Programación 1

Tecnicatura Universitaria en Inteligencia Artificial

2022

Apunte 1

1. Introducción

La **Informática** nace bajo la idea de ayudar al hombre en aquellos cálculos rutinarios, donde frecuentemente existe una repetición de tareas. Ejemplo de ello es la gestión de un censo, tal es el caso del primero automatizado en 1890 en EEUU utilizando tarjetas perforadas. En esa época se pensó que una máquina no sufriría cansancio ni cometería errores. Con el tiempo esta idea se fue afianzando en la sociedad, y es hoy día que se sigue trabajando para mejorar las prestaciones de los sistemas de cómputo.

Si repasamos la historia veremos que la Informática estuvo entre nosotros desde tiempos lejanos. A continuación transcribimos algunas ideas de importantes pensadores relacionados con la Informática. Para ubicarse en la historia, debe notarse que los primeros modelos de computadoras aparecieron alrededor del año 1940.

Gottfried Wilhelm Leibniz (1646-1716) fue uno de los grandes pensadores de los siglos XVII y XVIII, decía que *"Es una pérdida de tiempo que personas cualificadas pierdan horas como esclavos en la labor de calcular, lo que podría ser delegado en cualquier otro si se pudieran utilizar máquinas"*. Basándose en los trabajos de Pascal construyó una calculadora compuesta por cilindros dentados con longitud incremental que podía sumar, restar, multiplicar y dividir automáticamente, conocida como la rueda de Leibniz. Algunas máquinas basadas en estos mismos principios han jugado un papel preponderante en la Segunda Guerra Mundial, e incluso algunas están todavía en uso. Leibniz fue también el primer pensador occidental en investigar la aritmética binaria y sentó las bases de la lógica simbólica, basada en símbolos y variables abstrayendo la semántica de las proposiciones.

Ada Byron (1815-1852) desarrolló los primeros programas para la Máquina Analítica de Babbage (1833), debido a estos trabajos, se la considera el primer programador de computadoras del mundo. Ella escribió *"La Máquina Analítica no tiene la pretensión de crear nada. Puede realizar cualquier cosa siempre que conozcamos cómo llevarla a cabo. Puede seguir análisis; pero es incapaz de descubrir relaciones analíticas o verdades. Su potencialidad es la de ayudarnos a hacer posible aquello sobre lo que tenemos un conocimiento previo."* Muchas han sido las mujeres que han realizado grandes aportes a la Informática, aún así Ada Byron es la única mujer que cuenta con un lenguaje de programación que lleva su nombre: en 1979 el departamento de Defensa de los Estados Unidos creó un lenguaje de programación basado en Pascal que se llamó lenguaje de programación Ada en su honor.

Herman Hollerith (1860-1929) fue un estadístico estadounidense que inventó la máquina tabuladora. Es considerado como el primer informático, es decir, el primero que logra el tratamiento automático de la

información (Informática = Información + automática). En esos tiempos, los censos se realizaban de forma manual, con el retraso de unos 10 años en su procesamiento. Ante esta situación, Hollerith comenzó a trabajar en el diseño de una máquina tabuladora o censadora, basada en tarjetas perforadas que patentó en el año 1889. Un año después incluyó la operación de sumar con el fin de utilizarla en procesos de contabilidad.

Alan Turing (1912-1954) fue un matemático, lógico, científico de la computación, criptógrafo y filósofo británico. Es considerado uno de los padres de la ciencia de la computación siendo el precursor de la informática moderna. Proporcionó una influyente formalización de los conceptos de algoritmo y computación, la famosa máquina de Turing. Durante la Segunda Guerra Mundial, trabajó en descifrar los códigos nazis, particularmente los de la máquina Enigma. Tras la guerra diseñó una de las primeras computadoras electrónicas programables digitales en el Laboratorio Nacional de Física del Reino Unido (1945). Entre otras muchas cosas, también contribuyó de forma particular e incluso provocativa al enigma de si las máquinas pueden pensar, es decir a la Inteligencia Artificial. Turing decia: *"Las máquinas me sorprenden con mucha frecuencia."*

John Von Neumann (1903-1957) fue un matemático húngaro que realizó contribuciones fundamentales en física cuántica, análisis funcional, teoría de conjuntos, ciencias de la computación, economía, análisis numérico, cibernética, hidrodinámica, estadística y muchos otros campos. Está considerado como uno de los más importantes matemáticos de la historia moderna. Diseñó una arquitectura de computadoras que lleva su nombre, y aún es utilizada en casi todas las computadoras personales, microcomputadoras, minicomputadoras y supercomputadoras. Von Neumann decía *"Podría parecer que hemos llegado al límite de lo que es posible lograr con la tecnología informática, aunque hay que tener cuidado con tales declaraciones, ya que tienden a sonar bastante tontas en cinco años."*

Todos estos trabajos constituyen antecedentes de la informática. El término Informática nace recién en la década de 1960 en Francia bajo la denominación **INFORMATIQUE**, debido a la contracción de las palabras **INFOR**mation y auto**MATIQUE**, es decir el tratamiento de la información por medios automáticos. Es la ciencia que estudia el tratamiento automático y racional de la información. Se habla de tratamiento automático debido a que son máquinas las que procesan la información y se dice racional por estar los procesos definidos a través de programas que siguen el razonamiento humano.

2. El modelo computacional

Para encarar la resolución de un problema por computadora hay que considerar como punto de partida que el procesamiento de la información lleva en sí el hecho de tomar datos, elaborarlos y emitir resultados de acuerdo a dichos datos y a la lógica del proceso que se está ejecutando en ese momento. Por lo tanto, se puede pensar en un modelo computacional compuesto de tres partes: **ENTRADA**, **PROCESO** y **SALIDA**, como se muestra en la Figura 1.

Figura 1: El modelo computacional

Una vez entendido el problema hay que desentrañar del enunciado los datos necesarios para comenzar el proceso de resolución, abstrayéndose del proceso de resolución. Habría que preguntarse *"¿ Qué necesito para resolver el problema, si yo fuese el procesador?" "¿ Qué datos necesito que me de otra persona para poder obtener la solución del problema?"*, y anotarlos como **ENTRADA o DATOS**.Una vez determinados los datos, se deberá determinar cuál es el/los resultado/s, es decir, qué se obtiene con los datos de entrada y anotarlos como **SALIDA o RESULTADOS**.

A veces pueden ser más fáciles de visualizar los resultados antes que los datos de entrada, en dicho caso habría que preguntarse *"¿Para obtener este resultado, qué información necesito?"*. Luego se preguntará *"¿ Cuáles son los pasos para llegar al resultado partiendo de los datos?"*, con lo cual se podrá escribir el **PROCESO o ALGORITMO** asociado. Por ejemplo, si se desea realizar un algoritmo para calcular el área y el perímetro de un círculo en función de su radio, las diferentes partes del modelo computacional serán:

ENTRADA:	radio (R)
PROCESO:	Área: $\Pi * R^2$, Perímetro: $2 * \Pi * R$
SALIDA:	Área y Perímetro

El proceso (algoritmo) asociado a este problema será:

- Dar el valor del radio y guardar ese valor con el nombre R
- Calcular el área como $\Pi * R^2$
- Calcular el perímetro como $2 * \Pi * R$
- Informar los resultados de área y perímetro

Sin embargo, el proceso (algoritmo) expresado en forma coloquial no puede ser entendido por la computadora. Por lo cual habrá que escribirlo siguiendo reglas especiales que la computadora entienda. Vale la pena observar que la relación entre entrada, proceso y salida no es algo que necesariamente se realice en este orden: primero ingresar todos los datos, luego procesarlos, y por último exhibir todos los resultados. P. ej. un algoritmo puede solicitar la entrada de algunos datos, obtener ciertos resultados, y en base a esto pedir nuevos datos y mostrar otros resultados sobre la marcha del proceso.

3. Datos e Información

Por último, describiremos la diferencia que existe entre Datos e Información, términos que se pueden pensar como sinónimos.

En el ambiente de la Informática, el término información puede aplicarse a casi cualquier cosa que pueda comunicarse, tenga o no valor.

Los datos, en principio, son información no elaborada, que una vez procesados (comparados, ordenados, sumados, etc.) constituyen información útil.

Según esto, las palabras, los números, las imágenes de este libro son símbolos que representan datos. Si subrayase una frase, se añadirá información a la página.

En una dada situación problemática se puede disponer de muchos datos, de los cuales sólo algunos pueden ser útiles para la resolución del problema, es decir, considerados información útil.

Entonces podemos decir que la información es el conocimiento producido como resultado del procesamiento de los datos.

Otro término que se utiliza es Tratamiento de la información, referido al conjunto de operaciones que se

realizan sobre una dada información. Esto es *"lectura de datos"*, *"preservar datos"*, *"comparar datos"*, *"procesar aritméticamente datos"*, *"presentar resultados"*. Una adecuada combinación de estas operaciones lleva a resolver los problemas. En el subcapítulo siguiente comenzaremos a formalizar lo vertido en esta sección.

4. ¿Qué es un algoritmo?

El término algoritmo deriva del nombre del matemático persa Abu Ja'far Mohammed ibn Musa Al-Khwarizm (Mohammed, padre de Ja'far, hijo de Moisés, nacido en Khowarizm), el cual vivió alrededor del año 825 después de Cristo. Al-Khowarizm escribió tratados de Aritmética y Álgebra. La acepción original fue algorism y hacía referencia al proceso de prueba de cálculos realizados utilizando números arábigos, tema central del libro de Al-Khowarizm.

Un **ALGORITMO** es una forma de describir la solución de un problema, explicando paso a paso como se debe proceder para llegar a una respuesta encuadrada a los datos disponibles, en un tiempo finito.

Seguramente existirán distintos algoritmos para resolver un mismo problema y todos serán correctos, cada uno de ellos pensado por una persona diferente, cada uno de ellos con un diseño distinto, cada uno de ellos con un sello propio. Escribir algoritmos es un arte, en el que cada persona le dará su estilo personal. Las características de un algoritmo son:

- Un algoritmo debe ser **preciso**, es decir, debe indicar claramente, sin ambiguedades, cada uno de los pasos a seguir para conseguir el objetivo propuesto.
- Un algoritmo debe estar **exacto**, es decir, que si se sigue el algoritmo varias veces con el mismo juego de datos, los resultados obtenidos deben ser los mismos.
- Un algoritmo debe ser **finito**, de tiempo finito, su ejecución debe concluir en algún momento.

¿Quién "ejecuta" estas acciones? La respuesta es: un procesador.

Un procesador es aquel sujeto o máquina que puede entender un enunciado y ejecutar el trabajo indicado en el mismo.

La sintaxis y semántica a utilizar para escribir un algoritmo depende del lenguaje que conoce el procesador, pues según cuál sea éste, habrá que describir las tareas con más o menos detalle.

Entonces, frente a un problema, sabiendo quien será el procesador y su lenguaje de comunicación, podremos describir la solución del problema mediante un algoritmo que pueda ser entendido y ejecutado sin ambigüedades.

5. Programa

Definiremos qué se entiende por el término programa.

Un **programa** es un conjunto de acciones que puede entender y ejecutar una computadora.

Otra definición podría ser: "es un algoritmo traducido a algún lenguaje que pueda ser entendido por una computadora para poder ejecutarlo". Cada acción del programa se denomina instrucción.

Una **instrucción** es una combinación de palabras y símbolos que obedeciendo a la sintaxis propia de un lenguaje, son interpretados y utilizados por el computador para realizar una determinada acción.

Para llegar a hacer un programa es necesario el diseño previo del algoritmo. Esto es, representar la solución del problema en un lenguaje natural, en el cual se expresan las ideas diariamente.

Un **lenguaje de programación** es un conjunto de símbolos (sintaxis) y reglas (semántica) que permite expresar programas.

Los algoritmos deberían ser independientes tanto del lenguaje de programación en que se expresa el programa, como de la computadora que lo va a ejecutar.

6. Lenguajes de Programación

Los **lenguajes de programación** se pueden clasificar de la siguiente manera:

- **Lenguaje de máquina**
- **Lenguaje de bajo nivel**
- **Lenguaje de alto nivel**

6.1. Lenguaje de máquina

Está escrito en un "idioma" que entiende el microprocesador (el CPU, el cerebro de la máquina). Las instrucciones son cadenas de 0 y 1 en código binario. Por ejemplo, la instrucción sumar los números 9 y 11 podría ser 0001 1100 0111 0011 1001 1011

Ventaja: la velocidad de ejecución de los programas es directa pues el microprocesador las entiende y ejecuta.

Desventaja: es un lenguaje de símbolos muy diferentes al nuestro, por lo tanto la codificación del programa se hace lenta y se pueden cometer muchos errores. Las instrucciones en código de máquina dependen del hardware de la computadora y por lo tanto difieren de una a otra.

6.2. Lenguaje de bajo nivel

Los lenguajes de bajo nivel por excelencia son los ENSAMBLADORES o Lenguaje ASSEMBLER. Las instrucciones son mnemotécnicos, como por ejemplo ADD, DIV, STR, etc. Una instrucción del ejemplo anterior sería ADD 9,11.

Ventaja: mayor facilidad de escritura que el lenguaje anterior.

Desventaja: depende del microprocesador (existe un lenguaje para cada uno). La formación de los programadores es más compleja que la correspondiente a los programadores de alto nivel ya que exige no sólo técnicas de programación sino también el conocimiento del funcionamiento interno de la máquina. Se utiliza en la programación de PLC, control de procesos, aplicaciones de tiempo real, control de dispositivos. Un programa escrito en un lenguaje de este tipo no puede ser ejecutado directamente por el procesador, y debe ser traducido a código de máquina mediante un ensamblador. No se debe confundir, aunque en español adoptan el mismo nombre, el programa ensamblador ASSEMBLER encargado de efectuar la traducción del programa a código de máquina, con el lenguaje de programación ensamblador ASSEMBLY LENGUAGE.

6.3. Lenguaje de alto nivel

Está diseñado para que las personas escriban y entiendan los programas de modo mucho más natural, acercándose al lenguaje natural. Una instrucción del ejemplo anterior sería 9 + 11. Los lenguajes de alto nivel son los más populares y existe una variedad muy grande. Algunos ejemplos son BASIC, Pascal, C, C++, Cobol, Fortran, Delphi, Java, Python, SQL y todas sus variantes.

Ventajas: son los más utilizados por los programadores. Son independientes de la máquina, las instrucciones de estos lenguajes no dependen del microprocesador, por lo cual se pueden correr en diferentes computadoras. Son transportables, lo que significa la posibilidad de poder ser ejecutados con poca o ninguna modificación en diferentes tipos de computadoras. La escritura de los programas con este tipo de lenguajes se basa en reglas sintácticas similares a nuestro lenguaje, por lo tanto el aprestamiento de los programadores es mucho más rápida que en los lenguajes anteriormente nombrados.

Desventajas: Ocupan más lugar de memoria principal (RAM) que los anteriores. El tiempo de ejecución es mayor pues necesitan varias traducciones.

7. Paradigmas de Programación

La elaboración de un algoritmo se puede realizar en forma totalmente libre, sin seguir los lineamentos de ningún modelo o en caso contrario aplicando las orientaciones de un prototipo. En el desarrollo de un algoritmo siguiendo un modelo establecido podemos citar, entre otros, a los siguientes modelos:

- Procedimental (o imperativo)
- Declarativo
- Funcional
- Orientado a objetos

7.1. Procedimental (o imperativo)

Un lenguaje imperativo es un conjunto de instrucciones que se ejecutan una por una, de principio a fin, de modo secuencial excepto cuando intervienen instrucciones de salto de secuencia o control. (C, PASCAL).

7.2. Declarativo

Un lenguaje declarativo utiliza el principio del razonamiento lógico para responder a las preguntas o cuestiones consultadas. Se basa en la lógica formal y en el cálculo de predicados de primer orden.

En los lenguajes declarativos, se escribe una especificación que describe el problema que se resolverá, y la implementación del lenguaje averigua como realizar el cálculo de forma eficiente (SQL).

7.3. Funcional

Son lenguajes declarativos basados en funciones (con el comportamiento de funciones matemáticas) (Haskell, aunque muchos lenguajes soportan características de paradigma funcional, entre ellos Python).

7.4. Orientado a objetos

El enfoque orientado a objetos guarda analogía con la vida real. El desarrollo de software OO se basa en el diseño y construcción de objetos que se componen a su vez de datos y operaciones que manipulan esos datos. (C++, Python).

8. ¿De qué se trata este curso?

En la actualidad, la mayoría de nosotros utilizamos computadoras permanentemente: para mandar correos electrónicos, navegar por Internet, chatear, jugar, escribir textos. Las computadoras se usan para actividades tan disímiles como predecir las condiciones meteorológicas de la próxima semana, guardar historias clínicas, diseñar aviones, llevar la contabilidad de las empresas o controlar una fábrica. Y lo interesante aquí es que el mismo aparato sirve para realizar todas estas actividades: uno no cambia de computadora cuando se cansa

de chatear y quiere jugar al solitario. Este curso va a tratar precisamente de cómo se escriben programas para hacer que una computadora realice una determinada tarea.

Muchos definen una computadora moderna como *"una máquina que almacena y manipula información bajo el control de un programa que puede cambiar"*. Aparecen acá dos conceptos que son claves: por un lado se habla de una máquina que almacena información, y por el otro lado, esta máquina está controlada por un programa que puede cambiar.

Un programa de computadora es un conjunto de instrucciones paso a paso que le indican a una computadora cómo realizar una tarea dada, y en cada momento uno puede elegir ejecutar un programa de acuerdo a la tarea que quiere realizar.

Las instrucciones se deben escribir en un lenguaje que nuestra computadora entienda. Los lenguajes de programación son lenguajes diseñados especialmente para dar órdenes a una computadora, de manera exacta. Sería muy agradable poder darle las órdenes a la computadora en castellano, pero el problema del castellano, y de las lenguas habladas en general, es su ambigüedad.

En este curso vamos a usar un lenguaje de alto nivel, Python, porque es sencillo y elegante, pero éste no será un curso de Python sino un curso de programación.

Trabajaremos con problemas y aprenderemos a medir los recursos que nos demanda una solución, y empezaremos a buscar la solución menos demandante en cada caso particular.

Ejemplo de un problema que encararemos y de sus soluciones:

Problema 1

Dado un número N se quiere calcular N^{33}. i

Una solución posible, por supuesto, es hacer el producto $N \cdot N \ldots N$ que involucra 32 multiplicaciones.

Otra solución, mucho más eficiente es:

- Calcular $N \cdot N$.
- Al resultado anterior mutiplicarlo por sí mismo con lo cual ya disponemos de N^4.
- Al resultado anterior mutiplicarlo por sí mismo con lo cual ya disponemos de N^8.
- Al resultado anterior mutiplicarlo por sí mismo con lo cual ya disponemos de N^{16}.
- Al resultado anterior mutiplicarlo por sí mismo con lo cual ya disponemos de N^{32}.
- Al resultado anterior mutiplicarlo por N con lo cual conseguimos el resultado deseado con sólo 6 multiplicaciones.

Cada una de estas dos soluciones representa un algoritmo, es decir un método de cálculo, diferente. Para un mismo problema puede haber algoritmos diferentes que lo resuelven, cada uno con un costo distinto en términos de recursos computacionales involucrados.

9. Cómo darle instrucciones a la máquina usando Python

El lenguaje Python nos provee de un *intérprete*, es decir un programa que interpreta las órdenes que le damos a medida que las escribimos. La forma más típica de invocar al intérprete es ejecutar el comando `python3`

en la terminal.

9.1. La terminal

La terminal o consola del sistema operativo permite ingresar órdenes a la computadora en forma de líneas de texto. Los tres sistemas operativos más populares (Windows, Mac OS y Linux) están equipados con una terminal. En la Figura 2 se muestra un ejemplo de una terminal en el sistema operativo Mac OS. En otros sistemas operativos puede verse ligeramente diferente, pero siempre debería mostrar un espacio de texto con un cursor para escribir.

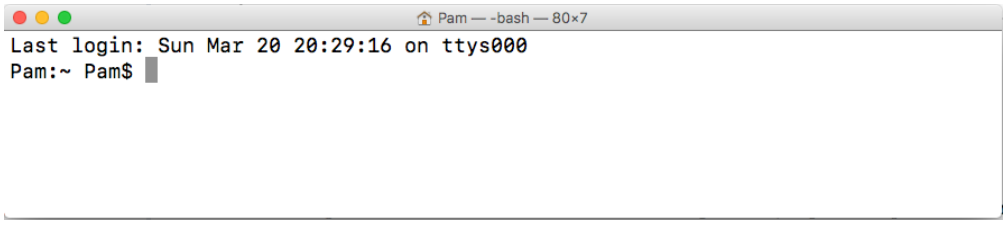

Figura 2: Ejemplo de una terminal de Mac OS.

9.2. El intérprete interactivo de Python

Una vez que accedimos a la terminal del sistema operativo, el próximo paso es abrir el intérprete de Python. Para eso, escribimos `python3` y presionamos `Enter`.

La terminal debería mostrar algo como se ve en la Figura 3.

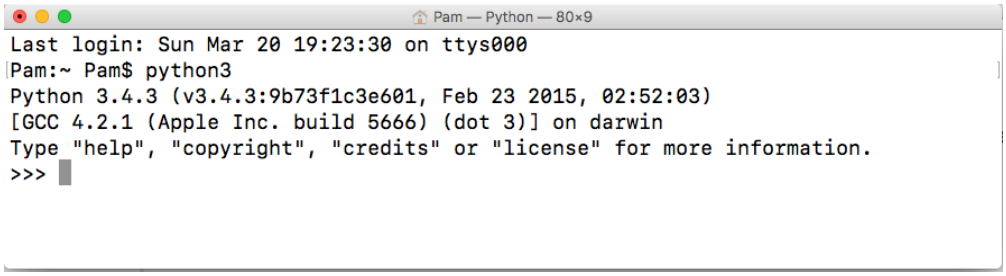

Figura 3: Intérprete de Python.

Las líneas que comienzan con `$` indican órdenes que le damos al sistema operativo (en este caso la orden es `python3`, es decir abrir el intérprete de Python).

Para orientarnos, el intérprete de Python muestra los símbolos `>>>` (llamaremos a esto el *prompt*), indicando que podemos escribir a continuación una sentencia u orden que será evaluada por Python (en lugar de ser evaluada directamente por el sistema operativo).

```
>>> 2 + 3
5
```

Python permite utilizar las operaciones `+`, `-`, `*`, `/`, `//` y `**` (suma, resta, multiplicación, división, división entera y potencia). La sintaxis es la convencional (valores intercalados con operaciones), y se puede usar paréntesis para modificar el orden de asociación natural de las operaciones (potencia, producto/división, suma/resta).

```
>>> 5*7
35
>>> 2+3*7
23
>>> (2+3)*7
35
>>> 10/4
2.5
>>> 10//4
2
>>> 5**2
25
```

9.3. Valores y tipos

En la operación 5 * 7 cuyo resultado es 35, decimos que 5, 7 y 35 son valores. En Python, cada valor tiene un tipo de dato asociado. El tipo de dato del valor 35 es número entero.

Hay dos tipos de datos numéricos: los números enteros y los números de punto flotante. Los números enteros (42, 0, -5, 10000) representan el valor entero exacto que ingresemos. Los números de punto flotante (5.3, -98.28109, 0.0) son parecidos a la notación científica, almacenan una cantidad limitada de dígitos significativos y un exponente, por lo que sirven para representar magnitudes en forma aproximada. Según los operandos y las operaciones que hagamos usaremos la aritmética de los enteros o de los de punto flotante.

Vamos a elegir enteros cada vez que necesitemos recordar un valor exacto: la cantidad de alumnos, cuántas veces repito una operación, un número de documento.

Cuando operamos con números enteros, el resultado es exacto:

```
>>> 1 + 2
3
```

Vamos a elegir punto flotante cuando nos interese más la magnitud y no tanto la exactitud, lo cual suele ser típico en la física y la ingeniería: la temperatura, el seno de un ángulo, la distancia recorrida, etc.

Cuando hay números de punto flotante involucrados en la operación, el resultado es aproximado:

```
>>> 0.1 + 0.2
0.30000000000000004
```

Además de efectuar operaciones matemáticas, Python nos permite trabajar con porciones de texto, que llamaremos **cadenas**, y que se introducen entre comillas simples (') o dobles (").

```
>>> '¡Hola Mundo!'
'¡Hola Mundo!'
>>> 'abcd' + 'efgh'
'abcdefgh'
>>> 'abcd' * 3
'abcdabcdabcd'
```

9.4. Primer programa en Python

Para escribir un programa en Python basta con abrir un editor de texto, escribir nuestro código y guardar el archivo con extensión **.py**.

Vamos a proponer un programa que escriba el mensaje "Hola Mundo"en la terminal. Para ello, escribimos lo siguiente en el archivo:

```
print("Hola Mundo")
```

y lo guardamos con el nombre **Hola.py**. Para ejecutarlo sólo es preciso que en la terminal escribamos el comando:

```
$python3 Hola.py
```

¿Qué observa cuando ejecuta la línea anterior en la terminal?

Ahora modifiquemos el código del archivo, reemplace `print("Hola Mundo")` por `"Hola Mundo"`. Vuelva a ejecutar el programa. ¿Qué pasa ahora? ¿Ve algún resultado en pantalla?

9.5. Entorno de desarrollo online

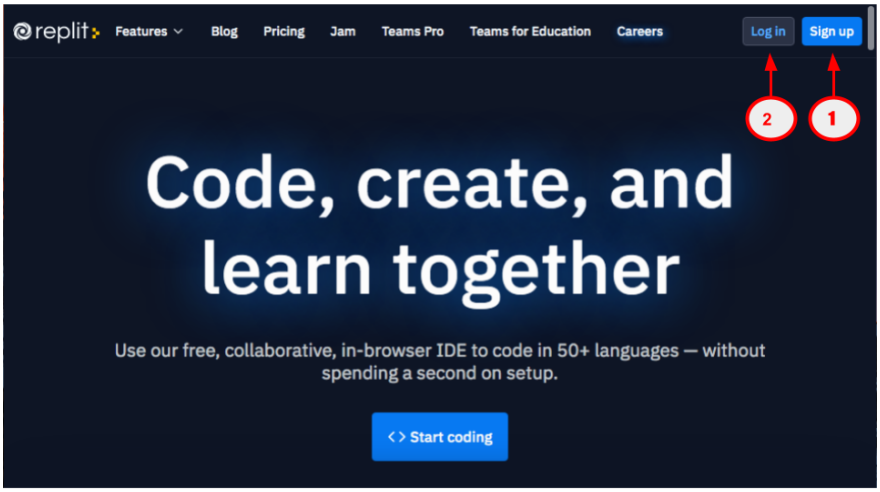

Figura 4: Página de inicio de https://replit.com/

Si bien podemos usar Python de manera local en nuestras computadoras, desde la cátedra alentamos el uso del entorno de desarrollo accesible desde https://replit.com/.

Cuando accedemos por primera vez, accedemos a la página de inicio que se ve en la Figura 4. Lo primero que debemos hacer es hacer click en `Sign up` (identificado en la imagen con el número 1) y crearse un usuario. Luego, una vez que ya hemos creado un usuario debemos conectarnos con dicho usuario haciendo click en `Log in` (identificado con el número 2).

Una vez que haya ingresado con su usuario, verá una página similar a la que se observa en la Figura 5. En la esquina superior izquierda aparecerá su nombre de usuario. Para comenzar a programar debe hacer click en el botón `+ Create` (identificado en la imagen con el número 3).

Se desplegará una ventana similar a la que aparece en la Figura 6 donde deberá elegir en qué lenguaje desea programar, aquí deberá elegir `Python` (ver número 4 en la imagen), deberá dar un nombre a su proyecto, ejemplo “1era Clase"(en el cuadrado identificado con el número 5) y por último deberá hacer Click en el botón `+ Create Repl` (identificado con el número 6). Accederá a continuación a una página con su primer proyecto, tal como se muestra en la Figura 7.

En dicha página, podemos ver tres secciones diferentes como se identifican en la Figura 8. Una sección donde puede navegar entre los diferentes archivos del proyecto (identificada esta sección con el número 7), otra

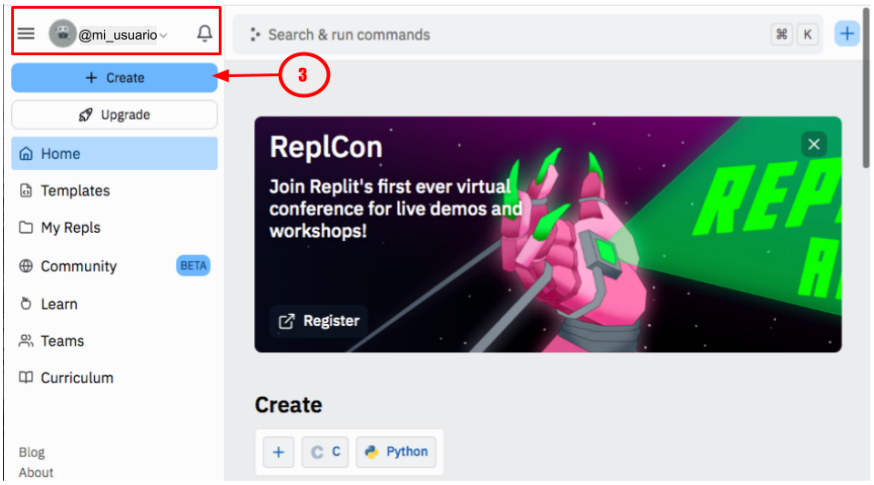

Figura 5: Página personal en replit

Figura 6: Para crear un proyecto en python

sección donde puede ver el contenido del archivo seleccionado en la sección anterior (identificada esta sección con el número 8) y la última sección a la derecha donde verá los resultados de la ejecución de su programa (identificada en la imagen con el número 9). Para ejecutar el programa debe hacer Click en el botón `Run`.

Para completar su primer proyecto, seleccione el archivo `main.py`, escriba en dicho archivo el código `print("Hola Mundo")` y ejecútelo. Felicitaciones! Ha escrito su primer programa en Python usando replit!

10. Sintaxis de Python

A continuación veremos la sintaxis de Python, viendo cómo podemos empezar a usar el lenguaje creando nuestras primeras variables.

El término sintaxis hace referencia al conjunto de reglas que definen como se tiene que escribir el código en un determinado lenguaje de programación. Es decir, hace referencia a la forma en la que debemos escribir las instrucciones para que el ordenador, o más bien lenguaje de programación, nos entienda.

En la mayoría de lenguajes existe una sintaxis común, como por ejemplo el uso de `=` para asignar un dato

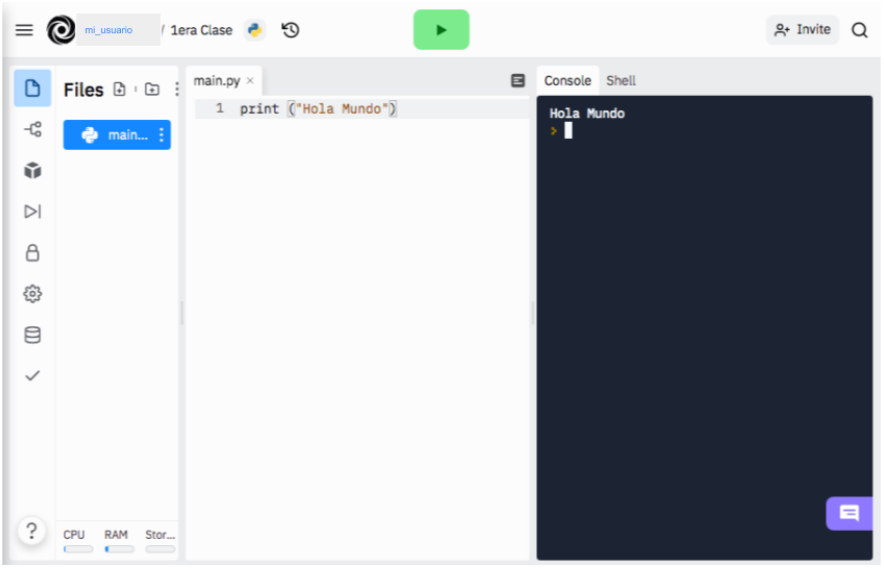

Figura 7: Para crear un proyecto en python

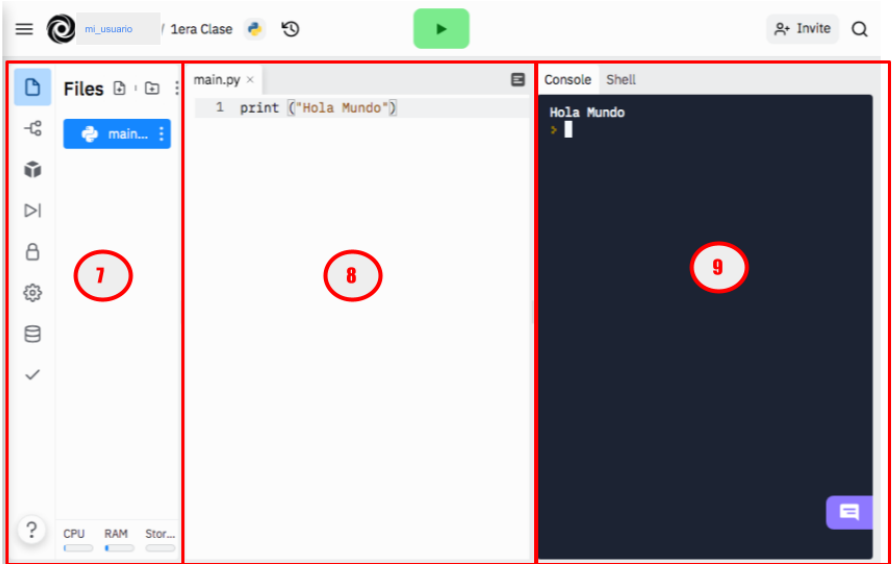

Figura 8: Para crear un proyecto en python

a una variable, o el uso de `{}` para designar bloques de código, pero Python tiene ciertas particularidades. Python no soporta el uso de `$` ni hace falta terminar las líneas con `;` como en otros lenguajes.

Por otro lado, de la misma forma que un idioma no se habla con simplemente saber todas sus palabras, en la programación no basta con saber la sintaxis de un lenguaje para programar correctamente en él. Es cierto que sabiendo la sintaxis podremos empezar a programar y a hacer lo que queramos, pero el uso de un lenguaje de programación va mucho más allá de la sintaxis.

Para empezar a perderle el miedo a la sintaxis de Python, vamos a ver un ejemplo donde vemos cadenas y operadores aritméticos.

```
# Definimos una variable x con una cadena
x = "El valor de (a+b)*c es"
```

```
# Podemos realizar múltiples asignaciones
a, b, c = 4, 3, 2

# Realizamos unas operaciones con a,b,c
d = (a + b) * c

# Imprimir, print()
print(x, d)

# Salida: El valor de (a+b)*c es 14
```

10.1. Comentarios

Los comentarios son bloques de texto usados para comentar el código. Es decir, para ofrecer a otros programadores o a nuestro yo futuro información relevante acerca del código que está escrito. A efectos prácticos, para Python es como si no existieran, ya que no son código propiamente dicho, solo anotaciones.

Los comentarios se inician con # y todo lo que vaya después en la misma línea será considerado un comentario.

```
# Esto es un comentario
```

Al igual que en otros lenguajes de programación, podemos también comentar varias líneas de código. Para ello es necesario hacer uso de triples comillas bien sean simples o dobles. Es necesario usarlas para abrir el bloque del comentario y para cerrarlo.

```
'''
Esto es un comentario
de varias líneas
de codigo
'''


"""
Esto es otro comentario
de varias líneas
de codigo
"""
```

10.2. Identación y bloques de código

En Python los bloques de código se representan con identación, y aunque hay un poco de debate con respecto a usar tabulador o espacios, la norma general es usar cuatro espacios.

En el siguiente código tenemos un condicional `if`. Justo después tenemos un `print()` identado con cuatro espacios. Por lo tanto, todo lo que tenga esa identación pertenecerá al bloque del `if`.

```
if True:
    print("True")
```

Esto es muy importante ya que el código anterior y el siguiente no son lo mismo. De hecho el siguiente código daría un error ya que el `if` no contiene ningún bloque de código, y eso es algo que no se puede hacer en Python.

```
if True:
print("True")
```

De todas formas, volveremos a recordar el tema identación cuando definamos las diferentes estructuras de control.

Por otro lado, a diferencia de en otros lenguajes de programación, no es necesario utilizar ; para terminar cada línea.

```
# Otros lenguajes como C
# requieren de ; al final de cada línea
x = 10;
```

Sin embargo en Python no es necesario, basta con un salto de línea.

```
x = 5
y = 10
```

Pero se puede usar el punto y coma ; para tener dos sentencias en la misma línea.

```
x = 5; y = 10
```

10.3. Múltiples líneas

En algunas situaciones se puede dar el caso de que queramos tener una sola instrucción en varias línea de código. Uno de los motivos principales podría ser que fuera demasiado larga, y de hecho en la especificación PEP8 se recomienda que las líneas no excedan los 79 caracteres.

Haciendo uso de \ se puede romper el código en varias líneas, lo que en determinados casos hace que el código sea mucho más legible.

```
x = 1 + 2 + 3 + 4 +\
    5 + 6 + 7 + 8
```

Si por lo contrario estamos dentro de un bloque rodeado con paréntesis (), bastaría con saltar a la siguiente línea.

```
x = (1 + 2 + 3 + 4 +
    5 + 6 + 7 + 8)
```

10.4. Creando variables

Anteriormente ya hemos visto como crear una variable y asignarle un valor con el uso de =. Existen también otras formas de hacerlo de una manera un poco más sofisticada.

Podemos por ejemplo asignar el mismo valor a diferentes variables con el siguiente código.

```
x = y = z = 10
```

O también podemos asignar varios valores separados por coma.

```
x, y = 4, 2
x, y, z = 1, 2, 3
```

10.5. Nombrando variables

Puedes nombrar a tus variables como quieras, pero es importante saber que las mayúsculas y minúsculas son importantes. Las variables `x` y `X` son distintas.

Por otro lado existen ciertas normas a la hora de nombrar variables:

- El nombre no puede empezar por un número
- No se permite el uso de guiones -
- Tampoco se permite el uso de espacios.

Se muestran unos ejemplos de nombres de variables válidos y no válidos.

```
# Valido
_variable = 10
vari_able = 20
variable10 = 30
variable = 60
variaBle = 10

# No valido
2variable = 10
var-iable = 10
var iable = 10
```

Una última condición para nombrar a una variable en Python, es no usar nombres reservados para Python. Las palabras reservadas son utilizadas por Python internamente, por lo que no podemos usarlas para nuestras variables o funciones.

```
import keyword
print(keyword.kwlist)

# ['False', 'None', 'True', 'and', 'as', 'assert',
# 'async', 'await', 'break', 'class', 'continue',
# 'def', 'del', 'elif', 'else', 'except', 'finally',
# 'for', 'from', 'global', 'if', 'import', 'in', 'is',
# 'lambda', 'nonlocal', 'not', 'or', 'pass', 'raise',
# 'return', 'try', 'while', 'with', 'yield']
```

De hecho con el siguiente comando puedes ver todas las palabras clave que no puedes usar.

```
import keyword
print(keyword.kwlist)
```

10.6. Uso de paréntesis

Python soporta todos los operadores matemáticos más comunes, conocidos como operadores aritméticos. Por lo tanto podemos realizar sumas, restas, multiplicaciones, exponentes (usando `**`) y otros que no vamos a explicar por ahora. En el siguiente ejemplo realizamos varias operaciones en la misma línea, y almacenamos su resultado en `y`.

```
x = 10
y = x*3-3**10-2+3
```

Pero el comportamiento del código anterior y el siguiente es distinto, ya que el uso de paréntesis () da prioridad a unas operaciones sobre otras.

```
x = 10
y = (x*3-3)**(10-2)+3
```

10.7. Uso de la función print()

Por último, en cualquier lenguaje de programación es importante saber lo que va pasando a medida que se ejecutan las diferentes instrucciones. Por ello, es interesante hacer uso de print() en diferentes secciones del código, ya que nos permiten ver el valor de las variables y diferente información útil.

Existen muchas formas de usar la función `print()`, pero por ahora basta con que sepas lo básico.

Como ya hemos visto se puede usar `print()` para imprimir por pantalla el texto que queramos.

```
print("Esto es el contenido a imprimir")
```

```
x = 10
print(x)
```

Y separando por comas `,` los valores, es posible imprimir el texto y el contenido de variables.

```
x = 10
y = 20
print("Los valores x, y son:", x, y)
# Salida: Los valores x, y son: 10 20
```

10.8. Interacción con el usuario

Ya vimos que la función `print` nos permite mostrar información al usuario del programa. En algunos casos también necesitaremos que el usuario ingrese datos al programa. Por ejemplo:

Problema 2

Queremos escribir en Python un programa que pida al usuario que escriba su nombre, y luego lo salude.

Ya habíamos escrito un programa que imprimía un saludo general. Pero aún no sabemos cómo obtener el nombre del usuario. Para esto podemos usar la función input, como se muestra siguiente código

```
nombre = input("Por favor ingrese su nombre: ")
saludo = "Hola " + nombre
print(saludo)
```

Usamos `input` para pedirle al usuario su nombre. input presenta al usuario el mensaje que le pasamos por parámetro, y luego le permite ingresar una cadena de texto. Cuando el usuario presiona la tecla `Enter`, `input` devuelve la cadena ingresada. Luego usamos la concatenación para generar el saludo, y a `print` para mostrarlo al usuario.

Veamos ahora otro ejemplo, donde se necesitan datos que no son cadenas de caracteres.

Problema 3

Escribir en Python un programa que haga lo siguiente:

- Muestre un mensaje de bienvenida por pantalla.
- Le pida al usuario que introduzca dos números enteros `n1` y `n2`.
- Imprima la suma y la multiplicación de `n1` y `n2`.
- Muestre un mensaje de despedida por pantalla

```
print("Se calcularán la suma y la multiplicación de dos números")
n1 = int(input("Ingrese un número entero: "))
n2 = int(input("Ingrese otro número entero: "))

suma = n1 + n2
producto = n1 * n2

print("El resultado de la suma es ", suma,
" y de la multiplicación, ", producto)

print("Es todo por ahora")
```

En el código aparece una función que no habíamos utilizado hasta ahora: `int`. ¿Por qué es necesario utilizar `int` para resolver el problema?

La función `input` interpreta cualquier valor que el usuario ingresa mediante el teclado como una cadena de caracteres. Es decir, `input` siempre devuelve una cadena, incluso aunque el usuario haya ingresado una secuencia de dígitos. Por eso es que introducimos la función `int`, que devuelve el parámetro que recibe convertido a un número entero:

```
int("42") # 42
```

De acuerdo al tipo de dato que se espera, se deberá realizar la conversión de tipo que corresponda. Lo iremos viendo a medida que lo vayamos necesitando.

11. Bibliografía

- Apunte de la materia Algoritmos y Programación 1, primera materia de programación de las carreras de Ingeniería en Informática y Licenciatura en Análisis de Sistemas dela Facultad de Ingeniería de la UBA: http://materias.fi.uba.ar/7501/apunte%20PYTHON.pdf
- Libro digital: https://ellibrodepython.com/
- Elementos esenciales para programación: Algoritmos y Estructuras de Datos - Laura Angelone y otros
- Fundamentos de programación - Luis Joyanes Aguilar
- https://docs.python.org/es/3.8/howto/functional.html (Doc oficial Python)
- http://www.frlp.utn.edu.ar/materias/sintaxis/clase9-Semantica.pdf (UTN - Reginal La Plata)