INTRODUCCIÓN

Subtemas:

- Sintaxis básica
- Cadenas
- Listas
- Diccionarios

Palabras clave: variables, declaraciones, expresiones, cadenas, subcadenas, while, for, listas, diccionarios.

Programación

- Una computadora es una máquina que puede ejecutar un programa.
- Con un programa la computadora puede realizar una tarea.
- El programa describe una serie precisa de pasos. Estos pasos indican la manera en que se debe ejecutar “automáticamente” el programa.

- Un lenguaje de programación es un lenguaje diseñado para producir programas que son escritos en lenguaje natural por los humanos y pueden ser interpretados por las computadoras.

- Phyton es un lenguaje de programación con syntaxis propia, que es interpretado y que se puede ejecutar en cualquier computadora que cuente con dicho lenguaje.

- Python, a diferencia de otros lenguajes toma en consideración el uso de espacios y tabulaciones en vez de los bloques mediantes llaves { }.

Expresiones

- Una expresión es algo que tiene un valor

Expresión -> Número
Expresión -> Expresión Operador Expresión
Expresión -> (Expresión)
Operador -> +
*Operador -> **
…
…
…

Ejemplo

- 01_expresiones.py: Este ejemplo muestra el uso de lagunas expresiones utilizadas en python.

Ejercicio

- Ahora que ya ha aprendido a usar las expresiones matemáticas, hacer un programa que despliegue el área de un triángulo, un círculo y un rectángulo escogiendo los valores que quiera.

Variables

- Una variable es un nombre que hace referencia a un valor.

- En Python se pueden usar letras, números y guiones bajos para nombrar variables.

PI = 3.1416
velocidad_de_la_luz = 300000

- El valor que guardan las variables puede variar durante la ejecución de los programas.

Ejemplo

- 02_variables.py: Este ejemplo muestra la creación y uso de variables.

Ejercicio

- Ahora que ya ha aprendido a definir variables, hacer el mismo programa del ejercicio anterior pero ahora declarando por ejemplo en caso del triángulo y el rectángulo la base y la altura y en el caso del circulo el radio. Checar que nos arroja el mismo resultado en ambos programas.

Cadenas

- Una cadena es una secuencia de caracteres, se puede utilizar comillas dobles o simples para crear una cadena.

  'Curso de Python'
  "Curso de Python"

- Por medio del operador + se puede hacer concatenación de cadenas.

- Para concaternar un número y una cadena se tiene que utilizar la función str().

Ejemplo

- 03_cadenas.py: En este ejemplo se muestra la declaración de cadenas y la combinación de éstas con números.

Ejercicio

- Ahora que ya sabe manejar cadenas tome su ejemplo anterior y agréguele formato, es decir título, área de la figura y su valor delante, además ponga el número del programa utilizando la función str().

Subcadenas

- En Python una manera fácil de subdividir cadenas es utilizando índices para obtener caracteres específicos de la cadena.

NOTA: I ***<string>[<expresión>]***

- También se puede obtener una sección de la cadena por medio de la especificación de un rango.

<string>[<inicio_expresión>:<fin_expresión>]

- *La subcadena que se obtiene va del inicio_expresión hasta un carácter antes de fin_expresión.*

- Si no se especifica la expresión final, se obtiene la cadena desde inicio_expresión y hasta el final de la cadena.

Ejemplo

- 04_subcadenas.py: Se muestran los ejemplos para extraer caracteres y subcadenas de una cadena.

Ejercicio

- Ahora que ya sabe el uso de subcadenas, escriba un programa que muestre su nombre completo, luego sólo su nombre de pila, después su apellido paterno, luego apellido materno y por último sus iniciales juntas.

Búsquedas en cadenas

- La función find(), sirve para buscar subcadenas y especifica la posición de inicio en la cadena, que es donde encontró la subcadena.

- Si la cadena no se encuentra, regresa el valor -1.

- También se puede especificar un índice a partir de donde se desea buscar dentro de la cadena.

cadena.find("<patrón>", índice)

Otras funciones en cadenas

- upper: Cambia todos los caracteres de la cadena a mayúsculas.

```
cadena.upper()
```

- lower: ulas.

```
cadena.lower()
```

- split: Descompone una cadena en una serie de elementos de acuerd

```
cadena.split()
```

- replace: se encarga de cambiar una cadena por otra

cadena.replace(“reemplazar”,”reemplazo”)

Ejemplo

- 05_busquedas.py: Este ejemplo muestra la combinación de buscar una cadena y generar una subcadena a partir de esa búsqueda. También contiene ejemplos de funciones que se pueden aplicar a las cadenas.

Ejercicio

- Ahora que ya sabe buscar cadenas vamos a hacer el proceso inverso con respecto al programa anterior, es decir ahora en vez de nosotros dar la posición de la cadena a imprimir, el programa nos debe de dar la posición.

Funciones o procedimientos

- Una función o procedimiento sirve para empaquetar código que sirve para ser reutilizado.

- Se puede usar ese mismo código con diferentes entradas y obtener resultados o comportamiento de acuerdo con esos datos.

```
def  <nombre>(<parametros>):
<bloque de código>
```

Ejemplo

- 06_funciones.py: Este ejemplo muestra el uso de funciones

Ejercicio

- Ahora que ya conoce como se utilizan las funciones realice un programa que haga lo mismo que el 2, es decir el de la obtención de áreas pero mediante funciones, además verá como declarar una función principal para que al momento que inicia el programa se vaya a ella y además usar la librería math.

Comparaciones

- Python tiene varias maneras de hacer comparaciones:

< menor que
> mayor que
<= menor que o igual a
>= mayor que o igual a == igual a
!= no igual a

Ejemplo

- 07_comparaciones.py: Aquí se muestra la manera de utilizar las comparaciones y el valor que regresan.

Ejercicio

- Ahora que ya sabe utilizar comparaciones hacer un programa que contenga la edad de varias personas y decir si es cierta o falsa la afirmación mediante una función, además use la implementación de la función main.

Estructuras Algorítmicas Selectivas

- Declaración IF (simple)

- La declaración IF sirve para ejecutar código dependiendo del resultado de una condición.

```
if <condición>:
   <bloque>
```

- Declaración IF-ELSE (compuesta)

- Este tipo de declaraciones se usan para dar una opción en el caso de que la condición no se cumpla.

```
if <condición>:
   <bloque>
else:
   <bloque>
```

- Declaración IF-ELIF-ELSE (compuesta)

- Este tipo de declaraciones sirve para generar varias casos de prueba. En otros lenguajes es similar a case o switch.

Estructura Algorítmica Repetitiva

- Ciclo WHILE

- Un ciclo es la manera de ejecutar una o varias acciones repetidamente. A diferencia del IF o IF-ELSE que sólo se ejecuta una vez.

```
while <condición>:
   <bloque>
```

- Para que el ciclo se ejecute, la condición siempre tiene que ser verdadera.

Ejemplo

- 08_if.py: Este ejemplo hace uso de if, if-else, if-else anidados.

- 09_while.py: Este ejemplo muestra como usar el ciclo while

Ejercicio

- Ahora que ya sabe usar el if y el else, hacer el mismo programa que el ejercicio anterior pero ahora si nos manda true escribir en consola verdadero y en caso contrario falso.

Ejercicio

- Ahora vamos a utilizar todo lo que sabe hasta ahora para hacer un menú que nos permita elegir que programa queremos ejecutar, 1 si queremos el de la obtención de áreas, 2 si queremos el de búsquedas, 3 si queremos el de comparaciones y 4 si queremos salir, por el momento probarlo moviendo las variables dentro del programa.

Listas

- Una lista está compuesta por cualquier cantidad y/o tipo de datos, ya sean cadenas, caracteres, números e inclusive otras listas.

```
diasDelMes = [31,28,31,30,31,30,31,31,30,31,30,31]
```

- Se puede acceder a las listas por medio de índices, estos índices comienzan desde 0 hasta el número de elementos menos 1.

Ejemplo

- 10_listas.py: Este ejemplo muestra el uso y manejo de las listas.

- 11_mutabilidad_y_alias.py: Este ejemplo muestra cómo hacer uso de la mutabilidad de las listas y el concepto de alias.

Ejercicio

- Ahora que ya sabe usar listas vamos a hacer una tabla de las tablas de multiplicar, es decir si pedimos 6 x 8 lo busque en una lista anidada y nos arroje el resultado.

Ejercicio

- Ahora que ya conoce como se maneja la mutabilidad y las referencias, hacer un ejercicio en el que se tenga una lista para actualizar datos de un usuario, además impleméntelo con una función para actualizar.

Operaciones en listas

- Existen operaciones ya definidas para hacer uso de ellas en las listas.

- Append: Esta función agrega nuevos elementos al final de la lista.

- Concat

<''lista''>.append(<''elemento''>)

- Length

<''lista_1''>+<''lista_2''>→<''lista_nueva''>

len(<''lista''>)→<''number''>

- Index: Encuentra la primera posición en la lista donde se encuentra un elemento.

<''lista''>.index(<''valor''>) → <''posición''> o error

- In: Sirve para indicar si un elemento se encuentra en una lista.

<''valor''> in <''lista''> → <''Boolean''>

- Not in: Se utiliza de la manera contraria a In, si no se encuentra en la lista regresa un valor verdadero.

<''valor''> not in <''lista''> → <''Boolean''>
not <''valor''> in <''lista''> → <''Boolean''>

- pop: Quita el último elemento de la lista

```
<'’lista_1''>.pop() → <'’lista_nueva”>
```

- del: Quita un elemento de la lista en el índice que se indique.

- insert: `del(<“lista[índice]”>)`).

```
lista.insert(índice, elemento)
```

Estructura Algorítmica Repetitiva

- Ciclo For

- La forma más conveniente de recorrer listas en Python es usando el ciclo for.

```
for <’’nombre''> in <''lista''>:
    <''bloque''>
```

- El ciclo recorre todos los elementos de la lista, asignando el elemento a “nombre” y evaluándolo dentro del bloque.

Ejemplo

- 12_funciones_listas.py: Este ejemplo muestra las funciones que se pueden usar en listas.

Ejercicio

- Ahora que ya sabe manejar las funciones de las listas hacer un programa que simule el registro de alumnos en una lista y al final la imprima en forma descendente.

Tuplas

- Las tuplas son como las listas, pero a diferencia de éstas, no son mutables.

- Se pueden aplicar las mismas operaciones que en as listas y su ventaja es que consumen menos memoria para almacenarse.

- Se crean, ya sea utilizando paréntesis o simplemente separando los valores por comas.

T = (1, 2, “hola”)
T = 1,2,”hola”

Ejemplo

- 13_tuplas: En este ejemplo se muestra la creación de tuplas y la obtención de sus elementos.

Ejercicio

- Ahora que ya sabe usar tuplas vamos a hacer el mismo ejercicio que en listas pero ahora con tuplas, una tabla de las tablas de multiplicar, es decir si pedimos 6 x 8 lo busque en una lista anidada y nos arroje el resultado, además diga cual es la diferencia.

Diccionarios

- Un diccionario se crea usando { } y consta de dos partes: llave y valor.

- Las llaves son inmutables, deben de tener un solo tipo de dato, una cadena o número. Una vez que es creado, no se puede cambiar su tipo.

- Mientras que el valor puede ser de cualquier tipo y se puede cambiar con el tiempo.

Ejemplo

- 14_diccionarios.py: Este ejemplo crea diccionario para agregarle conjunto de elementos y obtener los valores de cada elemento del diccionario.

Ejercicio

- Ahora que ya sabe usar diccionarios hacer un programa que maneje el precio y nombre de un inventario en una tienda de abarrotes.

Entrada de datos por consola

- Algunas veces necesitamos que el usuario introduzca datos al programa para que interactúe con él.

- La función raw_input() ayuda a leer del teclado una cadena introducida por el usuario.

raw_input('<frase_opcional>')

Ejemplo

- 15_lectura_de_datos: Este ejemplo muestra como leer datos del teclado.

Ejercicio

- Ahora que ya sabe capturar datos desde el teclado realice el ejercicio de las tablas de multiplicar pero pidiendo los datos y además ponga un menú.