

1 Acercandose a la programación

1.1 Que es la programacion

La programación es una herramienta que sirve para que hacer cosas 'rápido', con la ayuda de una computadora o aparato que pueda realizar operaciones más rápido que una persona a mano.

Consiste en darle a la computadora o cualquier equipo (a partir de ahora voy a generalizar en 'computadora' cualquier aparato programable) una serie de instrucciones, que denominaremos **algoritmo**, que la compu deberá seguir, para obtener un resultado. Por ejemplo, cuando usamos un microondas y le indicamos que funcione durante 30 segundos, en el modo "descongelar", estamos programándolo con esas instrucciones para obtener como resultado la comida descongelada.

Este artículo se centra en la programación sobre una computadora, no un microondas.

¿Cómo darle a la computadora instrucciones? ¿Escribo 'Abrir Paint' en un navegador web y la computadora 'abre Paint'?

Para que la computadora entienda lo que estamos queriendo decirle o pedirle, se lo tenemos que comunicar. Para eso hay distintos **lenguajes de programación**, similar a lo que sería para nuestra comunicación el francés, inglés, castellano, afrikáans, etcétera.

De la misma manera, hay lenguajes más complejos que otros, más sencillos, más básicos, más 'hablados'.

Si bien yo personalmente utilizo C++, que es uno de los más usados junto con Java, últimamente el lenguaje Python se viene usando cada vez más, y es el que recomiendo para empezar y en base al cual escribiré este texto.

1.2 Conceptos básicos para empezar

Hay algunos conceptos con los que hay que familiarizarse antes de arrancar, que listo acá abajo con sus definiciones y ejemplos para empezar a tener los conceptos en la cabeza de a poco:

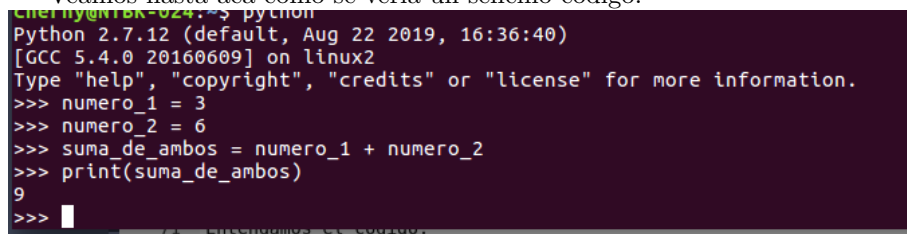
1. Código: Es lo que le decimos a la computadora, todo lo que vamos a escribir que luego la computadora va a tener que interpretar. Cada 'renglón' del código es una línea de código.
2. Plataforma: Es el lugar en el cual vamos a escribir el código. Es decir, no tiene mucho sentido, como decía antes, abrir Google Chrome o algún otro navegador como ese y escribir, en vez de 'www.google.com', líneas de código en el idioma Python. Entonces, ¿dónde va nuestro código?. Justamente, en estas plataformas. Hay plataformas online, es decir páginas web que entienden que vas a escribir en un cierto idioma y entonces la

computadora va a ejecutar lo que le pidas, y también hay plataformas offline, que no requieren de internet para funcionar.

3. Variable: Una variable es una porción de memoria en donde la computadora almacena información. Adónde está físicamente la memoria no importa, lo que importa de una variable es que tiene un **valor** guardado. Pero además del valor, tenemos que poder **acceder** a él. Entonces para esto, necesitamos que la variable tenga un **nombre**. El valor generalmente puede ser modificado, y podemos relacionar variables entre sí, por ejemplo sumando dos números. Por ejemplo, si yo le quiero decir a la compu que en un juego, un personaje empieza con un nivel de vida de 100, ese valor que empieza 100 tendremos que guardarlo en una variable, y a medida que avance el juego tenemos que poder restarle o sumarle algún número (a medida que el personaje reste o sume vida), y sobre todo tenemos que poder saber cuánta vida tiene el personaje en todo momento.
4. Tipos de variable: Las variables, estos espacios de memoria mencionados, pueden guardar distintos tipos de 'cosas':
 - (a) Pueden guardar números enteros (se las conoce como variables de tipo entero)
 - (b) palabras (se las conoce como strings)
 - (c) caracteres (uno solo por variable, no como una string que puede tener varios; se los conoce como char)
 - (d) 'booleanos' (boolean en inglés, son las llamadas variables 'de verdad', contienen el valor "verdadero" o "falso"); estos últimos pueden usarse para guardar en una variable por ejemplo si 'llovió en el día de hoy'. Es básicamente la respuesta a una pregunta de sí/no.

Hay otros tipos de variables pero estos son los básicos.

Veamos hasta acá cómo se vería un sencillo código:

A screenshot of a terminal window with a dark background and light-colored text. The prompt is 'chen@hyattuk-024:~\$ python'. The output shows the Python version 'Python 2.7.12 (default, Aug 22 2019, 16:36:40)' and the compiler '[GCC 5.4.0 20160609] on linux2'. It also displays instructions to type 'help', 'copyright', 'credits' or 'license' for more information. The code being executed is: '>>> numero_1 = 3', '>>> numero_2 = 6', '>>> suma_de_ambos = numero_1 + numero_2', and '>>> print(suma_de_ambos)'. The output of the print statement is '9'. The prompt '>>>' is followed by a cursor. At the bottom, there is a line of text that is partially obscured: '71 Entendamos el código.'

En esta imagen, además del código, se puede ver que el código debemos escribirlo en un lugar en particular, en donde la computadora entiende que 'bueno, acá me van a dar instrucciones que yo voy a tener que entender en el idioma Python, y voy a tener que hacer lo que quieren que haga'.

Entendamos el código:

```
numero_1 = 3
numero_2 = 6
suma_de_ambos = numero_1 + numero_2
print(suma_de_ambos)
```

Qué hace este código:

1. Primero, se guarda un pedazo de memoria con el nombre 'numero_1' y de valor 3
2. Luego, se guarda otro pedazo de memoria de nombre 'numero_2' y de valor 6
3. Se genera un pedazo de memoria de nombre 'suma_de_ambos', cuyo valor es la suma del valor que haya en la variable 'numero_1' (que es 3), y el valor de la variable 'numero_2' (que es 6). Entonces, en la memoria se guarda un número 9.
4. La última línea, simplemente nos muestra en la pantalla el valor que hay en el pedazo de memoria llamado 'suma_de_ambos', que como vimos es 9.

Acá aconsejo ir a la plataforma <https://repl.it/languages/python3> y en la parte negra de la página, escribir el código de arriba. Se puede jugar un poco también generando otra variables, editándolas (es decir, después de hacer 'numero_1 = 3', hacer 'numero_1 = 11' por ejemplo, y ver qué pasa con suma_de_ambos), etc.

Una vez que se entiende que lo que haremos será, en un lugar en particular, escribir acciones para que la computadora entienda y ejecuta, continuemos viendo algunos otros conceptos:

1. Funciones: Una función es simplemente un pedazo de código que se puede reusar. Uno de los objetivos de las funciones es que no hay que repetir código; si querés decirle a la compu que haga varias veces lo mismo, a ese 'lo mismo' le ponés un nombre, y le decís 'hacé eso', 'hacé eso de vuelta', etc. Por ejemplo, una función podría ser algo que nos diga cuánto vale $1+2+3+4$. Cada vez que queramos saber el valor de esa suma, llamamos a la función en una línea, y en nuestro código simplemente escribiríamos por ejemplo "sumarPrimerosNumeros()". En este caso, sumarPrimerosNumeros es el nombre de nuestra función, y para que la computadora ejecute el código de la función debemos terminar con '()'. Más adelante veremos otra aplicación súper interesante de las funciones.
2. Algoritmo: Como ya dije antes, un algoritmo es una serie de pasos a seguir.
3. Correr un código: Decirle a la compu que ejecuta el algoritmo que hicimos.

Pensemos ahora en un algoritmo típico, familiar: una receta de una comida. Eso es exactamente un algoritmo:

1. romper los huevos en un bowl;

2. poner harina con los huevos;
3. mezclar por 5 minutos;
4. prender el horno;
5. meter el bowl en el horno;
6. si la harina es blanca dejar el bowl media hora, si es harina negra dejarlo 20 minutos;
7. retirar el bowl del horno;
8. dejar enfriar.
9. Listo, postre terminado.

Más allá de que esta receta no parezca muy rica, no vamos a hablar del postre en este caso sino de los pasos a seguir, así que con eso nos sirve para seguir avanzando.

Veamos entonces este algoritmo, e intentemos reconocer qué características tiene una receta. Sugiero intentar pensar características de una receta como la que puse más arriba antes de seguir leyendo.

Una de las características, es el **orden**. Esto es sumamente importante. Si primero pusiéramos el bowl en el horno, lo sacáremos, y después pusiéramos el huevo, no tendría sentido. Por eso decimos que los algoritmos son **secuenciales**. Otra de las características es la necesidad de tener los **materiales**. Sin huevo, harina, bowl, no podríamos seguir estos pasos. En programación, esto se conoce como 'datos de entrada', o **input**, en inglés. Es decir, si nuestro algoritmo va a ejecutar una serie de acciones, tenemos que saber en qué situación estamos, y qué materiales tenemos. No voy a obtener el mismo postre si la harina es negra o blanca, por ejemplo.

Otra, quizás un poco más implícita, es la creación del postre, la finalidad del algoritmo, el **objetivo**. Esto lo nombraremos 'respuesta' de nuestro algoritmo, o **output** en inglés.

Otra característica de muchos algoritmos, que puse en este ejemplo a propósito, es la **toma de decisiones**. Como se ve en el paso 6, depende de la harina que tengamos, lo que tendremos que hacer.

No se me ocurrieron más características del ejemplo de la receta, pero podría haber más. Si se les ocurrieron más cosas, buenísimo.

1.3 Un poco más de diversión: toma de decisiones y repeticiones de código

Al principio decíamos que vamos a usar la programación para hacer cosas 'de manera rápida'. Entonces, podemos pensar ahora en algo que generalmente sentimos que nos lleva mucho tiempo, pensar cómo traducirla a Python, escribirla a la compu, y pedirle que la resuelva.

Un ejemplo que voy a traer ahora es el de saber en qué día cae mi cumpleaños.

Me parece algo típico, que por ejemplo estamos en un miércoles 3 de mayo, cumple años un amigo nuestro, y pensamos 'uh qué día caerá mi cumpleaños, viernes, sábado? Ojalá caiga sábado'. Obvio que podemos ver el calendario y listo, pero bueno, quizás queremos saber qué día va a caer nuestro cumpleaños número 80, por decir algo, y no es tan fácil de googlear.

Entonces pensemos por un rato en este problema: **Es miércoles 3 de mayo, y queremos saber qué día cae por ejemplo el 26 de julio.**

¿Cuál es la información que tenemos al principio, que sería nuestro input? Sabemos que el 3 de mayo de miércoles, y queremos averiguar el día que cae el 26 de julio (es decir, para empezar necesitamos saber dónde estamos, y hasta donde queremos ir, si no no podemos ni empezar a hacer algo), ¿Cuál es la información que queremos al final, es decir nuestro output? Qué día cae el 26 de julio de ese mismo año. Es decir, la respuesta del programa será alguno de 'Lunes', 'Martes', etcétera.

Ahora sí, a resolverlo! Algo que siempre necesitamos saber hacer, es saber resolver el problema que se nos presenta, de manera manual. Es decir, si yo no sé cómo saber a mano, por más que me lleve mucho tiempo, cómo calcular qué día de la semana va a caer una fecha, nunca le voy a poder decir a la computadora lo que tiene que hacer para resolver la cuestión.

Entonces, cómo haríamos a mano para resolverlo? Bueno, puede haber varias maneras: ir sumando de a 7 para que el día de la semana no cambie, ir sumando de a 1... y podría haber otras. Pensemos en la solución de ir sumando de a un día.

¿Qué hacemos? Sumamos uno a la fecha. Si es el último día del mes, arrancamos desde el 1, y vamos al siguiente mes. Después de esto, decimos 'estamos en el siguiente día', y si era lunes ahora estamos en un martes, si era martes estamos en miércoles, etc. ¿Hasta cuándo sumamos 1? Bueno, hasta que llegamos a la fecha buscada, en nuestro ejemplo el 26 de Julio.

Para esto, vamos a necesitar que la computadora **tome decisiones**. La decisión por ejemplo de ver si tenemos que cambiar de mes o no. También al decisión de 'cuál es el día siguiente', es decir, SI ES LUNES, pasa a martes, pero SI ES SABADO, pasa a domingo. Estos 'SI PASA TAL COSA', son decisiones que la compu tiene que poder evaluar. Tiene que ver con los booleanos que mencionábamos antes.

Veamos un pseudocódigo para resolver este problema (pseudocódigo es como un código pero no formal, no está en el lenguaje Python ni en otro lenguaje, sino que va en castellano para que sea entendible para personas):

```
Algoritmo Que_dia_cumple_anios (INPUT: Dia_de_hoy, Numero_de_hoy,
    Numero_de_mes_de_hoy, Numero_de_mi_cumple,
    Numero_de_mes_de_mi_cumple)
Numero_de_dia_actual = Numero_de_hoy
Mes_actual = Numero_de_mes_de_hoy
Dia_actual = Dia_de_hoy
Mientras el Numero_de_dia_actual y el Mes_actual no coindidan
    con Numero_de_mi_cumple y Numero_de_mes_de_mi_cumple, REPITO:
```

```

SUMAR 1 a Numero_de_dia_actual
SI Numero_de_dia_actual es el ultimo dia de Mes_actual:
    Numero_de_dia_actual = 1
    SUMAR 1 a Mes_actual
Dia_actual PASA A SER EL DIA SIGUIENTE (es decir, si era
    lunes va a ser martes, si era jueves va a ser viernes, y
    asi)
el RESULTADO es el valor de Dia_actual

```

Ahora, entendamos el código:

Recibimos 5 variables, como habíamos mencionado. Al principio del algoritmo, creo 3 variables más, que es donde voy a ir guardando los días por los que voy pasando. Al final del algoritmo, asumimos que llegué al día de mi cumpleaños, porque lo que REPITO lo repito hasta que llegue a mi cumple; cuando llego a mi cumple, no muevo más el Dia_actual.

Tómense un tiempo acá para releer el pseudocódigo, y entender todo paso a paso.

Una vez comprendido ese pseudocódigo y por qué está cada línea (todas las líneas son cruciales y tienen un propósito), veamos el código, en lenguaje Python. Hay cosas que verán por primera vez en este código. Abajo está la explicación, pero como hay cosas intuitivas, empiezo con el código a secas:

```

dia_de_hoy = 'Miercoles'
numero_de_hoy = 3
numero_de_mes_de_hoy = 5
numero_de_mi_cumple = 26
numero_de_mes_de_mi_cumple = 7
# Hasta aca tengo el input, ahora hago lo que hago en el pseudocodigo
  de arriba
numero_de_dia_actual = numero_de_hoy
mes_actual = numero_de_mes_de_hoy
dia_actual = dia_de_hoy
while (numero_de_dia_actual, mes_actual) != (numero_de_mi_cumple,
    numero_de_mes_de_mi_cumple):
    numero_de_dia_actual = numero_de_dia_actual + 1
    if numero_de_dia_actual == 32:
        if mes_actual == 1 or mes_actual == 3 or mes_actual == 5 or
            mes_actual == 7 or mes_actual == 8 or mes_actual == 10
            or mes_actual == 12:
                numero_de_dia_actual = 1
                mes_actual = mes_actual + 1
    if numero_de_dia_actual == 31:
        if mes_actual == 4 or mes_actual == 6 or mes_actual == 9 or
            mes_actual == 11:
                numero_de_dia_actual = 1
                mes_actual = mes_actual + 1
    if numero_de_dia_actual == 29:
        if mes_actual == 2:
            numero_de_dia_actual = 1

```

```

        mes_actual = mes_actual + 1
    if dia_actual == 'Lunes':
        dia_actual = 'Martes'
    elif dia_actual == 'Martes':
        dia_actual = 'Miercoles'
    elif dia_actual == 'Miercoles':
        dia_actual = 'Jueves'
    elif dia_actual == 'Jueves':
        dia_actual = 'Viernes'
    elif dia_actual == 'Viernes':
        dia_actual = 'Sabado'
    elif dia_actual == 'Sabado':
        dia_actual = 'Domingo'
    else:
        dia_actual = 'Lunes'
print("El dia de tu cumpleaños cae")
print(dia_actual)

```

1.4 Análisis del código

Las primeras líneas simplemente crean variables que son lo que llamamos el input. La siguiente línea, que empieza con el caracter `''`, es lo que se llama un *comentario*. Es una línea que al correr el código, no se ejecuta. Está bueno para aclarar cosas, por si escribimos código que le vamos a pasar a otra persona. En vez de explicarle todo paso a paso, comentamos entre líneas lo que sentimos que necesitamos aclarar.

Las siguientes tres líneas simplemente nombro las variables que van a ir recorriendo todos los días de a uno.

A partir de la línea del **while** se pone más complejo: 'While' significa 'mientras' en inglés. Entonces lo que hace esto es evaluar una condición, que es lo que escribimos después de la palabra `while`, y mientras esa condición se cumpla, 'entra' al ciclo que está abajo (es decir, lo que está corrido un poco más a la derecha), y hace una repetición más.

Condición del while: La condición del `while` chequea que el número y el mes del día actual no coincidan con el número y mes de mi cumpleaños. El término `!=` significa **es distinto**. El hecho de poner a las variables entre paréntesis significa que va a comparar lo de la izquierda y lo de la derecha del `!=` como vectores. Entonces, si coinciden en TODAS las posiciones, son iguales. Si no, son distintos. Entonces, como para nosotros haber llegado al día de mi cumpleaños es llegar al mismo número de día y mismo número de mes, deben coincidir ambos números. Mientras no coincidían ambos, sigo sumando de a uno.

Como dijimos al principio, nuestra solución se trata de ir sumando uno al día actual. Entonces, al 'entrar al while', como todavía no llegamos al día de mi cumpleaños sumo uno. Y acá es donde tengo que tener cuidado de ver si debo cambiar de mes o no. Para eso armo los **IF**.

'If' significa 'si' en inglés, y es lo que antes decíamos de tomar decisiones. Lo que hace la computadora es 'Si pasa la condición, entonces entrá a las líneas de código de adentro del if'. Acá de vuelta, las líneas 'de adentro del if' son las que están inmediatamente abajo, y corridas con una sangría más hacia la derecha. Entonces, si después de sumar uno al día actual, estamos en el día 32 de un mes de 31 días, debemos cambiar. Eso es lo que verifican los primeros dos 'ifs'. El primer if se fija que el número del día actual sea 32 (para comparar y preguntar si dos cosas son iguales, se usa `==`). Pero atención al segundo if: Acá, queremos preguntar si el mes es enero, o marzo, o mayo, o cualquiera que tenga 31 días. Para eso usamos el **or**. El 'or' nos da la posibilidad de preguntar 'si esta condición se cumple, o esta, o esta, ...'. Entonces, si el número de mes actual es 1, o si el número de mes actual es 3, o si el número de mes actual es 5, etcétera, entramos 'adentro' del if, en cuyo caso hacemos que el número de día actual sea 1, y le sumamos uno al mes.

Si se entendieron estos dos primeros ifs (que espero que sí), los siguientes son similares pero con los demás meses. Si no entendieron, pueden buscar bibliografía sobre la **sintaxis** del if.

Una vez que sabemos que estamos en un día válido (es decir, no estamos en el 32 de mayo), pasamos a mover el día de la semana del día actual. Para esto, se introducen palabras nuevas que son **elif y else**. El **else** se usa, luego de un if, para decir 'si no se cumple la condición del if, ejecutá el siguiente código'.

El **elif** es una mezcla entre un 'else' y un 'if'. Es decir, el else va sin condición, porque es simplemente 'si la condición de arriba no se cumplió'. En cambio, al escribir 'elif', estamos diciendo 'si no pasa la condición de arriba, pero se cumple la condición siguiente'. Y por ende necesita una condición en esta línea.

Entonces, básicamente, todas esas líneas de if, elif y else dicen: Si el día actual es Lunes *compara con el* `TT`, entonces hacé que la variable `dia_actual` *es decir, modifica la variable* `dia_actual` tenga el valor 'Martes'. Si no es Lunes, pero el día es Martes, ponle el valor Miércoles. Si TAMPOCO es Martes, ponle Miércoles. Si tampoco es miércoles, ponle jueves. Y así sigue, hasta que al final, si el día no es ninguno del Lunes a Sábado, sabemos que es Domingo porque no puede ser otra cosa. Entonces, si bien el último 'else' podría ser *elif* `dia_actual == 'Domingo'`, por lo que dijimos no es necesario, y con el else estamos bien.

Toooodo esto, para sumar uno a la fecha (numérica), y al día de la semana. Entonces, así va a seguir ejecutándose nuestro código, hasta que llegamos al día de mi cumpleaños. Ahí, salimos del código de adentro de while, y se ejecuta las últimas dos líneas que imprimen simplemente. En una línea se imprime 'El día de tu cumpleaños cae', y en otra línea imprime el día de la semana que corresponda al 26 de julio, que en este caso será Miércoles.

Se puede copiar este código en la plataforma antes mencionada y pegar en la parte negra, y ver que efectivamente se muestre lo que acá digo que se muestra. También podemos cambiar el día del 3 de mayo, el día de la semana, o el día de nuestro cumpleaños.

1.4.1 Preguntas para tener en cuenta

¿Qué pasa si la fecha que conozco es un día en enero, mi cumpleaños es en abril, y el año es bisiesto? ¿Sigue valiendo el mismo código? ¿Y si la fecha que conocemos es el 15 de diciembre, y nuestro cumpleaños es el 6 de enero (hay un cambio de año al sumar de a uno)? ¿Y si le pifio al input y pongo que el número del día de hoy es 33? ¿Y si en el día de hoy escribo 'miercoles' con 'm' minúscula?

Bueno, lo copado de programar es que todas estas situaciones las podemos realmente probar. Es decir, podemos poner que el número del día es 33 y decirle a la compu 'tomá correte esto' y ver qué pasa.