en resolver problemas cotidianos mediante el uso de los conceptos fundamentales de la programación informática cuyas soluciones pueden ser representadas mediante una serie de pasos o instrucciones.

En otras palabras, se trata del proceso mental a través del cual una persona se plantea un problema y para su posible solución utiliza una secuencias de instrucciones ejecutadas por una computadora, un humano o ambos. Es decir, aplica habilidades propias de la computación y del pensamiento crítico.

Ciclo 1: Fundamentos de programación con Python

Introducción al pensamiento computacional



- Pilares del pensamiento computacional
 - Pensamiento lógico
 - Sistema que se utiliza para distinguir entre argumentos correctos e incorrectos
 - Argumento: Cadena de razonamiento que termina en una conclusión

Deductivo	Inductivo
<ol style="list-style-type: none">1. Sócrates es un hombre.2. Todos los hombres son mortales.3. Luego Sócrates es un mortal	<ol style="list-style-type: none">1. Una bolsa contiene 99 pelotas rojas y una pelota negra.2. Hay 100 personas y cada una saca una pelota de la bolsa.3. Sara es una de esas 100 personas4. Luego Sara probablemente sacará una pelota roja.

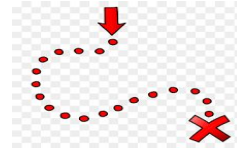
Ciclo 1: Fundamentos de programación con Python

Introducción al pensamiento computacional



- Pilares del pensamiento computacional
 - Pensamiento lógico.

1. Lassie es un perro.
2. Todos los perros son cafés.
3. Luego Lassie es de color café.



1. El razonamiento es válido.
2. Le da a la máquina información confiable.
3. ¿Es posible interpretar que conclusión reporta la computadora, es decir, el resultado es incuestionablemente verdadero (el razonamiento fué deductivo) o probablemente el razonamiento fué inductivo ?.

Ciclo 1: Fundamentos de programación con Python

Introducción al pensamiento computacional



- Pilares del pensamiento computacional
 - Pensamiento algorítmico.
 - Se basa en pensamiento lógico
 - Forma de expresar una tarea de varios pasos para explicar a un tercero (ya sea humano o máquina como realizarla con extrema precisión.
 - Propiedades:
 - Colección de pasos
 - Asertividad
 - Secuencialidad (continuidad)



Ciclo 1: Fundamentos de programación con Python

Introducción al pensamiento computacional



Conceptos fundamentales

- Descomposición: Divide y vencerás
- Generalización y reconocimiento de patrones: Problemas similares, soluciones similares.
- Abstracción: Lo que no sirve, que no estorbe.
- Modelado: Representación del problema a resolver.
- Evaluación: Mejor tarde que nunca.

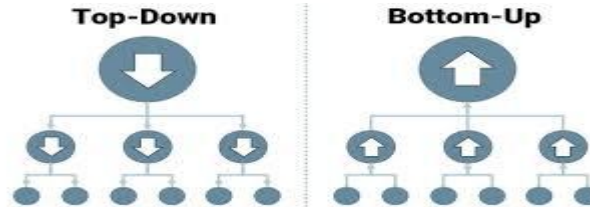


Ciclo 1: Fundamentos de programación con Python

Introducción al pensamiento computacional



Estrategias de aproximación



General a lo particular (Top-Down)	Particular a lo general (Bottom-Up)
<ul style="list-style-type: none">● Ciencia de la computación● Resumen del sistema sin detalles (caja negra).● Cada parte de la “caja negra” se redefine con mayor nivel de detalle, generando nuevas cajas negras.● Proceso sucesivo hasta definir todo el sistema detalladamente con cajas negras.	<ul style="list-style-type: none">● Desde el inicio se detalla cada parte del sistema.● Al haber completado las especificaciones de cada parte se vincula a un nivel superior.● Proceso sucesivo hasta definir todo el sistema como una sola gran unidad.

Ciclo 1: Fundamentos de programación con Python

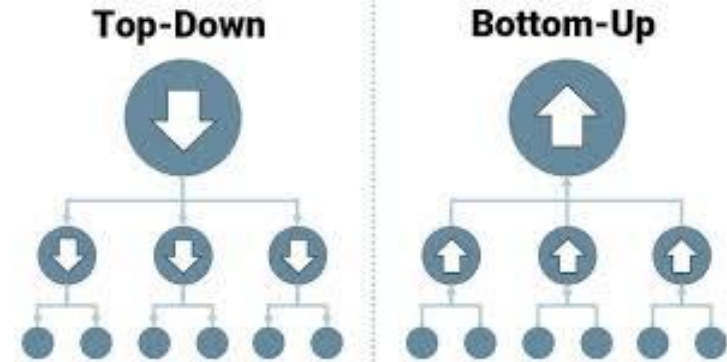
Introducción al pensamiento computacional



Estrategias de aproximación

Top-Down vs Bottom-Up

- Top-Down enfatiza en la planificación y conocimiento completo del proceso, minimizando errores o trabajo extra. Retrasa el inicio de la implementación.
- Bottom-Up permite el inicio temprano de la implementación pero es susceptible a errores por omisión de información



Ciclo 1: Fundamentos de programación con Python

Introducción al pensamiento computacional



Estrategias para el desarrollo del pensamiento computacional

- **Fillomino (<https://www.kakuro-online.com/fillomino/>)**
Es ideal para desarrollar el pensamiento lógico. Es un juego de rompecabezas de tamaño libre formado por bloques de cuadros que delimitan sus diferentes áreas con líneas más oscuras.
- **Sudoku (<https://www.sudoku-online.org/>)**
Este clásico juego matemático, es otro excelente juego para desarrollar el pensamiento lógico. Su objetivo es llenar la cuadrícula 9×9 con los números del 1 al 9 sin que estos se repitan ni horizontal ni verticalmente.
- **Scratch (<https://scratch.mit.edu/projects/editor/?tutorial=home>)**
Scratch es un lenguaje de programación el cual permite que cualquier persona se pueda iniciar en el mundo de la programación.

Ciclo 1: Fundamentos de programación con Python

Introducción al pensamiento computacional



Estrategias para el desarrollo del pensamiento computacional

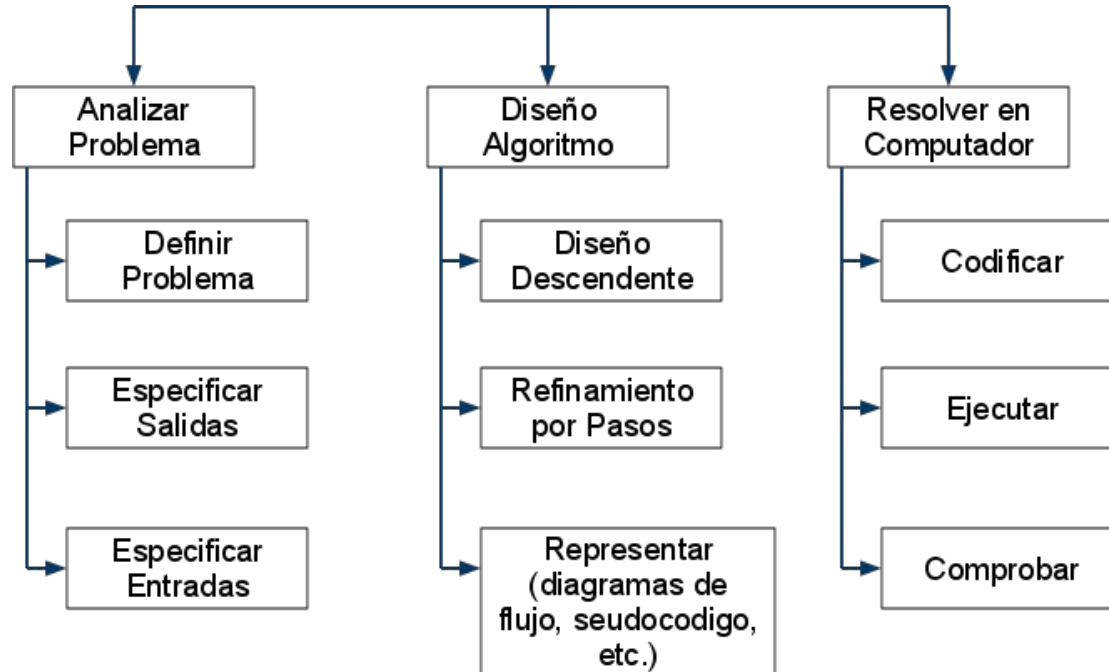
- **Microsoft MakeCode (<https://www.microsoft.com/es-es/makecode>)**
Microsoft MakeCode es un software gratuito de código abierto que ayuda a progresar hacia la programación real a través de la creación de experiencias atractivas de aprendizaje de la informática.

Ciclo 1: Fundamentos de programación con Python

Introducción al pensamiento computacional



Estrategias para el resolución de problemas

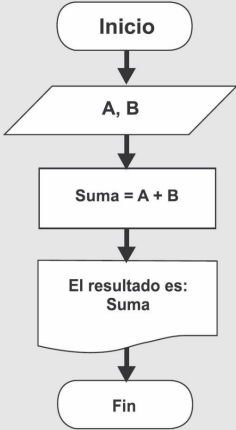


Ciclo 1: Fundamentos de programación con Python

Introducción al pensamiento computacional



Estrategias para la resolución de problemas

Definición del problema: Elaborar un algoritmo para calcular la suma de dos números y representar el algoritmo gráficamente.		
Análisis del problema	Algoritmo	Diagrama de flujo
<p>Entrada:</p> <p><i>A y B representan los dos números.</i></p> <p>Proceso:</p> <p><i>Suma = A + B</i></p> <p>Salida:</p> <p><i>Resultado es Suma</i></p>	<ol style="list-style-type: none">1.- INICIO2.- LEER A y B3.- SUMA = A + B4.- IMPRIMIR ("EL RESULTADO ES: SUMA")5.- FIN	 <pre>graph TD; Inicio([Inicio]) --> Input[/A, B/]; Input --> Process[Suma = A + B]; Process --> Output[El resultado es: Suma]; Output --> Fin([Fin]);</pre>

Ciclo 1: Fundamentos de programación con Python

Introducción al pensamiento computacional



Cómo plantear un problema

Ver documento anexo (comoPlantearResolverProblema.pdf).

Ciclo 1: Fundamentos de programación con Python

Introducción al pensamiento computacional



Ejercicio - Población de hormigas

En la isla del Edén hay un investigador que está analizando la población de hormigas, las cuales se reproducen a una tasa del 40% mensual; en la isla existe además un oso hormiguero que se come 7000 hormigas al final de cada mes (o todas las hormigas que hay si la población de hormigas en ese momento es inferior a esa cifra). Cuando la población de hormigas es superior a 28000, comienza a haber

problemas de alimentación lo que hace que se reduzca la tasa de crecimiento al 31.5% mensual.

Partiendo de un número de hormigas en un momento dado, el investigador quiere calcular la población de hormigas que existirá en la isla al cabo de un número dado de meses de estudio.



Ciclo 1: Fundamentos de programación con Python

Introducción al pensamiento computacional



Tarea - Población de hormigas

- ¿ Cuántas hormigas habrá en la isla del Edén después de cinco (5) meses, si la población inicial es de 25,000 hormigas ?
- Escriba paso a paso los cálculos y condiciones que tuvo en cuenta al desarrollar el ejercicio.



Ciclo 1: Fundamentos de programación con Python

Principios de pensamiento computacional



Ciclo 1: Fundamentos de programación con Python

Principios de pensamiento computacional



El pensamiento computacional consiste en resolver problemas cotidianos mediante el uso de los conceptos fundamentales de la programación informática cuyas soluciones pueden ser representadas mediante una serie de pasos o instrucciones.

En otras palabras, se trata del proceso mental a través del cual una persona se plantea un problema y para su posible solución utiliza una secuencias de instrucciones ejecutadas por una computadora, un humano o ambos. Es decir, aplica habilidades propias de la computación y del pensamiento crítico.

Ciclo 1: Fundamentos de programación con Python

Principios de pensamiento computacional



¿Cuáles son los pilares del pensamiento computacional?

El pensamiento computacional cuenta con cuatro principios, los cuales son:

1. Descomposición de un problema en fases más pequeñas: Consiste en la ruptura de un sistema o problema complejo en partes más pequeñas para que así sean más fáciles de solucionar.

Cada pequeño problema se irá resolviendo uno tras otro hasta solucionar el sistema completo.

2. Reconocimiento de patrones repetitivos: Una vez que hayas descompuesto el problema complejo en varios más pequeños, busca estándares de características comunes.

Encontrar estas semejanzas en los pequeños problemas descompuestos te ayudará a resolver el sistema de forma más eficiente.

Ciclo 1: Fundamentos de programación con Python

Principios de pensamiento computacional



¿Cuáles son los pilares del pensamiento computacional?

3. Abstracción de información irrelevante al problema propuesto: La abstracción se refiere a centrarse en la información importante, dejando de lado aquellas características irrelevantes e innecesarias.

Pero, ¿cuál es información importante? En la abstracción se trata principalmente de las características generales que son comunes a cada elemento, en lugar de detalles específicos.

Luego de contar con estas características generales, se debe proceder a crear un «modelo» del problema, el cual es la idea general del problema que se intenta resolver.

4. Algoritmos escritos presentados para la resolución del problema: Luego de dividir el gran problema en varios más pequeños, identificar las similitudes entre estos, centrarse en los detalles pertinentes y dejar atrás cualquier información irrelevante.

Ciclo 1: Fundamentos de programación con Python

Principios de pensamiento computacional



¿Cuáles son los pilares del pensamiento computacional?

Entonces, ha llegado el momento de desarrollar instrucciones paso a paso o plantear las reglas a seguir para resolver cada uno de estos problemas a través de la programación de una computadora, es decir, crear los algoritmos.

Estos algoritmos pueden ser creados a través de diagramas de flujos o usando pseudocódigos.

Sin embargo, ten en cuenta que un algoritmo es un plan, un conjunto de instrucciones paso a paso para resolver un problema y no siempre implica hazañas complicadas de la programación, sino que puede utilizarse en sistemas complejos externos.

Ciclo 1: Fundamentos de programación con Python

Principios de pensamiento computacional



Uso de diagramas de flujo

Diagramas de flujo ofrecen una manera perfecta para representar algoritmos.

"Un diagrama de flujo es un tipo de diagrama que representa un algoritmo, flujo de trabajo o proceso, mostrando los pasos como cajas de diversos tipos y su orden conectándolos con flechas. Esta representación esquemática ilustra un modelo de solución para un problema dado."– Wikipedia

Diagramas de flujo son una manera fácil de trazar algoritmos, especialmente si necesitan resultados diferentes en el camino de salida. Utiliza convenciones de estilo estándar. Diagramas de flujo de flujo de arriba a abajo y de izquierda a derecha.

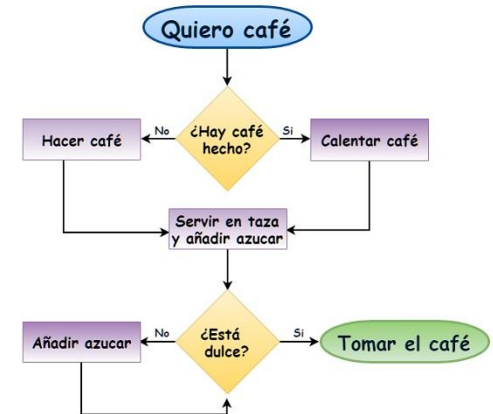
Ciclo 1: Fundamentos de programación con Python

Principios de pensamiento computacional



Uso de diagramas de flujo

Un Diagrama de Flujo representa la esquematización gráfica de un algoritmo, el cual muestra gráficamente los pasos o procesos a seguir para alcanzar la solución de un problema. Su correcta construcción es sumamente importante porque, a partir del mismo se escribe un programa en algún lenguaje de programación.



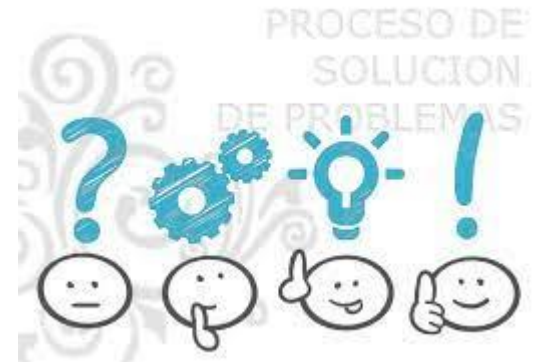
Ciclo 1: Fundamentos de programación con Python

Principios de pensamiento computacional



Solución de problemas

- Definición de problema
- Método para la solución de problemas



Ciclo 1: Fundamentos de programación con Python

Principios de pensamiento computacional



Solución de problemas

- ¿Qué es un problema ?
 - Situación
 - Tarea / interrogante sin inmediata solución
 - Cuestión / proposición dudosa
 - Averiguar/indagar por resultados bajo ciertos datos



Ciclo 1: Fundamentos de programación con Python

Principios de pensamiento computacional



Problemas y estrategias

- Para lograr un objetivo/meta
- Lineamientos
- Para resolver problemas deben desarrollarse estrategias



Ciclo 1: Fundamentos de programación con Python

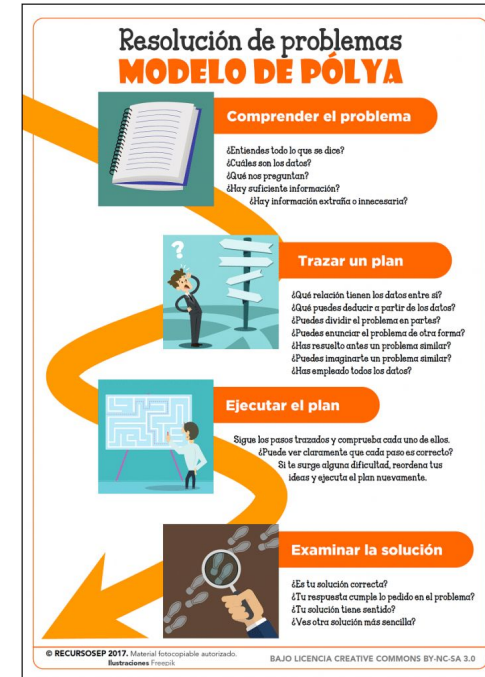
Principios de pensamiento computacional



Método para resolver problemas



<http://mates.aomatos.com/metodo-de-polya-para-resolver-problemas/>



Ciclo 1: Fundamentos de programación con Python

Principios de pensamiento computacional



Método para resolver problemas

1

Entender el problema

- ¿Entiendo todo lo que dice el problema?,
- ¿Puedo replantear el problema con mis propias palabras?,
- ¿Cuáles son los datos que hacen parte del problema?
- ¿Sé a dónde quiere llegar?
- ¿Hay suficiente información?,
- ¿Hay información que no es clara?,
- ¿Es similar a algún otro que ya haya resuelto antes?

2

Configurar un plan

- ¿Puedes usar alguna de las siguientes estrategias
- Ensayo y Error (Conjeturar y probar la conjetura).
- Buscar un Patrón
- Hacer una lista.
- Resolver un problema similar más simple.
- Hacer una figura.
- Hacer un diagrama
- ...

3

Ejecutar el plan

- Aplica la estrategia que escogiste hasta solucionar completamente el problema o hasta que la misma acción te sugiera tomar un nuevo curso.
- Concédete un tiempo razonable para resolver el problema. Si no tienes éxito solicita una sugerencia o deja el problema un rato.
- No tengas miedo de volver a empezar. Suele suceder que un comienzo fresco o una nueva estrategia conducen al éxito.

4

Examinar la solución

- ¿Puedes verificar el resultado? ¿Puedes el razonamiento?
- ¿Puedes obtener el resultado en forma diferente? ¿más sencillo?
- ¿Puedes verlo de golpe?
- ¿Puedes emplear el resultado o el método en algún otro problema?



Ciclo 1: Fundamentos de programación con Python

Principios de pensamiento computacional



Ejercicio 1.

Tres ladrones roban una jarra que contiene 24 onzas de un valioso bálsamo. En su huida se toparon con un vendedor de recipientes, al que le compraron tres jarras vacías.

Cuando llegaron a su guarida decidieron repartir el bálsamo, pero descubrieron que en las jarras solo cabían 5, 11 y 13 onzas, respectivamente.



¿Cómo pudo separarse en tres partes iguales el valioso líquido usando las cuatro jarras disponibles?

Ciclo 1: Fundamentos de programación con Python

Principios de pensamiento computacional





Ejercicio 1.




Inicio: Jarras: 24/24 → 0/5 ; 0/11 ; 0/13

1)  8/24 → 5/5 ; 11/11 ; 0/13

2)  8/24 → 5/5 ; 0/11 ; 11/13

3)  0/24 → 5/5 ; 8/11 ; 11/13

4)  16/24 → 0/5 ; 8/11 ; 0/13

5)  3/24 → 0/5 ; 8/11 ; 13/13

6)  3/24 → 5/5 ; 8/11 ; 8/13

7)  8/24 → 0/5 ; 8/11 ; 8/13

Ciclo 1: Fundamentos de programación con Python

Principios de pensamiento computacional



Ejercicio 2.

Dos empresas tienen ofertas de trabajo iguales en términos de las responsabilidades de los mismos, con la diferencia de la forma de retribución económica, las ofertas son:

Empresa A: \$100'000.000 pagaderos anualmente con aumento anual de \$20'000.000

Empresa B: \$50'000.000 pagaderos semestralmente con aumento semestral de \$5'000.000

¿Cual trabajo escoge?



Ciclo 1: Fundamentos de programación con Python

Principios de pensamiento computacional



Posible solución:

Año 1			
Empresa A	\$100.000.000,00		\$100.000.000,00
Empresa B	\$50.000.000,00	\$55.000.000,00	\$105.000.000,00
Año 2			
Empresa A	\$120.000.000,00		\$120.000.000,00
Empresa B	\$60.000.000,00	\$65.000.000,00	\$125.000.000,00
Año 3			
Empresa A	\$140.000.000,00		\$140.000.000,00
Empresa B	\$70.000.000,00	\$75.000.000,00	\$145.000.000,00



Ciclo 1: Fundamentos de programación con Python

Principios de pensamiento computacional



Ejercicio 3.

El hotel PUJA Resort ha decidido implementar un sistema de automatización de edificios, que cuente con seguridad biométrica, sistema de calefacción automático y apertura de puertas autónomo. Usted hace parte del equipo de implementación de soluciones para puertas automáticas de la empresa de seguridad “security in SAS”.

Su trabajo es desarrollar un algoritmo que controle los accesos en las puertas automáticas, y así lograr la apertura de las mismas.



Ciclo 1: Fundamentos de programación con Python

Introducción al lenguaje de programación Python



Ciclo 1: Fundamentos de programación con Python

Introducción al lenguaje de programación Python



Python es un lenguaje de programación de propósito general muy poderoso y flexible, a la vez que sencillo y fácil de aprender.

Este lenguaje fue creado a principios de los noventa por Guido van Rossum en los Países Bajos.

Python se desarrolla bajo una licencia de Open source o código abierto aprobada por OSI, por lo que se puede usar y distribuir libremente, incluso para uso comercial.

El Python Package Index (PyPI) o en español significa Índice de paquetes de Python alberga miles de módulos de terceros para Python.

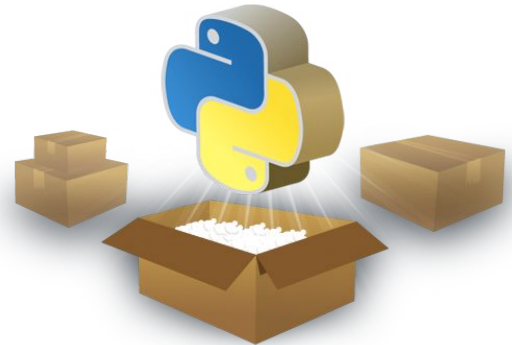
Ciclo 1: Fundamentos de programación con Python

Introducción al lenguaje de programación Python



Tanto la biblioteca estándar de Python como los módulos aportados por la comunidad permiten infinitas posibilidades.

- Desarrollo web e Internet.
- Acceso a la base de datos.
- GUIs de escritorio.
- Científico y numérico.
- Educación.
- Programación de red.
- Desarrollo de Software y Juegos.



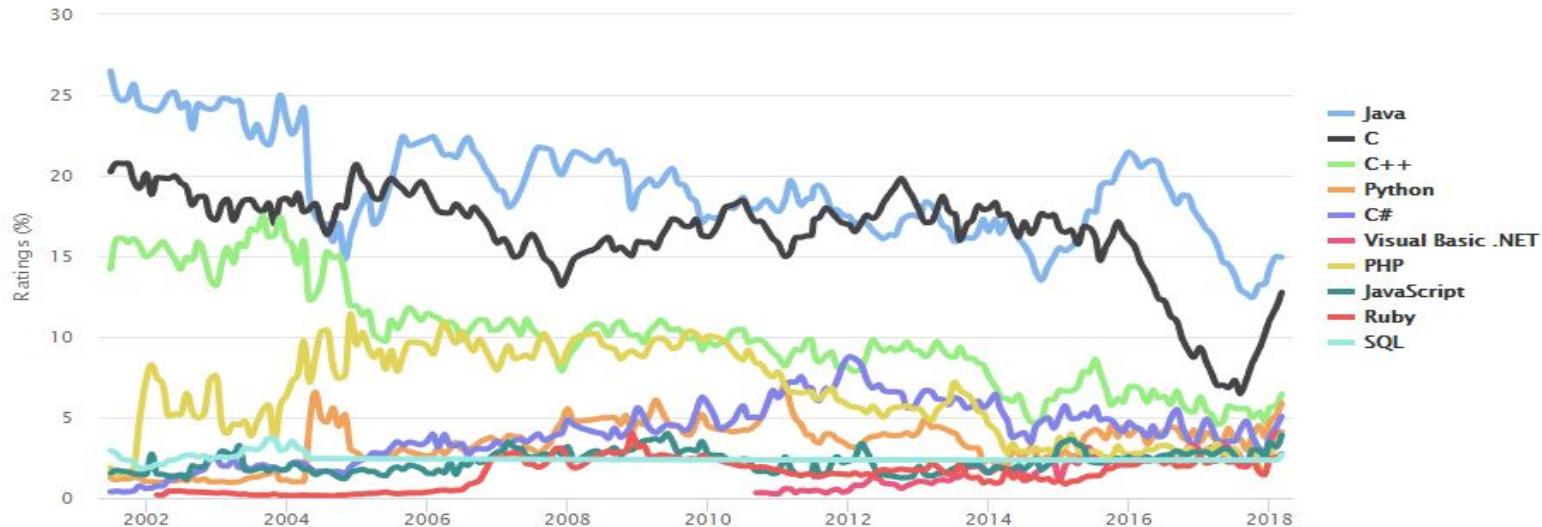
Ciclo 1: Fundamentos de programación con Python

Introducción al lenguaje de programación Python



TIOBE Programming Community Index

Source: www.tiobe.com



Ciclo 1: Fundamentos de programación con Python

Principios de pensamiento computacional



Programación orientada a objetos

La Programación Orientada a Objetos (POO) es un paradigma de programación, es decir, un modelo o un estilo de programación que nos da unas guías sobre cómo trabajar con él. Se basa en el concepto de clases y objetos. Este tipo de programación se utiliza para estructurar un programa de software en piezas simples y reutilizables de planos de código (clases) para crear instancias individuales de objetos.

Con el paradigma de Programación Orientado a Objetos lo que buscamos es dejar de centrarnos en la lógica pura de los programas, para empezar a pensar en objetos, lo que constituye la base de este paradigma. Esto nos ayuda muchísimo en sistemas grandes, ya que en vez de pensar en funciones, pensamos en las relaciones o interacciones de los diferentes componentes del sistema.

Ciclo 1: Fundamentos de programación con Python

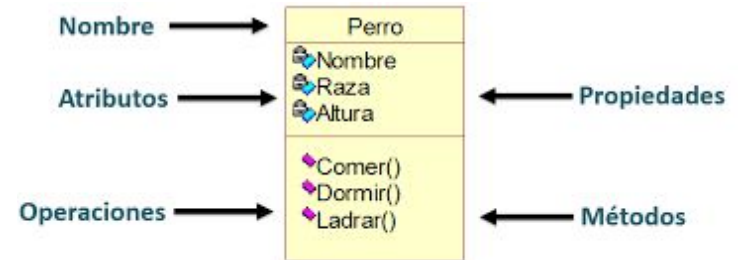
Principios de pensamiento computacional



Programación orientada a objetos

¿Por qué POO?

La Programación Orientada a Objetos permite que el código sea reutilizable, organizado y fácil de mantener. Sigue el principio de desarrollo de software utilizado por muchos programadores DRY (Don't Repeat Yourself), para evitar duplicar el código y crear de esta manera programas eficientes. Además, evita el acceso no deseado a los datos o la exposición de código propietario mediante la encapsulación y la abstracción.



Ciclo 1: Fundamentos de programación con Python

Principios de pensamiento computacional



Programación orientada a objetos

Clases, objetos e instancias

¿Cómo se crean los programas orientados a objetos? Resumiendo mucho, consistiría en hacer clases y crear objetos a partir de estas clases. Las clases forman el modelo a partir del que se estructuran los datos y los comportamientos.

El primer y más importante concepto de la POO es la distinción entre clase y objeto.

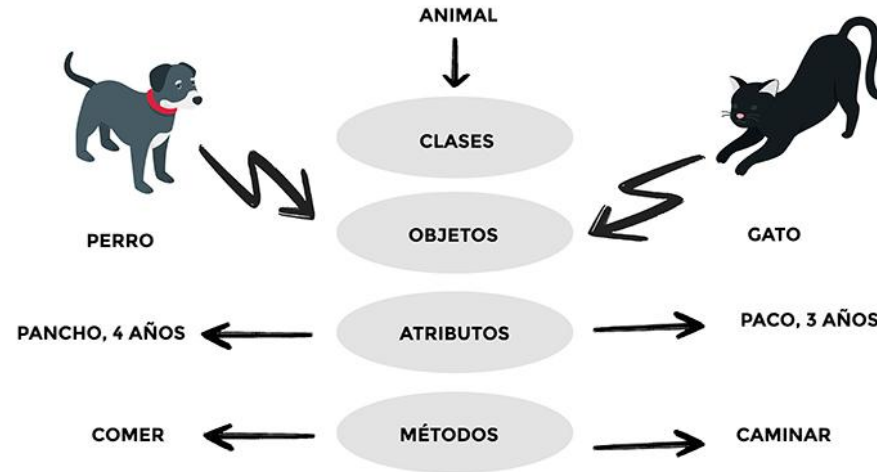
Una clase es una plantilla. Define de manera genérica cómo van a ser los objetos de un determinado tipo. Por ejemplo, una clase para representar a animales puede llamarse 'animal' y tener una serie de atributos, como 'nombre' o 'edad' (que normalmente son propiedades), y una serie con los comportamientos que estos pueden tener, como caminar o comer, y que a su vez se implementan como métodos de la clase (funciones).

Ciclo 1: Fundamentos de programación con Python

Principios de pensamiento computacional



Programación orientada a objetos



Ciclo 1: Fundamentos de programación con Python

Principios de pensamiento computacional



Ciclo 1: Fundamentos de programación con Python

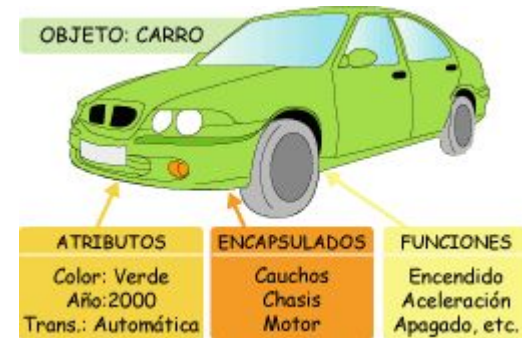
Principios de pensamiento computacional



Principios de la programación orientada a objetos

Encapsulación: La encapsulación contiene toda la información importante de un objeto dentro del mismo y solo expone la información seleccionada al mundo exterior.

Esta propiedad permite asegurar que la información de un objeto esté oculta para el mundo exterior, agrupando en una Clase las características o atributos que cuentan con un acceso privado, y los comportamientos o métodos que presentan un acceso público.



Ciclo 1: Fundamentos de programación con Python

Principios de pensamiento computacional



Principios de la programación orientada a objetos

La abstracción

La abstracción es cuando el usuario interactúa solo con los atributos y métodos seleccionados de un objeto, utilizando herramientas simplificadas de alto nivel para acceder a un objeto complejo.



Ciclo 1: Fundamentos de programación con Python

Principios de pensamiento computacional



Principios de la programación orientada a objetos

La herencia

La herencia define relaciones jerárquicas entre clases, de forma que atributos y métodos comunes puedan ser reutilizados. Las clases principales extienden atributos y comportamientos a las clases secundarias. A través de la definición en una clase de los atributos y comportamientos básicos, se pueden crear clases secundarias, ampliando así la funcionalidad de la clase principal y agregando atributos y comportamientos adicionales.

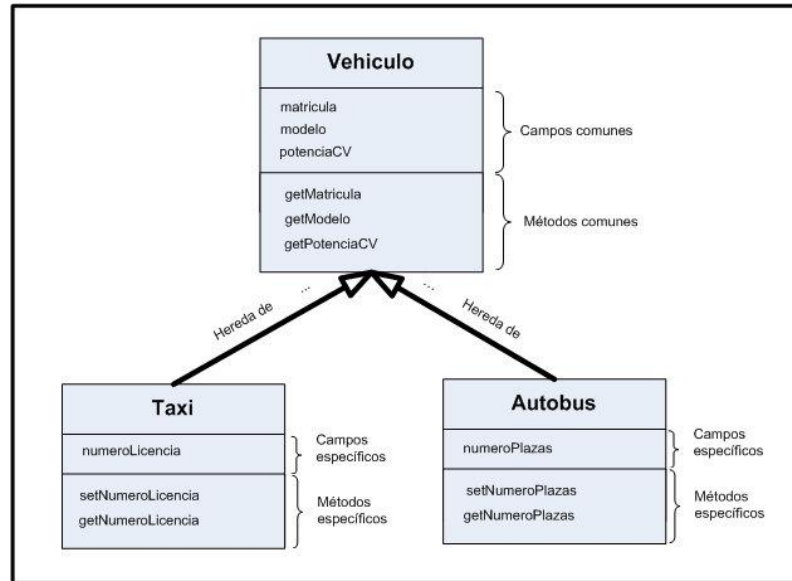
Ciclo 1: Fundamentos de programación con Python

Principios de pensamiento computacional



Principios de la programación orientada a objetos

La herencia



Ciclo 1: Fundamentos de programación con Python

Principios de pensamiento computacional



Beneficios de Programación Orientada a Objetos

- Reutilización del código.
- Convierte cosas complejas en estructuras simples reproducibles.
- Evita la duplicación de código.
- Permite trabajar en equipo gracias al encapsulamiento ya que minimiza la posibilidad de duplicar funciones cuando varias personas trabajan sobre un mismo objeto al mismo tiempo.
- Al estar la clase bien estructurada permite la corrección de errores en varios lugares del código.
- Protege la información a través de la encapsulación, ya que solo se puede acceder a los datos del objeto a través de propiedades y métodos privados.
- La abstracción nos permite construir sistemas más complejos y de una forma más sencilla y organizada.



Ciclo 2: Fundamentos de programación con Python

POO (Programación Orientada a Objetos)



La programación orientada a objetos (POO) es una forma especial de programar, más cercana a como expresaríamos las cosas en la vida real que otros tipos de programación.



OBJETOS



Ciclo 2: Fundamentos de programación con Python

POO (Programación Orientada a Objetos)



¿ Cómo pensar en objetos ?



Atributos (características):

- color
- marca
- kmRecorridos

Métodos (Acciones)

- encender
- acelerar
- frenar



Ciclo 2: Fundamentos de programación con Python

POO (Programación Orientada a Objetos)



¿ Cómo pensar en objetos ?

$\frac{3}{2}$

Atributos (Características):

- numerador
- denominador

Métodos (Acciones)

- simplificarse
- sumarse con otra fracción
- restarse con otra fracción





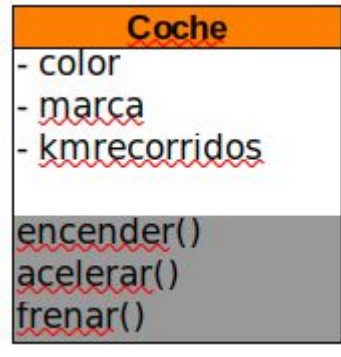
Ciclo 2: Fundamentos de programación con Python

POO (Programación Orientada a Objetos)



Clases en POO

Son un conjunto de objetos con características similares



coche1

color: Blanco
marca: Audi
kmrecorridos: 0



coche2

color: Rojo
marca: Ferrari
kmrecorridos: 100

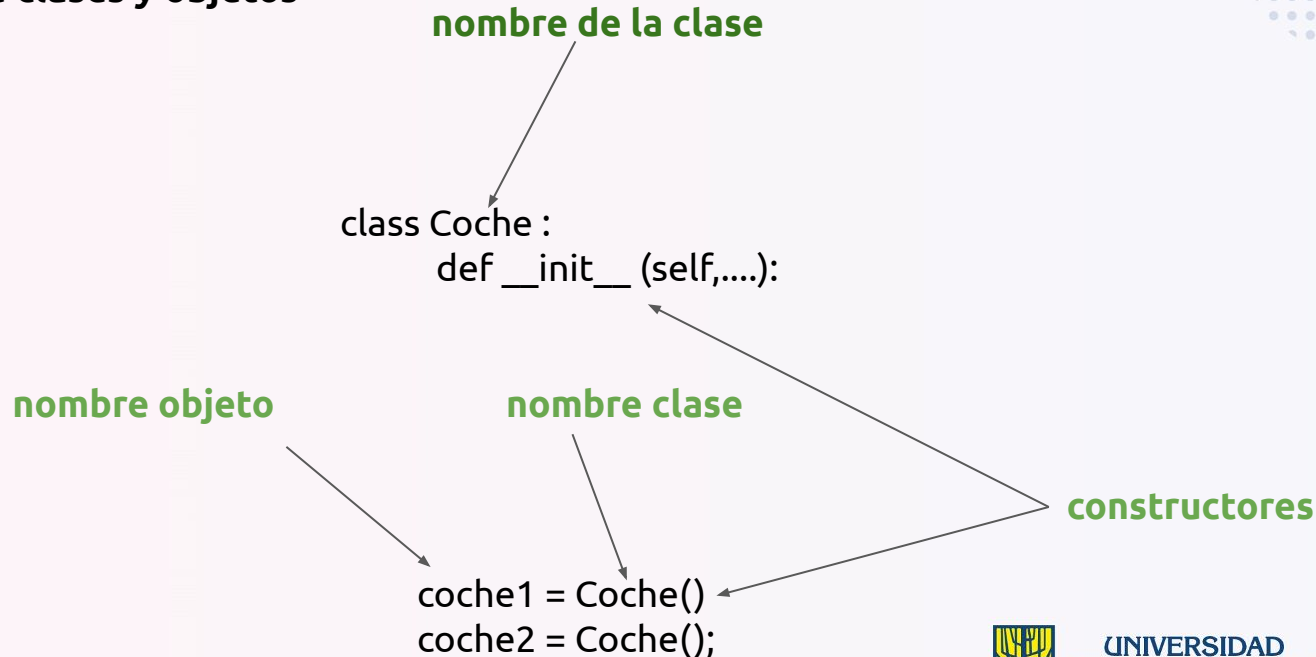


Ciclo 2: Fundamentos de programación con Python

POO (Programación Orientada a Objetos)



Creación de clases y objetos





Ciclo 2: Fundamentos de programación con Python

POO (Programación Orientada a Objetos)



Ejemplo Visual Studio Code - clase Coche

- `poo.py`
- `superPoo.py`
- `superPooHerencia.py`



Ciclo 1: Fundamentos de programación con Python

Realización de operaciones aritméticas en programación



Ciclo 1: Fundamentos de programación con Python

Realización de operaciones aritméticas en programación



¿Qué son los operadores aritméticos?

En programación y matemáticas, los operadores aritméticos son aquellos que manipulan los datos de tipo numérico, es decir, permiten la realización de operaciones matemáticas (sumas, restas, multiplicaciones, etc.).

El resultado de una operación aritmética es un dato aritmético, es decir, si ambos valores son números enteros el resultado será de tipo entero; si alguno de ellos o ambos son números con decimales, el resultado también lo será.

Ciclo 1: Fundamentos de programación con Python

Realización de operaciones aritméticas en programación



Operación	Operador	Descripción	Ejemplo	Resultado
Suma	+	Obtiene el resultado de sumar los operandos	$s=5+4$	9
Resta	-	Obtiene la diferencia entre los operandos	$r=5-4$	1
Multiplicación	*	Obtiene el producto entre los operandos	$m=5*4$	20
División	/	Obtiene la división según el tipo de dato entre los operandos. Entero / entero: genera un entero Real / real; Entero / real; real / entero: genera un real	$d=5/2$	2

Ciclo 1: Fundamentos de programación con Python

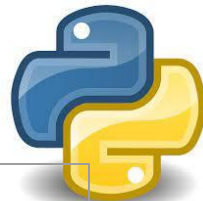
Realización de operaciones aritméticas en programación



Operación	Operador	Descripción	Ejemplo	Resultado
División modular	%	Obtiene el residuo de una división entera. Por lo tanto los operadores únicamente pueden ser enteros	dm=5%2	1
Potencia	**	Obtiene la potencia de la base elevada al exponente. La radicación puede obtenerse del inverso de la potencia.	p=5**2	25
División parte entera	//		dpa=15//2	7

Ciclo 1: Fundamentos de programación con Python

Realización de operaciones aritméticas en programación



Prioridad en operadores

Prioridad	Operadores
Mayor	()
...	^
...	*/
Menor	+ -

Ciclo 1: Fundamentos de programación con Python

Realización de operaciones aritméticas en programación



Prioridad en operadores

Ejercicio:

- $6/2*(2+1)$

- $6/2*2+1$

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

$$a=1, b=2, c=1$$

Ciclo 1: Fundamentos de programación con Python

Realización de operaciones aritméticas en programación



Prioridad en operadores

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \quad a=1, b=2, c=1$$

$$x = (-b + (b^2 - 4*a*c)^{(1/2)}) / (2*a)$$

$$x = (-b - (b^2 - 4*a*c)^{(1/2)}) / (2*a)$$

Por lo tanto se evaluará cada una, considerando primero los paréntesis y dentro de los paréntesis por prioridades; así, la respuesta para la primera ecuación si a=1, b=2, c=1, sería:

$$x = (-2 + (2^2 - 4*1*1)^{(1/2)}) / (2*1)$$

En esta línea $4*1*1$ tienen la misma prioridad, entonces $4*1 = 4$ y la respuesta por 1.

$$x = (-2 + (4 - 4)^{(0.5)}) / (2)$$

$$x = (-2 + 0^{0.5}) / 2$$

$$x = (-2 + 0) / 2$$

$$x = -2 / 2$$

$$x = -1$$

Ciclo 1: Fundamentos de programación con Python

Realización de operaciones aritméticas en programación



Operadores de asignaciones

Operador	Descripción	Ejemplo
=	asigna valor a una variable	<pre>>>> r = 5 >>> r1 = r</pre>
+=	suma el valor a la variable	<pre>>>> r = 5 >>> r += 10; r 15</pre>
-=	resta el valor a la variable	<pre>>>> r = 5 >>> r -= 10; r -5</pre>
*=	multiplica el valor a la variable	<pre>>>> r = 5 >>> r *= 10; r 50</pre>
/=	divide el valor a la variable	<pre>>>> r = 5 >>> r /= 10; r 0</pre>

Ciclo 1: Fundamentos de programación con Python

Realización de operaciones aritméticas en programación



Operadores de asignaciones

<code>**=</code>	calcula el exponente del valor de la variable	<pre>>>> r = 5 >>> r **= 10; r 9765625</pre>
<code>//=</code>	calcula la división entera del valor de la variable	<pre>>>> r = 5 >>> r //= 10; r 0</pre>
<code>%=</code>	devuelve el resto de la división del valor de la variable	<pre>>>> r = 5 >>> r %= 10; r 5</pre>

Ciclo 1: Fundamentos de programación con Python

Realización de operaciones aritméticas en programación



Operadores relacionales

Operador	Descripción	Ejemplo
==	¿son iguales a y b?	<pre>>>> 5 == 3 False</pre>
!=	¿son distintos a y b?	<pre>>>> 5 != 3 True</pre>
<	¿es a menor que b?	<pre>>>> 5 < 3 False</pre>
>	¿es a mayor que b?	<pre>>>> 5 > 3 True</pre>
<=	¿es a menor o igual que b?	<pre>>>> 5 <= 5 True</pre>
>=	¿es a mayor o igual que b?	<pre>>>> 5 >= 3 True</pre>

Ciclo 1: Fundamentos de programación con Python

Realización de operaciones aritméticas en programación



Ejercicios de algoritmos

- Realice un algoritmo para determinar si en la variable **a** existe un número múltiplo de 5. En caso afirmativo asigne a la variable **b** el número 1, de lo contrario asigne a la variable **b** el número 0.
- Suponiendo que en la variable **a** existe un número entre 1 y 99 elabore un algoritmo que almacene en la variable **b** la suma de los dígitos que componen el número almacenado en **a**. Por ejemplo si **a** tiene el dato 82, en la variable **b** debe quedar el número 10 ya que 8 más 2 es 10.
- En las variables **a**, **b**, **c** existen 3 números en el rango de 0 a 9. Conforme un número en la variable **x**, compuesto por los dígitos almacenados en **a**, **b** y **c**. Por ejemplo si en las variables **a**, **b** y **c** existen los dígitos 8, 5, 3, en la variable **x** debe quedar el número 853.
- Dados dos datos almacenados en las variables **a** y **b**, elabore un algoritmo que intercambie los valores almacenados en **a** y **b**, sin necesidad de utilizar ningún tipo de variable auxiliar.

Ciclo 1: Fundamentos de programación con Python

Realización de operaciones aritméticas en programación



Ejercicios de algoritmos

- Diseñe Un Algoritmo que lea tres longitudes y determine si forman o no un triángulo. Si es un triángulo determine que tipo de triángulo se trata entre: Equilatero (si tiene tres lados iguales), isosceles (si tiene dos lados iguales) o escaleno (si tiene tres lados desiguales).