INTRODUCCIÓN A PYTHON

E01

Contenido

“

Este espacio curricular permite fortalecer el espíritu crítico y la actitud creativa del futuro graduado. Lo alienta a integrar los conocimientos previos, recreando la práctica en el campo real, al desarrollar un proyecto propio.

”

Introducción al pensamiento Computacional

Nos detenemos un momento en este lugar, no quisiera avanzar sin darles un concejo. Programar es una de las actividades más emocionantes que pueden vivir, el placer que se obtiene al ver el producto terminado es único. Pero los comienzos son algo turbulentos, y suele cuásar algo de estrés.

Los primeros pasos son los más importantes, pero también los más difíciles. Pueden tener algunas frustraciones, pero a no abandonar. La curva de aprendizaje al principio puede exigir mucha dedicación y esfuerzo. Van a sentir que dedican muchas horas de estudio y avanzan poco, pero es normal. Créanme que, en algún momento, esa ecuación cambia y se invierte por completo.

Ese punto de inflación llega, se los puedo asegurar. ¿Pero cuando? Bueno eso depende de Uds. y de las horas que les dediquen a sus estudios. Por eso digo que programar es un ejercicio de paciencia, es tiempo, es estudio… pero al final, los frutos lo valen…

¡¡Así que... A disfrutar... Que, al terminar el cuatrimestre, ¡¡estaremos disfrutando con nuestros propios proyectos!!

Presentación

Es esta asignatura, vamos a incursionar en el desarrollo de proyectos usando Python como lenguaje de programación. El desarrollo involucra competencias no solo de programación, sino también capacidades de abstracción y creatividad

El mundo del desarrollo está dominado por muchos lenguajes como C++ o C#, pero hoy en día Python es un lenguaje que toma mucha fuerza en la industria.

Aprender un lenguaje de programación es muy fácil, pero, difícil a la vez. Depende de Uds. y de las horas de práctica que dediquen a esta asignatura. Por eso digo que programar es un ejercicio de paciencia, es tiempo, es estudio… pero al final, se puede aprender…

Así que... A disfrutar... Que, al terminar el cuatrimestre, ¡¡estaremos jugando con nuestros propios juegos o disfrutando de nuestras producciones!!

Contenidos mínimos

Esta asignatura que estas cursando, es una de las primeras en toda la carrera. Por eso mismo, daremos pasos cortos y muy pausados. vamos a dividir cada encuentro en Teoría y práctica. Durante el desarrollo de la teoría, nos adentraremos en distintos paradigmas de programación, pero eso lo veremos luego.

A modo de introducción Podemos resumir los contenidos mínimos de la asignatura en estos 4 focos. En todos los casos, el alcance de cada tema será presentado tanto en su teoría como en la práctica. Y en pocas palabras podemos decir que:

Daremos una introducción a las estructuras básicas, que en general aplican a cualquier lenguaje. Pero en Python, algunas cuestiones relacionadas a su sintaxis lo hacen un poco más complejo al principio. Aquí, veremos cómo usar variables, cadenas, colecciones. Veremos cómo aprovechar las estructuras de decisión y los bucles de repetición. Terminaremos programando funciones y el uso de algunas librerías elementales de Python como también instrucciones típicas de entrada y salida que brinda el lenguaje.

Es muy importante, tanto en Python como en cualquier otro lenguaje, trabajar en las competencias que nos permiten representar con un nivel adecuado de abstracción cualquier solución a los problemas planteados. Aunque no es un tema de la asignatura, trabajaremos indirectamente sobre algunos tópicos de algoritmia que pueden ser útiles en el futuro.

Como introducción, podemos decir que, en los últimos años, fueron apareciendo distintos lenguajes para desarrollar una aplicación digital. Cada vez que programamos, usamos un paradigma de programación. Podemos decir que un paradigma es un conjunto de acuerdos que guían al desarrollador al momento de sentarse a programar. El

problema es que, como ocurre siempre, estos acuerdos suelen cambiar con el tiempo. En la actualidad, algunos paradigmas están más arraigados en la comunidad que otros. Lo interesante de Python es que nos permite trabajar con varios de ellos.

Continuado, les cuento que algunos lenguajes como C, brindan un conjunto limitado de funciones. Pero, permiten importar paquetes para potenciar el desarrollo. Python no es diferente, y en esta asignatura veremos varias de las librerías del lenguaje.

Criterios de aprobación

Tenía que llegar, hablemos de la parte fea de estudiar... ¿Cómo hacemos para aprobar?

Para aprobar, tendrás que sumar créditos en varios desafíos, en todos los casos la nota debe ser igual o superior a 4 (cuatro).

Desafíos Semanales: Cada clase, tendrás actividades para completar, muchas de ellas se autoevalúan. Pero son muy importante para que vos mismo puedas ver si la base de conocimiento esta firme.

Trabajos prácticos: son dos trabajos integradores, lo deberás presentar antes de finalizar el cuatrimestre y previo a cada evaluación. Cada instancia de entrega tendrá también una defensa Grupal del trabajo y su contenido.

Exámenes: Tendremos dos instancias de evaluación, una en la clase 7 y la otra a fines del cuatrimestre. Más precisamente en la clase 17. Los temas que se observarán en cada instancia serán todos los temas vistos hasta el momento en cada uno de los encuentros.

Por último, el **presentismo**, recuerda que no puedes tener menos del 75% de asistencia.!! ¿Esto es lo más fácil, no me digas que no puedes cumplirlo? 😉

¡¡Si al terminar el cuatrimestre cumpliste y aprobaste todo esto, felicitaciones!! La materia **es tuya…**

Trabajos Prácticos

Te cuento como será el trabajo práctico. Se entrega al terminar el cuatrimestre y antes de cada parcial, pero es muy sencillo. Haremos un programa sencillo integrando todos los temas y un programa para Analizar de datos.

El objetivo es que desarrolles una solución que use lo visto en la materia, hasta ese momento. Vamos a ir viendo en cada clase, los temas... ¡Pero terminarlo va a depender solo de vos!

La originalidad será premiada, y todo lo que hagas para potenciar el proyecto será valorado también. A veces nos traiciona el acceso a Internet, porque seguro ya habrá en la red varios ejemplos resueltos. No es malo ver el trabajo de otros. Pero si quieres aprender, mi recomendación es que programes tu propio proyecto.

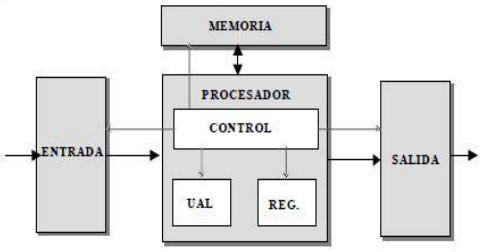
Confío en tu buen juicio 😉

Calendario

Te voy a dejar en el campus un calendario de clases y contenidos, es sólo una guía para ordenarnos. Todos sabemos que la vida no es tan estática, y por supuesto el calendario tampoco…

Encuentro 1

La computadora



La computadora es la herramienta que usamos para programar, conocer sus partes y la utilidad es muy importante para desarrollar soluciones optimas a los distintos problemas.

Cada computadora tiene una CPU, esta es la que se encarga de ordenar a todas las partes de la PC y además de ejecutar el lenguaje de programación. Cada orden, la realiza en un ciclo de cómputo. La idea de un buen programa es usar la mejor cantidad de esos ciclos. La memoria RAM, es donde se guardan todos los datos que usa la PC. Nuevamente, la idea de un buen programa es hacer un uso adecuado del espacio de esta memoria.

Entorno de Trabajo

1. Instalar Python
2. Instalar Vs Code
3. Agregar Extensiones

Vamos a llamar entorno de trabajo al ecosistema de software que instalamos en nuestra PC. Al momento de tomar esta decisión, debemos pensar en que sistema operativo vamos a trabajar, y luego cual es el software que necesitamos.

En este caso, solo necesitamos un IDE (Integrated Development Environment). Son muchos los entornos posibles, te presento a continuación algunos de ellos:

Visual Studio

"IDE de 64 bits más rápido y confiable. Incluye .NET 6, incluida la compatibilidad con MAUI, las aplicaciones Blazor y Recarga activa. Finalizaciones mejoradas de IntelliCode. Cadena de herramientas más reciente para el destino en C++20 [1]". Tenemos varias versiones profesionales de pago y una versión gratuita llamada Visual Studio Community.

Xcode es una IDE muy potente para usar en OS Mac, deberás configurarlo, pero es muy práctico para desarrollar.

Pycharm es una alternativa interesante si no sos amante de usar un editor de Microsoft y te gusta estar menos atado al sistema operativo. Tiene licencias educativas por lo que podrás probarlo. La empresa que lo desarrolla es JetBrain.

Otra alternativa a Visual Studio con una versión más liviana es VSCode[2]. Ofrece casi las mismas utilidades que su padre Visual Studio, tal vez te cueste un poco al principio si usabas la versión profesional.

VSCODE Visual Studio Code

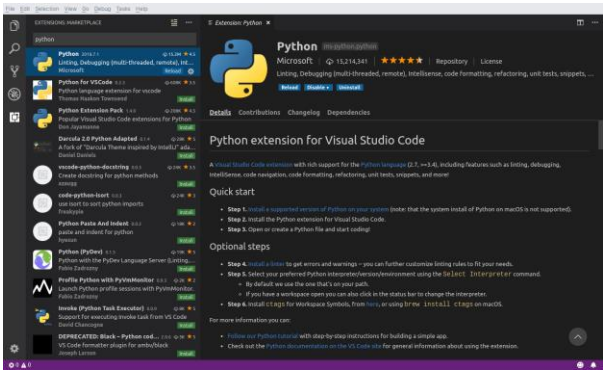
[1] https://visualstudio.microsoft.com/es/

[2] https://code.visualstudio.com/

Ahora es cuando, saltamos la democracia. En esta asignatura, no hay elección vamos a trabajar con VsCode.

Instalación de Vscode

Lo primero que haremos es descargar el instalador, para eso entramos a la página https://code.visualstudio.com/ y buscamos el archivo para instalar en la versión de OS que queremos y en el idioma que más nos guste.



Instalación de Python

Algo importante que debemos saber, es que VSCode es un IDE multi uso. Para usar Python, debemos descargar un intérprete.

Pasos que debemos seguir:

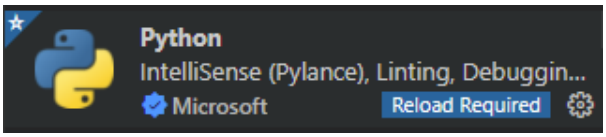
1- Descargar el intérprete de la página de descarga. Es importante descargar la versión 3 o superior. https://www.python.org/downloads/
2- Siga los pasos de instalación.

Agregar Extensiones

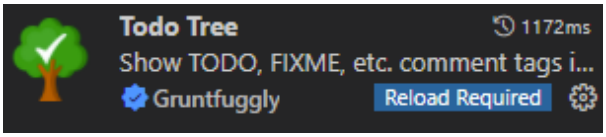
Para mejorar la experiencia de trabajo, vamos a instalar algunas extensiones. En este paso, hacemos una pausa. Vas a encontrar muchas extensiones, siempre es aconsejable instalar aquellas que tiene un autor conocido.

primero seleccionamos el icono de extensiones del editor

Luego en el buscador escribimos Python, cuando encontramos la extensión de Python la instalamos



Como adicional, te invito a instalar esta extensión de administración de comentarios que usaremos luego para mantenimiento del código.



Para ayudarte, te comparte este video

https://youtu.be/Pcn8nUBJyPQ

Lenguajes de Programación

Si lo vemos desde un punto de visto mucho más amplio, Podemos decir que el lenguaje es un conjunto de símbolos escritos u orales por medio del cual dos personas pueden comunicarse.

¿Cuál es el sentido de un lenguaje?

Podemos decir que el fin de un lenguaje es permitir que dos personas puedan comunicar un determinado asunto, con el fin de compartir o ponerse de acuerdo. Es claro, y creo que no abrimos debate con esto, para que ambas personas se comuniquen deben comprender el mismo lenguaje. En el caso contrario, la comunicación puede tornarse un poco más compleja o imposible.

Ahora bien, esto se complica más para los desarrolladores.

¿y nos preguntamos por qué?

Porque cuando nos referimos a programar una computadora, no se trata de lograr que dos personas con un promedio de intelectualidad razonable puedan de comunicarse entre sí. Es, mucho más complejo. En este caso, necesitamos comunicarnos con una computadora… y claro, la computadora no es muy amigable con quienes no hablan el mismo lenguaje.

Es por ello, que tenemos que usar lo que llamamos un “lenguaje de programación”. En el mundo de la programación, son muchos lenguajes, pero en esta asignatura sólo usaremos Python.

¿Qué es un algoritmo?

Ya hemos cubierto las bases para entender porque necesitamos un lenguaje de programación, pero como podemos desarrollar programas. Es aquí donde aparecen los algoritmos.

¿Cómo pasamos la idea al lenguaje de programación?

El lenguaje no es suficiente para lograr explicarle a la computadora lo que queremos, necesitamos seguir una lógica y respetar algunas premisas. Esa lógica, es el algoritmo

Cuando hablamos de algoritmos, decimos que es un conjunto ordenado de pasos que permiten obtener un resultado o encontrar la solución a un problema. Simplificando, decimos que, partiendo de un conjunto de datos de entrada, siguiendo pasos sucesivos no ambiguos se llega a un resultado final y se obtiene la solución al problema. Con el tiempo, vamos a estandarizar procesos algorítmicos para cada problema y la practica hará que encuentres esa muy nombrada solución mucho más rápido.

A lo largo de todo este cuatrimestre, vamos a ejercitar nuestras competencias para resolver por medio de algoritmos, todo tipo de situaciones.

Para entender que es un algoritmo, este es un conjunto de palabras reservadas del lenguaje de programación. Ese conjunto de palabras forma un código/solución. Todos los

algoritmos deben cumplir básicamente tres premisas simples:

1. **PRECISO** (Orden)
2. **DEFINIDO** (Resultado)
3. **FINITO**

Cuando decimos que un algoritmo, es un conjunto de pasos no ambiguos, queremos decir que debe respetar por lo menos estas tres premisas

Que sea Preciso: Implica el orden de realización de cada uno de los pasos.

Que sea Definido: Si se sigue dos veces, se obtiene el mismo resultado.

Que sea Finito: Tiene un número determinado de pasos, implica que tiene un fin.

Veamos un ejemplo …

ALGORITMO "Hacer del 2"

1. Caminar hacia el baño
2. Abrir la puerta
3. Sacarse los pantalones o cualquier vestimenta que tenga puesta
4. Sentarse en el inodoro
5. Hacer fuerza
6. Usar el papel Higiénico
7. Vestirse
8. Salir del baño

Clasificación de los lenguajes

Antes de comenzar con un lenguaje de programación, en este caso Python, tratemos de entender que es un lenguaje que se puede usar en una computadora

Cuando nace la computadora, esta solo entendía una cosa – **"Encendido/apagado"**. Esto, fue conocido luego como el código binario representando 1 (unos) y 0 (ceros).
La unidad más pequeña que se tenía era el bit. Sencillamente, este bit, solo podría indicar cuando algo estaba encendido (tenía energía) y cuando algo estaba apagado (no tenía energía).

la evolución de la tecnología demandada utilizar mucha más información, y tener solo un cero o un uno, no era suficiente. Para afrontar esta necesidad, comenzaron a representar datos con grupos de más de un solo digito. Es decir, usaban muchos ceros y unos. Para dejarlo mucho más claro, se juntaron muchos bits. La historia es más larga, pero vamos a quedarnos solo con la noticia que se juntaron 8 bits en un solo grupo. Así llegamos al Byte.

El Byte, era un grupo de ceros y unos con un total de 8 bits. La combinación de 8 bits les permitía representar hasta 256 datos diferentes, que no era poca cosa.

Uds. se preguntarán, **¿se puede lograr mucho con un Byte?** —y la respuesta es sí!!—

Todos los caracteres que se encuentran en el teclado están representados en una tabla que se llama tabla ASCII, en total son 256 caracteres; **¿esto te responde la pregunta?**

Pues bien, no es intención de esta asignatura dar una clase de historia de la computación digital. Es por eso, que dejamos para otra materia la responsabilidad de explicar cómo llegamos del bit al teraByte 😉

En fin, solo nos interesa saber que la computadora solo conoce el código binario.

Entonces, ¿Dónde entra Python?

Te lo explico a continuación.

Es raro que programemos en el lenguaje que la computadora entiende, ceros y unos. Para evitar programar en un modo tan complejo, se crearon lenguajes de programación. Según el estilo de la sintaxis del lenguaje se pueden clasificar en **Bajos** (cercanos a la PC), **Medios** y **Altos** (cercanos al humano). La diferencia es que cuanto más cerca estemos del nivel Alto, las palabras que formen parte del conjunto de comandos del lenguaje serán más claras para el humano y menos entendible para la computadora. Esto último, nos pone frente a un desafío.

Hoy, programamos siempre en lenguajes de Alto nivel. Pero si un día quisieras programar directamente sobre una placa de video, tal vez tengas que usar lenguajes de bajo nivel.

Entonces… ¿Cuál es el desafío?

Tenemos que programar en un lenguaje de alto nivel para luego convertirlo en ceros y unos. Este problema, se resuelve usando compiladores o intérpretes. Esto lo veremos más adelante, pero primero, veamos ejemplos de lenguajes de bajo, medio y alto nivel.

Lenguaje máquina

Como comentamos anteriormente, el leguaje de la computadora es el Código binario. Este es el único lenguaje que la PC entiende. Para el humano, no es tan sencillo. Además, es difícil de mostrar… para resumir… raro es que programemos usando el código en binario.

Lenguaje máquina

0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1

De todos modos, todo dentro del pc está programado en ceros y unos. Pero, si esto te parecía raro, bueno todavía hay más. El código binario, no es el único modo de mostrar los datos que están dentro de la computadora. Tenemos también el código Hexadecimal.

¿Qué diferencia hay entre estos dos códigos?

El código binario tiene base 2, mientras que el Hexadecimal tiene base 16. Eso significa que en binario solo podemos usar dos números el cero y el uno. Sin embargo, en hexadecimal tenemos 16. Podemos usar del 0 al 9 y luego

usamos letras. Donde 10 es una A y 15 es una F. Pero, este hexadecimal, es solo para mostrar los datos cuando abrimos un archivo compilado. Y esto se usa, porque ocupa menos espacio en la pantalla… fin del comunicado.

Lenguaje de Alto nivel

Para resumir, lenguajes como Python, C# o java son considerados de alto nivel.

cuando el objetivo sea desarrollar un programa, usaremos lenguajes de alto nivel. Su propósito radica en poder desarrollar sin un alto grado de dependencia con el hardware que se utilice. Obviamente, usar palabras más simples. Y esta simpleza, tiene como aspecto negativo que el lenguaje de alto nivel se aleja del código máquina. Entonces, los lenguajes de alto nivel se tienen que convertir en binario.

Como programamos alejados del lenguaje de la máquina, necesitamos un programa traductor que se encargue de convertir este lenguaje de alto nivel. Este proceso de transformación se conoce como compilación o interpretación el código fuente. Es decir, necesitamos un programa que traduzca el lenguaje de alto nivel convirtiéndolo a lenguaje máquina.

Lenguajes interpretados o Compilados

A esta altura ya sabemos que es un lenguaje de programación. Cuando creamos el archivo (con Python), lo que estamos creando es nuestro código Fuente. El código Fuente es la base de todo nuestro trabajo, y generalmente lo que más valor tiene para nosotros.

El código fuente sufre un proceso de transformación para traducirlo al lenguaje de la máquina, pero esto es cuando lo queremos ejecutar y hacerlo funcionar. Hasta allí, no vemos un problema latente. Pero, cuando ese mismo código lo tiene que ejecutar otra persona comienza la parte difícil. Una de las razones, es que no quiero compartir mi código fuente, solo quiere compartir el resultado de mi código fuente. Cuando se quiere compartir el producto con otra persona, el código tiene que pasar por un proceso de traducción. Generalmente, esta implementación se resuelve convirtiendo el código fuente un programa ejecutable (compilar).

Esto parece resolver el problema de compartir mi producto, si el código fuente lo convierto en un .EXE el código fuente se vuelve ilegible. El punto importante que tenemos que considerar es que a veces los lenguajes de programación no se pueden compilar. En estos casos el lenguaje es solo interpretable.

Para entender esto último, veamos una definición muy simple de cada concepto:

Interpretado: el proceso de traducción a lenguaje máquina se hace cada vez que se ejecuta. Es decir, no tengo un .EXE

Compilado: El proceso de traducción se hace por única vez, luego de obtener el .EXE ese archivo se usan sin necesidad de volver a compilar.

Un ejemplo de esto es javascript, este lenguaje no es un lenguaje compilable. Esto significa que no obtengo un .EXE. Para cerrar la idea, javascript es un lenguaje interpretado. Para comprobarlo, sólo debemos entrar en una

página WEB, siempre veras el código fuente de javascript y podrás tomar su código. Python, también es un lenguaje interpretado, pero siempre hay algo por ahí para cambiar esta limitación.

Ahora ya sabes cómo cuidar tu código fuente, podrás hacer tus juegos o programas y compartir el .EXE a tus amigos.

Ya hemos abordado bastante del tema, sabes que es un lenguaje y sabemos porque programamos en Python. Ahora bien, ese algoritmo que programamos, lo podemos encarar usando diferentes técnicas. Esas técnicas definen la manera que lo hacemos. Es ahora cuando reaparece la palabra paradigma

Algoritmos (parte 2)

Bueno, a esta altura ya sabemos que es un algoritmo. Si todavía no te quedo claro, te invito a volver a leer ¿Qué es un algoritmo? Por otro lado, hicimos una mira muy aérea de los tipos de lenguajes de programación. Pero, antes de comenzar con los lenguajes de programación, podemos practica con algo mas conocido. A ese lenguaje de prueba lo llamaremos seudocódigo.

Seudocódigo

El seudocódigo, es una manera de expresar los diferentes pasos de una solución, usando una forma mas conocida o amigable. Su función principal, es representar cada uno de los pasos que usarías para resolver un problema. Esta solución es simulada, pensando de una forma muy cercana al como lo harías en una computadora. La intención, es describir los pasos usando español de la forma mas detalla posible. Pero, tengamos en cuenta que el seudocódigo, es un escrito que se usa como entrenamiento y no sirve para implementarlo en una computadora.

Como se escribiría un seudocódigo:

Inicio

1. Acercarse a la puerta
2. Tomar el picaporte con la mano izquierda
3. Empujar el picaporte hacia abajo
4. Empujar la puerta hacia atrás, son golpearse con la puerta 😊

Fin

Técnicas algorítmicas

La algoritmia, es una técnica utilizada para resolver problemas mediante un conjunto de procesos mentales. Consiste en implementar pasos, que en general están probados y validades. A continuación, veremos algunas de esas técnicas.

enumeración exhaustiva

También conocida como fuerza bruta o búsqueda exhaustiva, consiste en probar cada opción posible de manera secuencial y exhaustiva. Ese método, evalúa si cada opción satisface las condiciones del problema planteado hasta encontrar la solución correcta.

Pasos para seguir:

1. *Definir las posibles opciones: Se identifican todas las combinaciones o elecciones que*

podrían formar una solución válida para el problema.

2. *Generar todas las combinaciones*
3. *Evaluar cada combinación*
4. *Encontrar la solución: Si se encuentra una solución válida, el algoritmo se detiene y presenta la solución. Caso contrario se repite.*

Veamos un ejemplo de esta aplicación:

Algoritmo: **Recorrer un laberinto hasta salir.**

Inicio

1- Repetir todo hasta encontrar la salida

a. Caminar hacia adelante mientras no encuentres una abertura para doblar a la izquierda.

i. Si encuentra una salida, girar a la izquierda

ii. Si se Encuentra al final de un camino y no puede avanzar, girar 90 grados

fin

Algoritmo de aproximación

Es una propuesta para subsanar el costo del método anterior, el objetivo es el mismo, pero tiene mejor rendimiento.

Este método, también conocido como algoritmo heurístico, en lugar de intentar encontrar una solución exacta proporciona una solución aproximada.

El algoritmo comienza con una solución inicial aleatoria y luego itera a través de un conjunto de vecinos de la solución actual. Para cada vecino, el algoritmo comprueba si es mejor que la solución actual. Si es así, el algoritmo reemplaza la solución actual con el vecino. El algoritmo continúa iterando hasta que se encuentre una solución satisfactoria.

Ejemplo algoritmo: **buscar ruta más corta**

INICIO

2- Repito hasta llegar al destino

3- Cuando llego a una esquina, selecciono la cuadra más corta

FIN

Veamos otro ejemplo, queremos calcular la raíz cuadrada de un número:

Inicio

1- Leer el número N

2- Leer un valor inicial aproximado A para la raíz cuadrada (puede ser simplemente N/2)

3- Definir una precisión deseada, epsilon (por ejemplo, 0.0001)

4- Mientras la diferencia absoluta entre A al cuadrado y N sea mayor que epsilon, repetir:

5- Calcular una nueva aproximación, A, como el promedio entre A y N/A (A = (A + N/A) / 2)

6- Mostrar el valor aproximado de la raíz cuadrada de N: A

Fin

Paradigmas de programación

Programación estructurada: También conocido como paradigma imperativo. Es un tipo de programación que se define mediante subrutinas y flujos de control

Programación Orientada a Objetos: En este paradigma se construyen modelos de objetos considerando un nivel de abstracción según el problema a resolver

Programación funcional: es un paradigma declarativo. En este paradigma las funciones tienen un rol protagónico, y podrán ser asignadas a variables y ser usadas como entradas y salidas de otras funciones

Ya hemos abordado bastante del tema, sabes que es un lenguaje y sabemos porque programamos en Python. Ahora bien, ese algoritmo que programamos, lo podemos encarar usando diferentes técnicas. Esas técnicas definen la manera que lo hacemos. Es ahora cuando reaparece la palabra paradigma

Para más información

te invito a ver la presentación de la clase

Encuentro 1

El video de la clase

La Bibliografía

- Python.org. (s.f.). [Página oficial de Python]. recuperado de Welcome to Python.org (2022)
- Varó, A. M., Sevilla, P. G., & Luengo, I. G. (2014). Introducción a la programación con Python 3. Recuperado de http://dx.doi.org/10.6035/Sapientia93
- Robledao, A. (2019). Qué es un seudocodigo. OpenWebinar. https://openwebinars.net/blog/que-es-pseudocodigo/ Recuperado en 2023