Nivel 1 Introduccion a Programacion

Tomas Rodriguez - 202212868

January 2022

1 Descubriendo mundo de la programacion

Para programar, primero se tienen en cuenta unos cuantos pasos adicionales. Estos pasos son los que ayudan a un **programador** a solucionar los diversos problemas a los que se enfrenta.

1. **Analisis**: Esta parte consiste en **leer muy bien** un problema, con esto se puede especificar: que se quiere resolver, de donde partimos y a donde queremos llegar.

2. **Diseño**: En este paso se busca que el programador sea capaz de dividir el problema en subproblemas mucho mas pequeños y se halle la solucion a estos mismos por medio de una secuencia de pasos (*Algoritmo*).

3. **Construccion**: Finalmente, una vez hallada la solucion solo queda *implementar* el **algoritmo** por medio de un lenguaje de programacion y probar siempre su funcionamiento.

1.1 Algoritmos

Estos son en resumidas palabras, un conjunto de pasos a seguir para resolver un problema.
Estos a su vez, tienen una secuencia para desarrollarlos:

1. Expresarlos en lenguaje natural

2. Implementarlos en codigo

Figure 1: Algoritmos

2 Introduccion a la Programacion

Las computadoras son simples objetos sin instrucciones las cuales pueda ejecutar. De ahi nace la palabra de algoritmo. Un **algoritmo** es una secuencia de pasos a desarrollar para cumplir un objetivo en concreto y esto traducido a un lenguaje la cual la maquina pueda entender, se conoce como **programa**. El lenguaje es el medio y la herramienta de comunicacion y expresion mas conocida en nuestro mundo, por ende:
Existen dos tipos de lenguajes:

- Natural: Aquel que los seres humanos desarrollamos (Corporal, palabras, etc)
- Programacion: Son los lenguajes de maquina, es decir, el lenguaje que usa la maquina para ejecutar los algoritmos. Sin embargo, este es diseñado por humanos y no por las maquinas.

Los lenguajes poseen 4 grandes componentes:

1. Un alfabeto: Conjunto de simbolos para formar palabras (Ingles, Ruso, Japones).
2. Lexico: Conjunto de palabras que el lenguaje ofrece a sus usuarios.
3. Sintaxis: Conjunto de reglas que hacen que las posibles combinaciones de palabras tengan sentido.
4. Semantica: Lo mismo que la sintaxis solo que con frases enteras.

Debido a la necesidad de que los humanos nos comunicaramos con las maquinas, surgen una mezcla de lenguajes intermedios que:

- No son tan simples como el lenguaje de maquina
- No son tan complejos como el lenguaje natural

Surgen los **Lenguajes de Programacion de alto nivel**, los cuales permiten a los humanos dar instrucciones a las maquinas por medio de una combinacion de simbolos nueva que los humanos puedan entender. Estos tambien se conocen como **codigos fuente** y se guardan en **archivos fuente**.

2.1 Compilacion vs Interpretacion

Es el principio de convertir el codigo fuente en lenguaje de maquina para que la computadora pueda desarrollar cada una de las instrucciones descritas en dicho codigo. Ahora bien, este proceso afortunadamente lo realiza la computadora para que sea mas rapido y efeciente.

2.1.1 Compilacion

El codigo fuente se traduce una vez generando un archivo ejecutable o *.exe* con el cual se puede ejecutar el programa realizado. Sin embargo, cada vez que se genere un cambio en el codigo fuente, este proceso se tiene que repetir una y otra vez. Quien realiza esta accion se denomina **compilador** o **traductor**.

2.1.2 Interpretacion

Cualquier persona puede traducir el codigo cada vez que este se deba ejecutar, ya que la funcion de un **interprete** es interepretar el codigo cada vez que se ejecute. Sin embargo, el codigo no se puede distribuir libremente sin asegurarse que las otras personas tengan el interprete instalado en su sistema. Un iterprete realiza los siguentes pasos:

1. Lee el codigo de arriba a abajo y de derecha a izquierda
2. Verifica los 4 componentes del lenguaje
3. Si encuentra un error, finaliza la tarea dando la informacion de ese error al usuario
4. Termina ejecucion (Exito o Fracaso)

	Compilación	Interpretación
Ventajas	✓ la ejecución del código traducido suele ser más rápida; ✓ solo el usuario debe tener el compilador; el usuario final puede usar el código sin él; ✓ el código traducido se almacena usando lenguaje máquina; como es muy difícil de entender, es probable que tus propios inventos y trucos de programación sigan siendo tu secreto.	✓ puedes ejecutar el código tan pronto como lo completes; no hay fases adicionales de traducción; ✓ el código se almacena usando un lenguaje de programación, no un lenguaje máquina; esto significa que se puede ejecutar en computadoras que usan diferentes lenguajes máquina; no compila tu código por separado para cada arquitectura diferente.
Desventajas	✗ la compilación en sí puede ser un proceso que consume mucho tiempo; es posible que no puedas ejecutar su código inmediatamente después de realizar una modificación; ✗ debes tener tantos compiladores como plataformas de hardware donde desees que se ejecute tu código.	✗ no esperes que la interpretación acelere tu código a alta velocidad: tu código compartirá el poder de la computadora con el intérprete, por lo que no puede ser realmente rápido; ✗ tanto tu como el usuario final deben tener el intérprete para ejecutar tu código.

Figure 2: Ventajas y Desventajas de ambos modelos

A todo esto, es para que sepas que **Python** es un Lenguaje interpretado, es decir, utiliza el modelo del interprete y sus caracteristicas, ademas a este tipo de lenguajes de interprete se les conoce como **Lenguajes de Scripting**.

3 Python

Python es un lenguaje de programacion de alto nivel, interpretado y OOP con un uso generalizado con semantica dinamica. La seleccion de python se da poe los siguentes aspectos:

- Facil de aprender: Su tiempo de aprendizaje es mas corto que otros lenguajes de programacion.
- Facil de enseñar: No requiere de trucos exoticos, cosas xtrañas o cosas por el estilo, tan solo requiere tecnicas generales de programacion.
- Facil de utilizar: CCuando se implemneta software es mas facil hacerlo en python.
- Facil de entender: Ya que se asumeja mucho al ingles.

3.1 Competidores

1. Perl: Derivado de C
2. Ruby: Mas creativo

Python se encuentra entre estos dos tipos de lenguajes, respetando lo clasico pero a su vez siendo innovador.

3.2 Usos

Este se encuentra en una gran cantidad de aplicaciones, desde servidores de internet, almacenamiento en nube, redes sociales, etc. Sin embago, python no llega a todas las areas, unas de estas son:

- Programacion alto nivel: Aca el ideal o el rey es C
- Aplicaciones moviles

3.3 Tipos

Existen dos tipos de python:

- Python 2: Es un tipo estancado, con actualizaciones centradas en corregir errores mas no revolucionar el lenguaje.
- Python 3: La version mas actual y versatil de python

Estas dos versiones son incompatibles entre si. Ademas cabe resaltar que ambas versiones estas implementadas en C.

3.4 Mensajes de error

Estos se componen de 4 partes:

- TraceBack: La ruta que recorre el codigo al momento de la ejecucion.
- Ubicacion del error: Es el nombre del archivo, la linea y el nombre del modulo (Es engañoso ya que es cuando python detecta por primera vez el error, mas no significa que sea ahi).
- Nombre del error y explicacion de este.

4 Valores y Tipos de Datos

Los valores son aquellos los cuales son manipulados por los programas para hacer tareas especificas. Estos se pueden clasificar de acuerdo a su tipo:

- Enteros - **int**: Ocupan menos memoria y son mas rapidas sus operaciones.
- Decimales - **float**
- Strings o texto - **str**: Cadena o secuencia de caracteres de cualquier cosa.

Para conocer el tipo de valor que estas manejando, **python** tiene una funcion predefinida llamada *type* la cual indica que tipo de dato es un valor.

4.1 Enteros

Son aquellos numeros que no tienen partes fraccionarias. Hay distintos tipos de numeros en python:

- Binario: 0,1
- Octal:0oNumero
- Hexadecimal: 0xNumero
- N: Los naturales (Positivos y Negativos)

4.2 Flotantes

Son aquellos numeros los cuales si poseen parte decimal o fraccionaria.

- Fraccionarios: $\frac{Numero}{Numero}$ o $Numero.Numero$
- Notacion cientifica: Python permite el uso de e como la notacion cientifica de 10^N

4.3 Cadenas

Son el conjunto de caracteres de texto encerrados entre las comillas dobles o sencillas.

4.4 Booleanos

Son el True y el False o 1 y 0. Estos estan mejor definidos en al algebra booleana (Revisar Introduccion a la programacion)

5 Variables e instrucciones de asignacion

Las **variables** son una parte fundamental de un programa, estas en pocas palabras son *contenedores* en los cuales se pueden **almacenar** valores de distintos tipos, con el fin de usarlas mas adelante en el programa. Ahora bien, La instruccion de **asigancion** es aquella que permite guardar los valores en las variables y se representa por medio de un =. Ademas, hay que tener en cuenta que las varibales solo recuerdan el ultimo valor asignado, por ende, si se cambia, no se podra recuperar.
Por otra parte, una caracteristica interesante de **python** es su *tipado dinamico*, es decir, una variable puede cambiar entre tipos de datos, de int a str o float y viceversa.
Retomando lo anterior, las variables son reconocidas por su nombre, el cual actua como un **identificador** unico de cada variable. Sin embargo, estas tienen restricciones:

1. Pueden contener: letras, digitos, caracteres especiales

2. Los identificadores no pueden iniciar con un digito
3. Se distingue entre mayusculas y minusculas
4. No pueden coincidir con palabras reservadas

and	as	assert	break	class	continue
def	del	elif	else	except	exec
finally	for	from	global	if	import
in	is	lambda	nonlocal	not	or
pass	raise	return	try	while	with
yield	True	False	None		

Figure 3: Palabras Reservadas Python

5.1 Expresiones y Operadores

Las expresiones, son una combinacion entre las **variables**, **operadores** y **valores**. Estas expresiones se desarrollan con el fin de obtener un **valor** en concreto ya que su finalidad es ser almacenadas en variables para ser usadas mas adelante.

5.1.1 Operadores y Operandos aritmeticos

Los operadores son simbolos los cuales representan instrucciones o calculos como las operaciones aritmeticas.

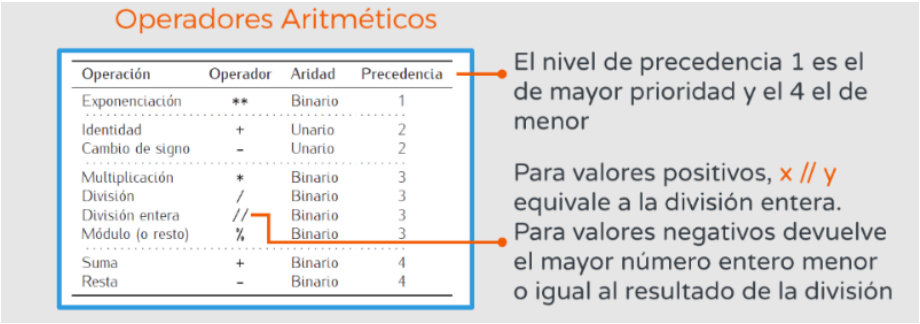

Figure 4: Operadores Aritmeticos

5.1.2 Operadores Strings

Dentro de los operadores, no solo existen operaciones sobre numeros (int, float) sino que tambien existen sobre los **strings**:

- **Suma (+)**: Este operador no actua como en numeros, sino que en este caso su funcion es **concatenar** o **pegar** dos strings.
- **Multiplicacion (*)**: Este operador cumple la funcion de **repeticion**, es decir, repite un string n veces el cual es el numero por el cual se repetira el string.

```
SumaStrings = "Hola " + "Como estas?"
MultiplicacionStrings = "Hola"*5
```

Listing 1: Operaciones sobre Strings

El resultado de esto sera:

```
Hola Como estas?
HolaHolaHolaHolaHola
```

Listing 2: Operaciones sobre Strings

5.2 Conversion de tipos

Los tipos de dato pueden ser convertidos de unos a otros, esto en python se hace por medio de las funciones:

1. int(): La cual convierte strings numericos("123") y floats a numeros enteros.
2. float(): La cual convierte strings flotantes("12.3") y ints a numeros flotantes o decimales.
3. str(): La cual convierte ints y floats a strings o cadenas de texto.

6 Programas

6.1 Funciones

Hasta el momento se han usado funciones basicas de python como lo son:

1. str()
2. int()
3. float()
4. type()

Estas ya son funciones predefinidas por el lenguaje, sin embargo hay muchisimas funciones mas.

- abs(): Valor absoluto.

- round(): Redondeo numeros a n decimales o a enteros.
- min(): Numero minimo.
- max(): Numero maximo.
- pow(): Elevar un numero a la potencia de n.

Asi mismo, como existen funciones numericas como ya las antes mencionadas, existen funciones sobre las cadenas de caracteres.

- ord(): Devuelve el valor numerico de un caracter en la tabla ASCII.
- chr(): Dado un valor numerico devuelve el caracter asociado segun ASCII.

Ahora bien, las funciones van mas alla de los tipos de dato, como por ejemplo.

- input(): Funcion de entrada que permite leer datos tecleados por el usuario (Consola).
- print(): Imprimir algo en pantalla (Consola).
- help(): Es una funcion dada otra funcion, nos explica que hace la funcion pasada como parametro.

Sin embargo, existen un sin fin de funciones que pueden:

- Estar integradas con python
- Provenir de modulos externos de python
- De tu codigo (Definicion de funciones)

Algo importante que resaltar sobre las funciones en cualquier lenguaje de programacion, es que cada una de estas debe tener un nombre **unico** y **descriptivo** con el fin de que sea mas sencillo conocer o recordar cual es su objetivo.

6.2 Argumentos

Estos hacen parte de una funcion, estos son ni mas ni menos, aquel valor o valores que necesita la funcion para producir un efecto o resultado, pueden existir funciones sin argumentos, sin embargo, es muy inusual considerando que casi todas las acciones requieren tener algo a la mano o un valor en dado caso.

```
1 print("Hola, Mundo!")
```

Listing 3: Tu primer Programa

Aca el argumento es: "Hola, Mundo!". Tambien cabe resaltar que las funciones simpre deben tener parentesis de inicio y cierre con el fin de encerrar los argumentos.

6.3 Invocacion

Este termino se refiere a ni mas ni menos que usar la funcion. Una funcion puede estar definida o puede ser parte del lenguaje pero para usarla, esta debe ser invocada, es decir, se debe escribir el nombre de la funcion ya definda con los argumentos que necesita.

6.4 Funcion Print

Un ejemplo de todo lo dicho anterior es la funcion print, que ademas tiene estos usos interesantes:

- Sin parametros: Es un salto de linea.
- $\backslash n$: Es un salto de linea inducido en el parametro.
- Las comas: Se usa para separar varios argumentos de la funcion.
- end = : Se refiere a con que debe termianr la impresion de los argumentos de la funcion, por default es $\backslash n$.
- sep = : Se refiere a como se deben separar los datos en la impresion.

6.5 Definicion de funciones

A parte de las funciones predefinidas por el lenguaje de programacion, el programador puede definir o crear nuevas funciones con el fin de enseñarle al lenguaje nuevas carateristicas que antes no era posible hacer.

```
def monedas(total):
    nikel = 5
    dime = 10
    total = total*100
    if(total%dime == 0):
        return total/dime
    else:
        numDimes = total//dime
        temp = total - (dime*numDimes)
        numNikel = temp//nikel
        return numDimes + numNikel

def rotar(x1,x2,x3):
    temp = x2
    x2 = x1
    x1 = x3
    x3 = temp
    return x1,x2,x3

def intereses():
    pesos = float(input('Ingrese su cantidad a invertir: '))
    i = float(input('Ingrese la tasa de interes a invertir: '))
    n = int(input('Ingrese numero de peridodos de la inversion: '))
    return round(pesos*(pow(1+(i/100),n)),2)
```

Listing 4: Ejemplos de funciones

Como se puede observar en los ejemplos, las funciones pueden ser **invocadas**, es decir, se pueden usar, pero tambien las componen ciertas partes.

- Signatura def: Es lo que se usa para definir una funcion.
- Nombre de la func ion: Nombre caracteristico que la diferencia del resto y va despues de la signatura *def*.
- Parametros o valores de entrada: Son los que van en los parentesis.
- Cuerpo de la funcion: Son todas las instrucciones necesarias a hacer para la funcion.
- Instruccion de retorno: el lo ultimo de la funcion, y tiene el papel de dar al usuario el resultado de la funcion dados unos parametros.

Cabe resaltar que:

1. Las funciones son independientes y diferentes
2. Un **argumento** es lo que se le pasa a la funcion cuando se **invoca** y un **parametro** es el que **hace parte de la funcion** y es necesario para el funcionamiento.
3. Cuando se piden mas de 2 valores a una funcion, es mala practica usar la funcion input con el fin de obtenerlos del teclado, por ende, deben ser parametros.
4. En las funciones, hay un tipo de variables llamadas **variables locales**, estas son aquellas que solo son usadas en una funcion y nadie mas las puede usar.

7 Estilo de programacion

Se recomienda hacer lo siguente cuando se programa:

1. Nombres de varibles y funciones distintivos
2. Documentacion de funciones: Descripcion, Parametros y Retorno
3. Siempre intentar usar instrucciones sencillas
4. Descomposicion de funciones, una funcion cumple una sola actividad
5. Comentarios de instrucciones

8 Modulos

Son colecciones de funciones y de valores que se encuentran en un archivo. Estos se pueden usar en diferentes programas con el fin de no tener que programar las instrucciones una y otra vez.
Finalmente, para hacer uso de estos conjuntos de funciones, se usa la palabra *import*.

9 Interfaz Usuario

Los programas deben ser:

- Seguros
- Buen desempeño
- facil de usar (Usabilidad)
- Escalable
- Tolerante a fallos
- Tolerable a cambios (Mantenibilidad)

La interfaz de usaurio es la forma de comunicacion entre el usuario y el programa desarrollado. Esta debe ser un modulo separado de la logica del programa, es decir no pueden estar en el mismo archivo. Ademas, asi como existe la funcion **print** con el fin de presentarle al usuario los resultados de algunas operaciones, tambien existen funciones que permiten al usuario interactuar directamente con el. La funcion mas conocida para la interaccion con el usuario por medio de consola es la funcion **input()**.