



UNIVERSIDAD NACIONAL
DE ASUNCIÓN
FACULTAD DE
INGENIERÍA



Cátedra de Fundamentos de Programación

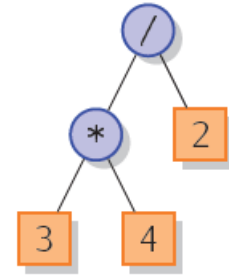
Introducción a Python

Elementos básicos de un programa



Basado en materiales de referencia elaborados por los Profesores Cristian Cappelletti, Diego Stalder, José Colbes y Nestor Barreto

¿Qué veremos hoy?

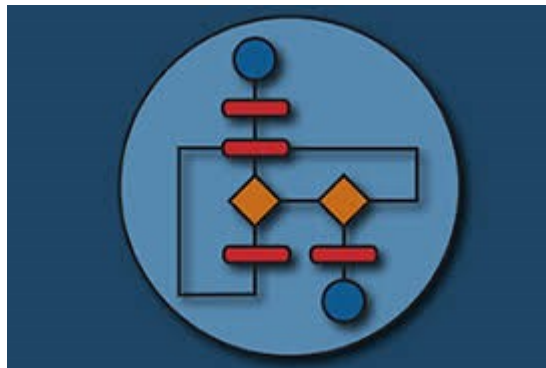


- Programa e instrucciones
- Elementos básicos de un programa
- Datos, Tipos de datos simples.
- Constantes y Variables.
- Introducción a Python:
 - Tipos de datos
 - Operadores y expresiones
 - Reglas de precedencia de los operadores
 - Funciones internas
 - Funciones de entrada y salida de datos



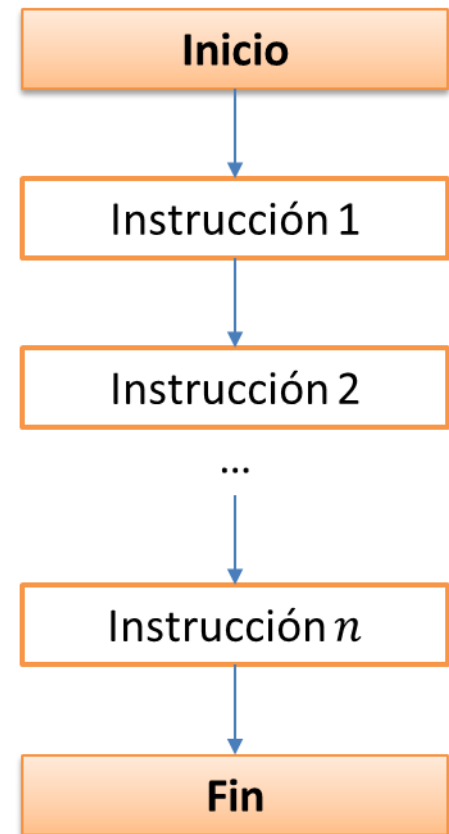
Concepto de programa

- Conjunto de instrucciones (órdenes dada a un equipo) que producirán la ejecución de una tarea.
- Es la representación concreta de un algoritmo en algún lenguaje de programación (ej. Python, C/C++, Java).



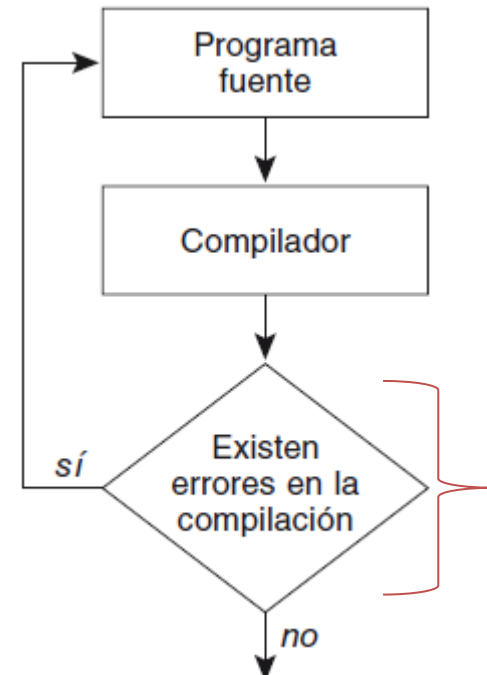
Instrucciones

- Una instrucción (o sentencia) es una **acción** que se indica a realizar a la computadora. Un programa puede verse como un conjunto de instrucciones.
- Las instrucciones se deben escribir y posteriormente almacenar en memoria RAM en el mismo orden en que han de ejecutarse, en una **secuencia**.



Tipos de instrucciones

- Instrucciones de inicio/fin.
- Instrucciones de asignación (=).
- Instrucciones de lectura (leer o cargar).
- Instrucciones de escritura (imprimir o escribir).
- Instrucciones de bifurcación/selección



B
i
f
u
r
c
a
c
i
ó
n

Tipo de instrucción	Pseudocódigo inglés	Pseudocódigo español
comienzo de proceso	begin	inicio
fin de proceso	end	fin
entrada (lectura)	read	leer
salida (escritura)	write	escribir
asignación	$A \leftarrow 5$	$B \leftarrow 7$

Elementos básicos de un programa

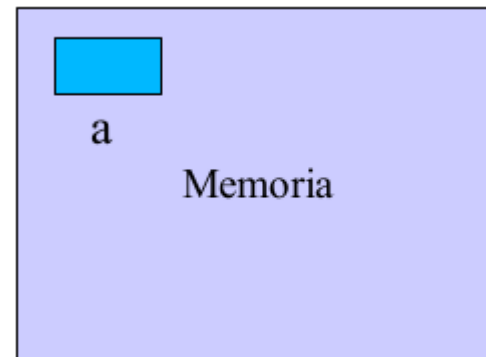
- **Palabras reservadas** (if, else, elif, while, for, print, int, etc)
- **Identificadores** (nombres de variables, nombre de constantes, nombre de tipo de dato estructurado, nombre del programa, subprograma: procedimientos, funciones ...)
- **Caracteres especiales** (coma, apóstrofo, ...)
- **Variables** (a, b, c, peso, edad, nombre, _fecha ...)
- **Constantes** (PI, IVA, ...)
- **Expresiones** ((a + b / c), A>B, not True, c == d)
- **Instrucciones** (x=10; x+= 1; input("edad: "); print(nombre);...)

Datos – Tipos de datos simples

- Un **dato** es la expresión general que describe los objetos con los cuales opera una computadora.
- El **tipo de dato** indica la naturaleza del dato. Delimita el rango posible de valores (dominio) y las operaciones posibles aplicables al dato.
- Los tipos de datos básicos son (en pseudocódigo):
 - **Numérico** (enteros, reales)
 - **Lógico o booleano** (falso, verdadero)
 - **Carácter** (carácter, cadena)

Variables

- Una variable es un espacio de memoria reservado para almacenar valores, cuyo contenido puede variar durante el desarrollo del algoritmo o la ejecución del programa.
- Las variables pueden ser de tipo numérica, cadena o lógica (en pseudocódigo), o ser variables complejas o estructuradas.
- En general, una variable se identifica por su nombre y tipo.
- En general, una variable que es de un cierto tipo puede tomar únicamente valores de ese tipo.



Constantes

- Una constante es un valor que permanece inalterado durante todo el desarrollo del algoritmo o durante la ejecución del programa. Recibe un valor en el momento de la compilación y este permanece inalterado durante todo el programa.
- La constante puede ser de tipos simples o estructurados.
- $E = 2,71828182845904$
- `SALUDO = "Hola"`

Constantes nominales

- También llamadas **constantes con nombre**, son las que se declaran asignándoles un valor directamente.
- Por convención, los nombres de las constantes suelen estar en MAYÚSCULAS. Por ejemplo:
 - ASIGNATURA = “Fundamentos”
 - PI = 3.1416
 - BASE = 2
 - PORCENTAJE_IVA = 10

Python no tiene constantes nominales directamente, aunque puede obtenerse similares resultados mediante la definición de clases (esto escapa a los objetivos del curso).

Constantes literales

- Son valores de cualquier tipo que se utilizan directamente. No se declaran, ya que no tienen nombre, y forman parte del texto del programa.
- Es un valor en sí mismo, el cual es almacenado como parte del programa.
- Ejemplo de constantes literales: **180**, **3.1416**, **"Hola Mundo\n"**.

```
A_rad = a*math.pi/180 //usamos el valor 180 como literal entero
```

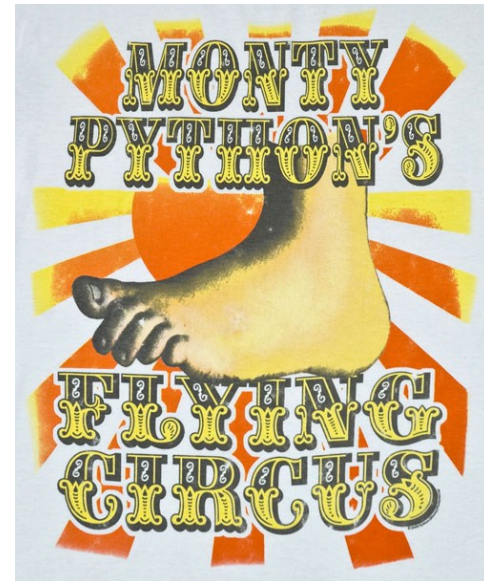
Identificadores

- Son los **nombres** elegidos para cada una de las variables, constantes, subprogramas, registros, etc. Estos identificadores usados en el algoritmo o el programa deben ser significativos y tener relación con el objeto que representan.
- Como buena práctica, para nuestro curso un identificador:
 - Comienza con una letra (mayúscula/minúscula) o con un “_” (guión bajo). No debe tener espacios.
 - No tienen límite de longitud (caso de Python, otros LP pueden tener limitaciones).
 - Las mayúsculas y minúsculas son diferentes.

Python

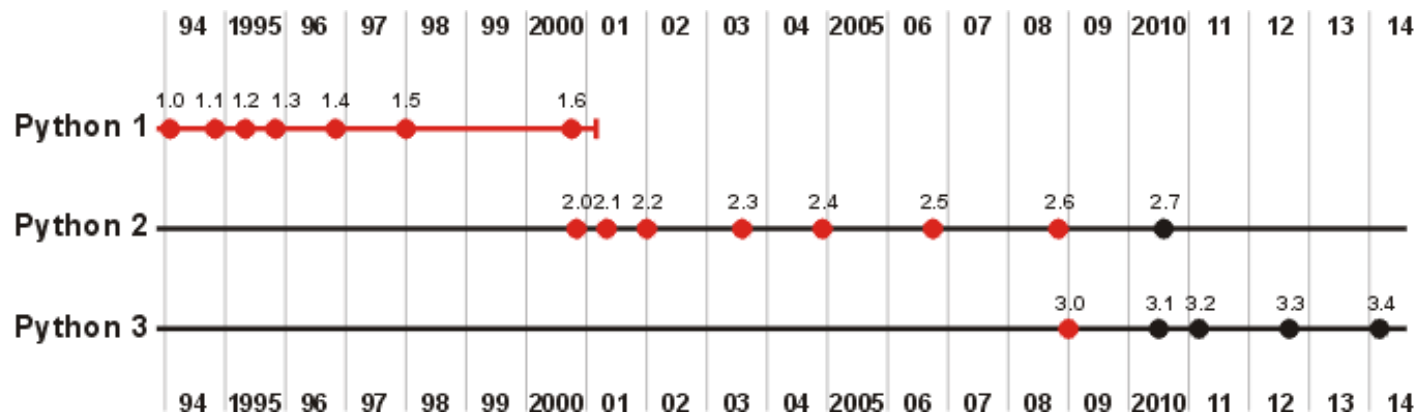


Guido van Rossum



Python - Historia

- Python fue concebido y desarrollado al final de los 80's y al comienzo de los 90's por Guido van Rossum en el Instituto Nacional de Matemáticas y Ciencias de la Computación en Países Bajos.), en principio como un proyecto de afición para mantenerse ocupado durante las vacaciones de Navidad.
- La versión 1.0 fue lanzada en Enero de 1994.
- La sintaxis de Python y ciertos aspectos de su filosofía son directamente heredadas del lenguaje ABC, pero Python es también influenciado por Modula-3, C, C++, Perl, Java, el Shell de Unix, y otros lenguajes de *script*.

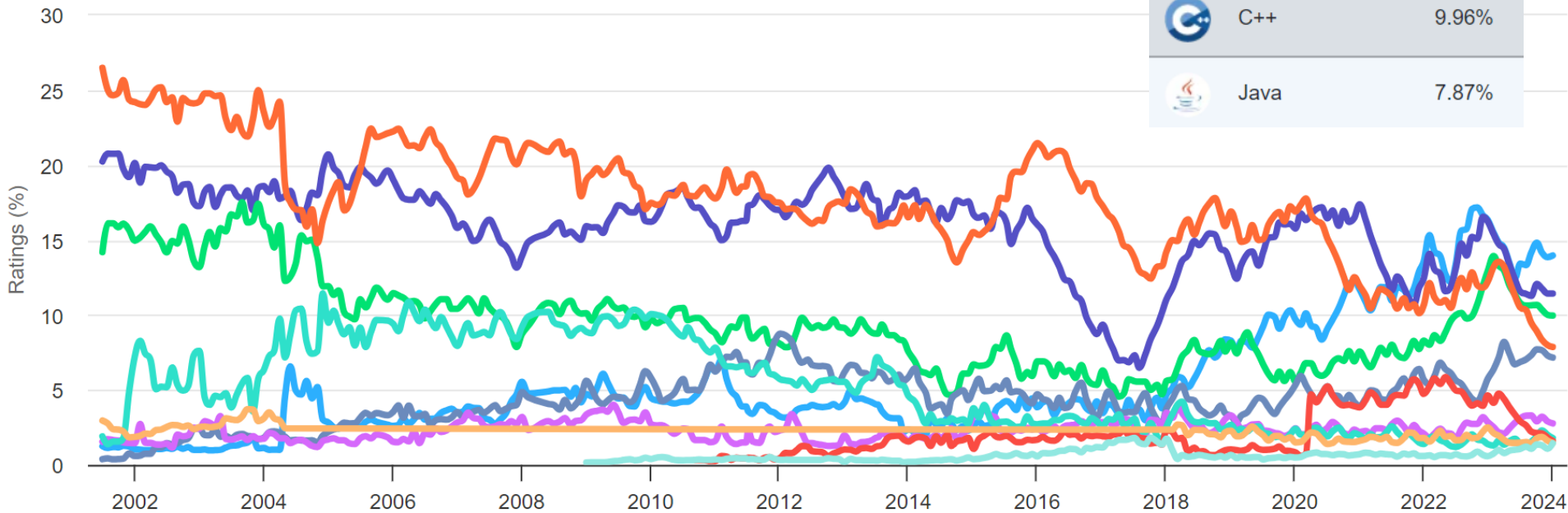


¿Por qué aprender Python?

Su popularidad

TIOBE Programming Community Index

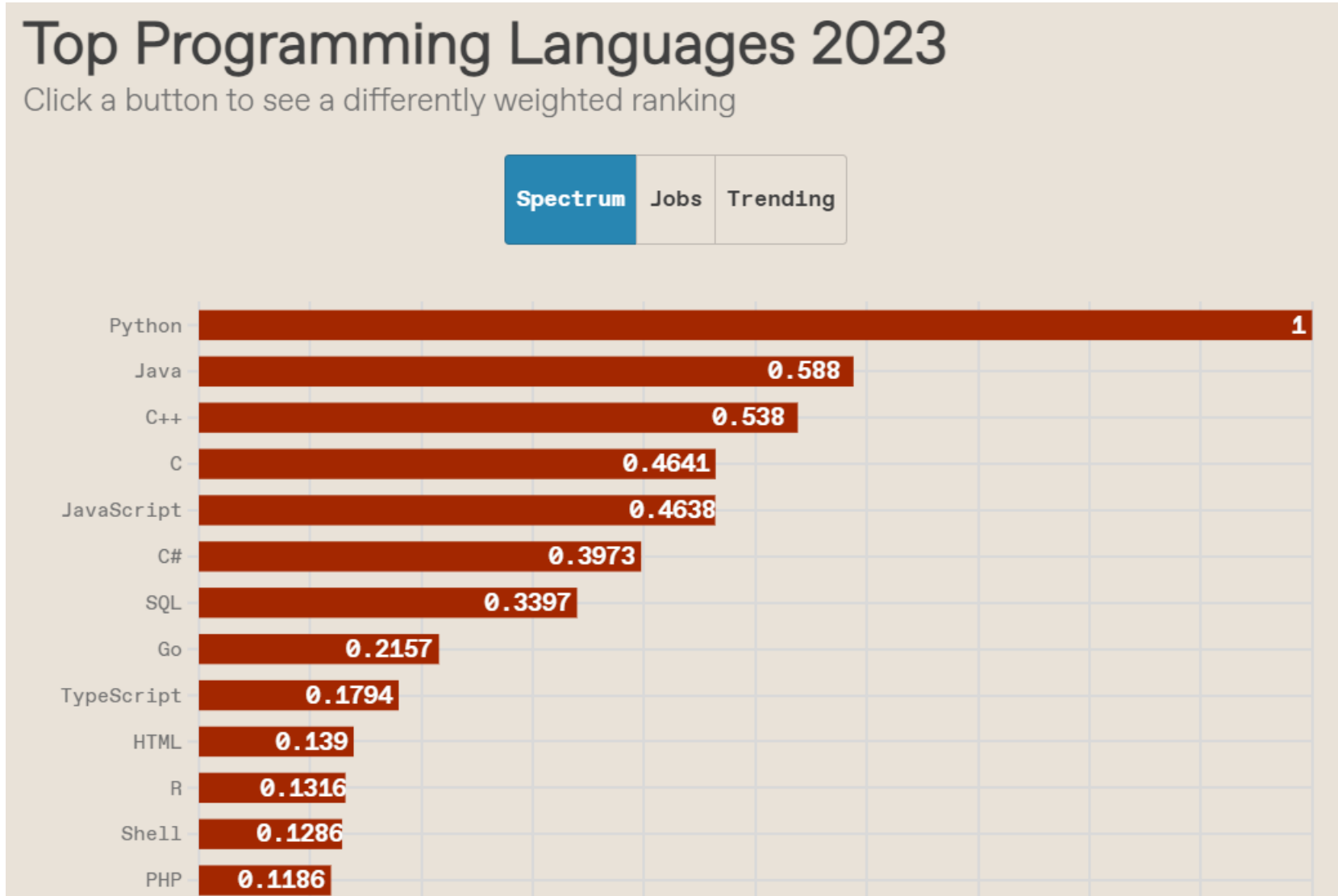
Source: www.tiobe.com



<https://www.tiobe.com/tiobe-index/>

The Top Programming Languages 2023

IEEE Spectrum magazine <https://spectrum.ieee.org/the-top-programming-languages-2023>



Python Developer Salaries by Degree Level

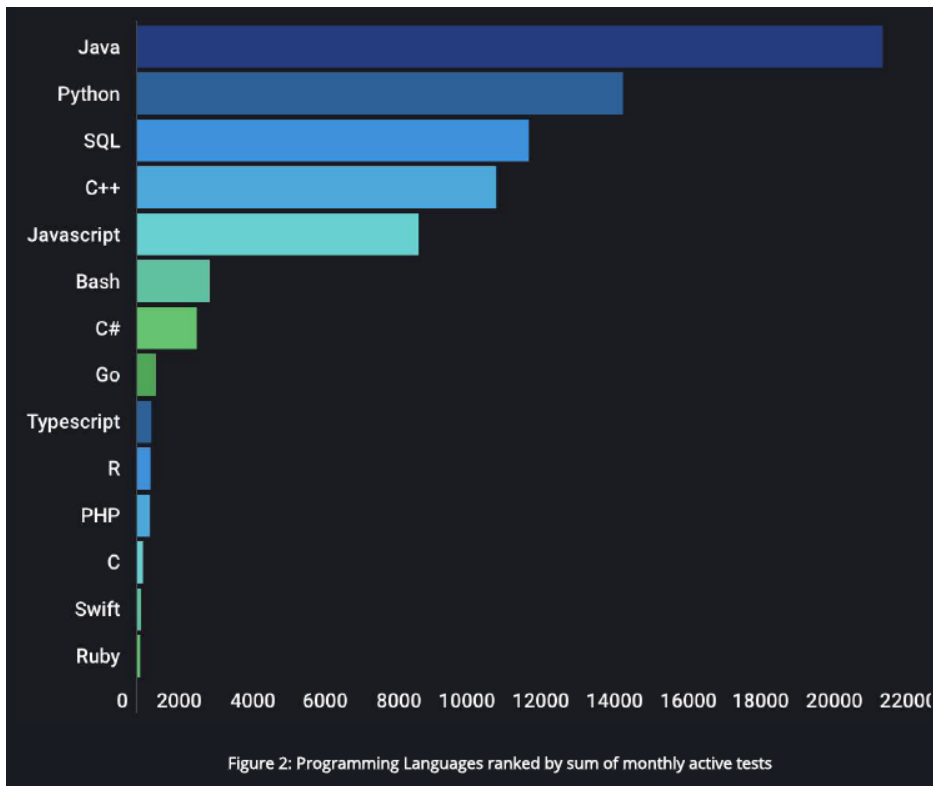
Python Developer with the following degree	Will likely fall in this salary range
Associate's Degree	\$66,805 - \$70,765
Bachelor's Degree	\$67,921 - \$71,450
Master's Degree or MBA	\$68,937 - \$72,138
JD, MD, PhD or Equivalent	\$69,952 - \$72,826

* <https://www1.salary.com/Salaries-for-python-developer-with-a-JD-MD-PhD-or-Equivalent>

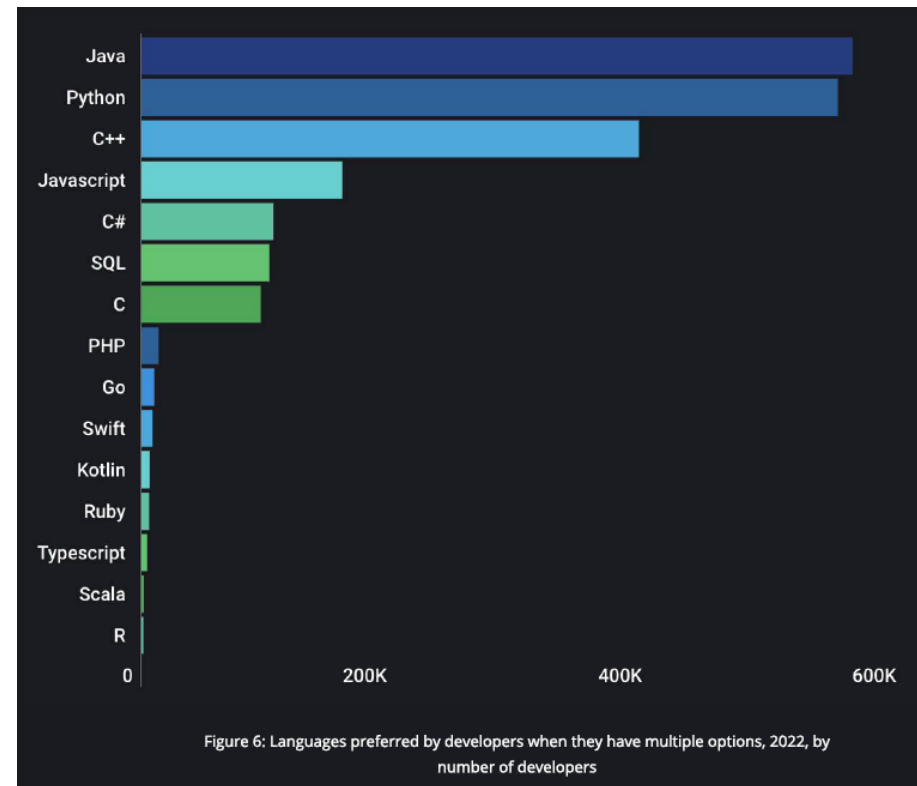
HackerRank Developer Skills Report 2023

Comunidad de 21 millones de miembros

Ranking de lenguajes de programación (demanda)



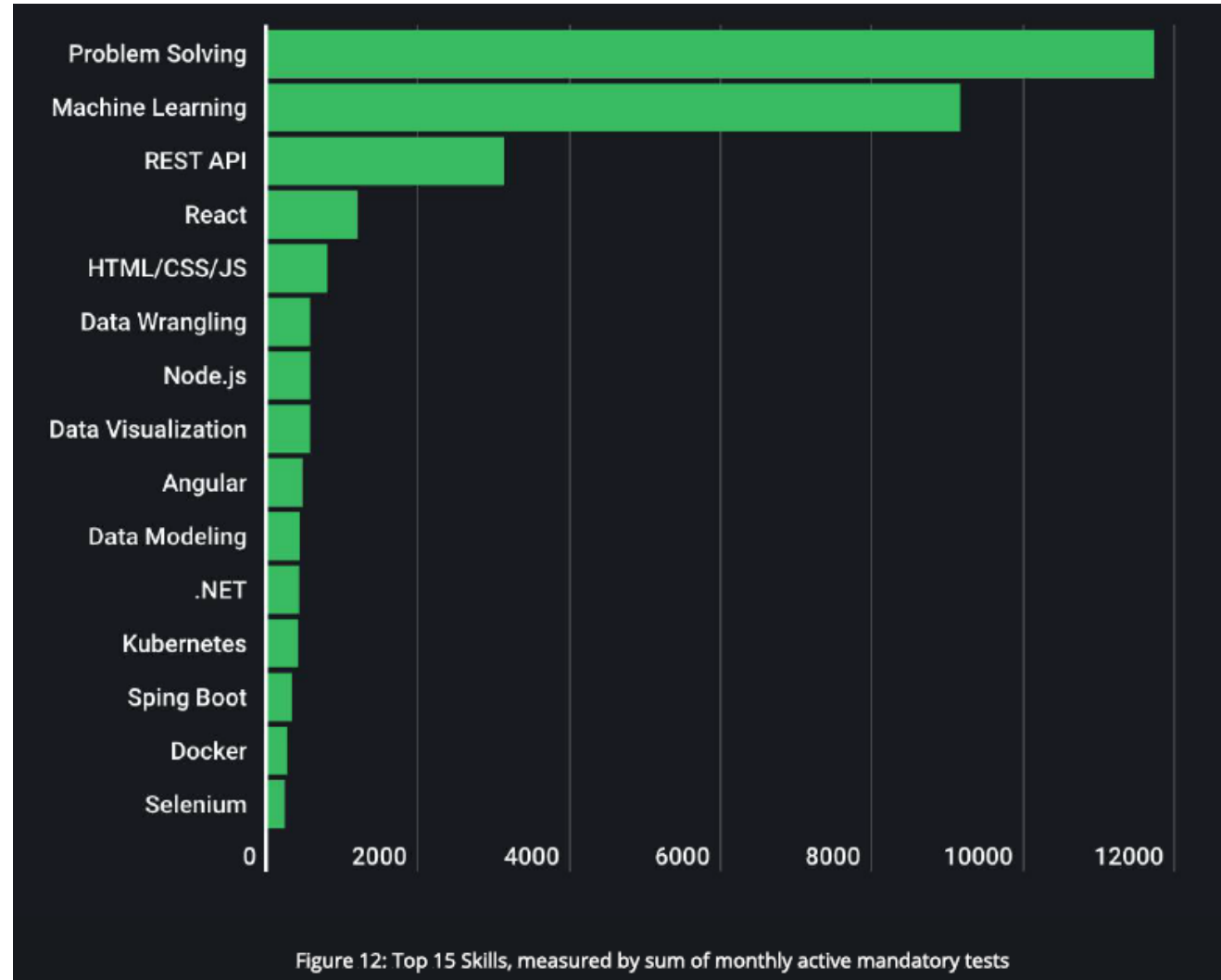
Lenguajes preferidos por los programadores



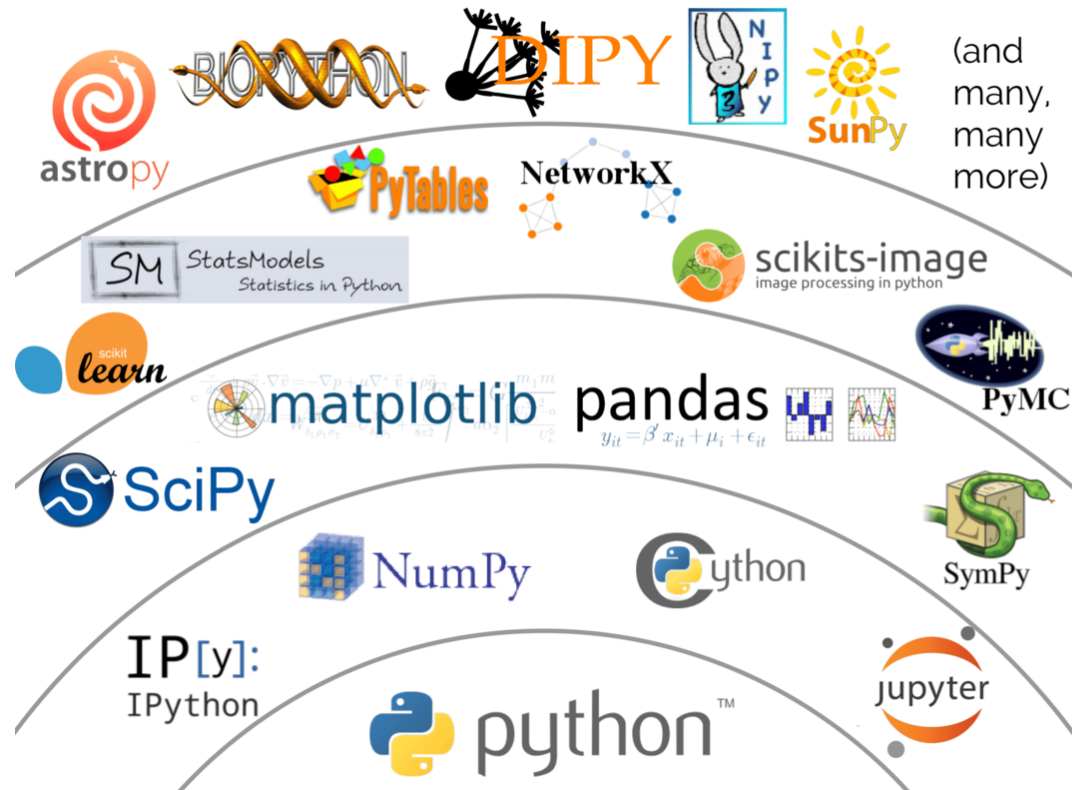
HackerRank Developer Skills Report 2023

Comunidad de 21 millones de miembros

Habilidades
solicitadas



Librerías en Python



Existen muchísimos proyectos (como librerías) además de las librerías que son estándar. Actualmente más de 380K proyectos (<https://pypi.org/>) para todas las áreas.

Python - Características

- Es simple de usar!



```
#include <stdio.h>
main()
{
    printf("Hola mundo!!\n");
}
```



```
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello, World");
    }
}
```

<http://www2.latech.edu/~acm/HelloWorld.shtml>



```
#include <iostream>

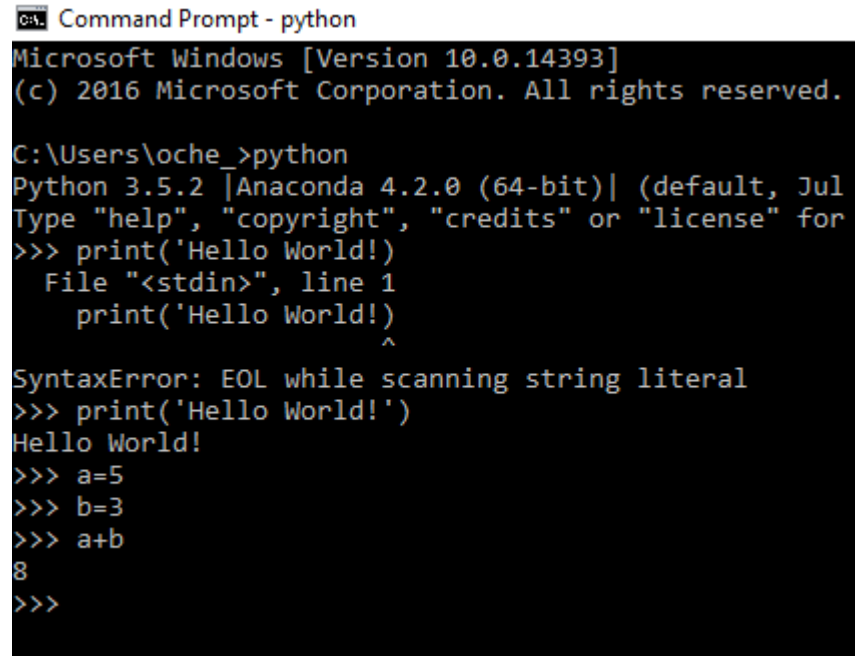
int main()
{
    std::cout << "Hello, world!\n";
}
```



```
print("Hello World!")
```

Python - Características

- Es simple de usar!
- Es interactivo
- Es portable
- Es extensible
- Tiene una gran librería estándar
- Es escalable



```
C:\> Command Prompt - python
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

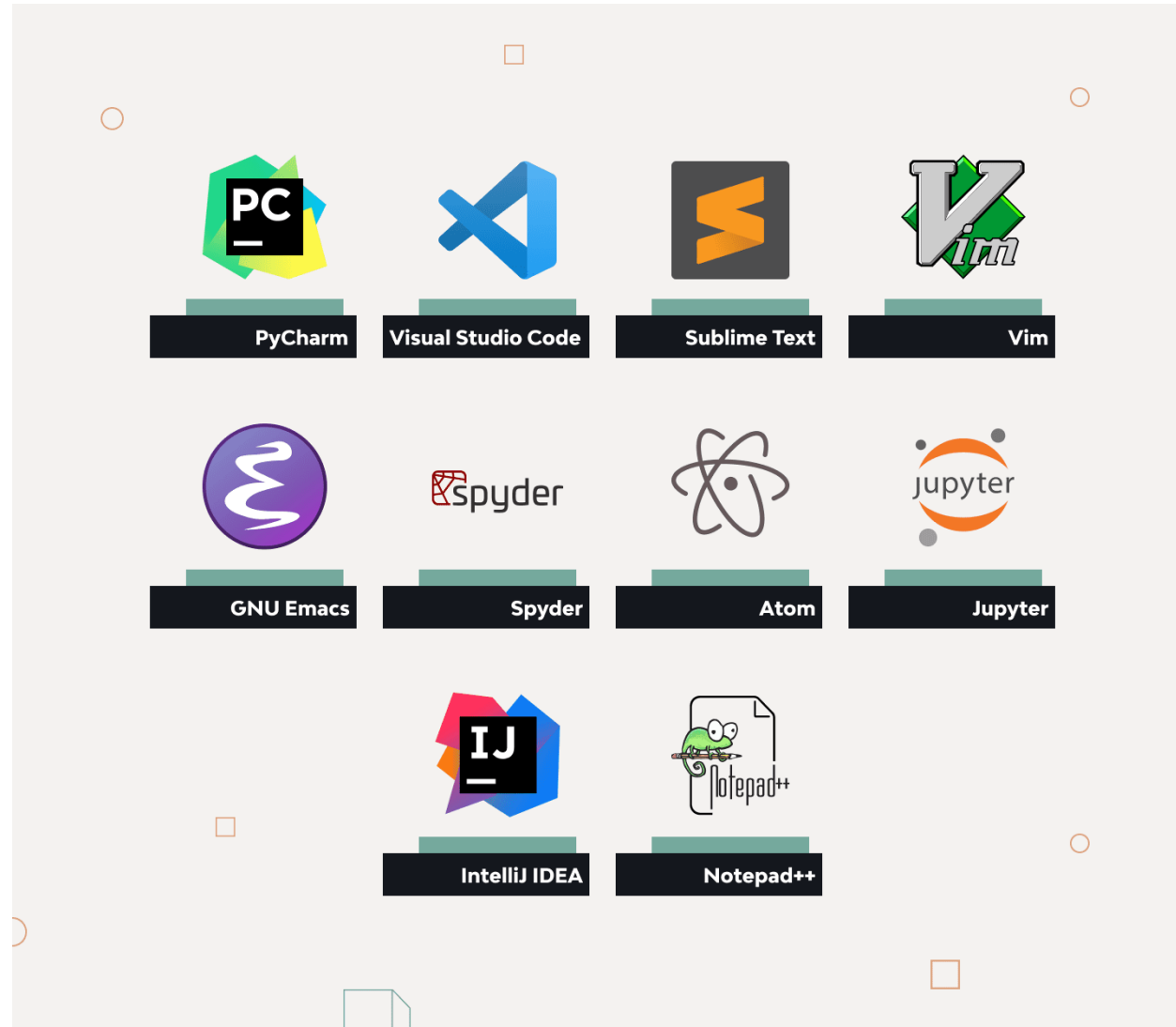
C:\Users\oche_>python
Python 3.5.2 |Anaconda 4.2.0 (64-bit)| (default, Jul
Type "help", "copyright", "credits" or "license" for
>>> print('Hello World!)
      File "<stdin>", line 1
        print('Hello World!)
              ^
SyntaxError: EOL while scanning string literal
>>> print('Hello World!')
Hello World!
>>> a=5
>>> b=3
>>> a+b
8
>>>
```

Python - Características



- Disponible en: <https://www.python.org>
- Es un lenguaje de programación de propósito general. Es multiparadigma (POO, funcional, concurrente, reflectivo, etc.).
- Es multiplataforma, portable.
- Es libre y de fuente abierta.
- Es interpretado. De sintaxis sencilla y código compacto.
- ¿Quién usa? Intel, IBM, NASA, Pixar, Netflix, FB, Spotify, Google, etc.
- Existen muchísimos proyectos (como librerías) además de las librerías que son estándar. Actualmente más de 380K proyectos (<https://pypi.org/>) para todas las áreas.

Editores e IDE



Estructura de un programa

<i>Comentarios</i>
import numpy as np
funciones

```
# Imprimir Pi
```

```
"""
```

```
comentario
```

```
largo
```

```
"""
```

```
import math
```

```
x = math.pi
```

```
print(x)
```

Los comentarios son ignorados por el intérprete, pero son útiles para los programadores, pues pueden proporcionar información sobre el código.

Variables en Python

- Las variables en su propia declaración ya pueden ser inicializadas con valores del tipo correspondiente.
- Ejemplos de variables:

```
>>> nombre = "juan"
>>> precio_unitario = 100
>>> Promedio = 4.7
>>> promedio = 3.9
>>> SueldoBruto = 1000
>>> esPar = True
```

Creando variables

- No se declaran variables en python.
- La variable se crea en la primera asignación

```
x = 5
y = "Juan Perez"
print(x)
print(y)
```

Con Comilla simple es equivalente
y = 'Juan Perez'

- La variable puede cambiar de tipo

```
x = 4      # x es entero int
x = "Juan Perez" # x ahora es cadena str
print(x)
```

Reglas para los identificadores

1. Un identificador se forma con una **secuencia de letras** (minúsculas de la a a la z; mayúsculas de la A a la Z; y dígitos del 0 al 9).
2. El carácter subrayado o **guión bajo** (_) se considera como una letra más.
3. Un identificador **no puede contener espacios en blanco, ni otros caracteres distintos de los citados**, como por ejemplo (* , ; . : - + , etc.).
4. El **primer carácter** de un identificador debe ser siempre **una letra o un (_)**, es decir, no puede ser un dígito.
5. Se hace **distinción entre letras mayúsculas y minúsculas**. Así, Masa es considerado como un identificador distinto de masa y de MASA.

Ejemplos de identificadores válidos son los siguientes:

- tiempo, distancia1, caso_A, Pl, velocidad_de_la_luz

Por el contrario, los siguientes nombres no son válidos (¿Por qué?)

- 1_valor, tiempo-total, dolares\$, %final

Tipos de datos simples en Python

En el curso distinguiremos tres tipos de datos básicos (entre paréntesis se indican los tipos que corresponden a Python):

- **Numérico** (`int`, `float`, `complex`): representa información numérica ya sea entera o fraccionaria. Por ejemplo: `1912`, `18121000`, `14e+30` (`=14e30=1.4e+31`), `0.0002`, `0.152`, `1e-10`, `1j+45`, `5j-87`, etc
- **Cadena** (`str`): representa información textual, y está compuesta por uno o más caracteres. Por ejemplo: `"Hola!"`, `"Estamos en el curso de fundamentos de programación"`, `'Pepe'`, `'Coche'`, etc.
- **Lógico** (`bool` o `boolean`) : representa datos que pueden tomar dos valores posibles: verdadero (`True`) o falso (`False`).

En Python, una variable adquiere su tipo de dato de acuerdo al valor asignado (tipo de constante literal)

Ejemplos de tipos de datos

<code>x = "Hello World"</code>	<code>str</code>
<code>x = 20</code>	<code>int</code>
<code>x = 20.5</code>	<code>float</code>
<code>x = 1j</code>	<code>complex</code>
<code>x = ["apple", "banana", "cherry"]</code>	<code>list</code>
<code>x = ("apple", "banana", "cherry")</code>	<code>tuple</code>
<code>x = range(6)</code>	<code>range</code>
<code>x = {"name" : "John", "age" : 36}</code>	<code>dict</code>
<code>x = {"apple", "banana", "cherry"}</code>	<code>set</code>
<code>x = frozenset({"apple", "banana", "cherry"})</code>	<code>frozenset</code>
<code>x = True</code>	<code>bool</code>
<code>x = b"Hello"</code>	<code>bytes</code>
<code>x = bytearray(5)</code>	<code>bytearray</code>
<code>x = memoryview(bytes(5))</code>	<code>memoryview</code>
<code>x = None</code>	<code>NoneType</code>

Tipos de dato y asignación



Python todas las variables tienen tipo. Existe un caso especial que es **None** (como ausencia de valor)

- Para determinar el tipo se utiliza la función **type(<var>)**

Datos Numéricos

x = 1 # int

y = 2.8 # float

z = 1j # complex

Notación científica

x = 35e3

y = 12E4

z = -87.7e100

#Enteros

x = 1

y = 35656222554887711

z = -3255522

#Complejos

x = 3+5j

y = 5j

z = -5j

#Reales

x = 1.10

y = 1.0

z = -35.59

Binarios(literales enteros)

x = 0b001

y = 0b010

z = x+y

Expresiones y operadores

- Las expresiones son una combinación válida de símbolos de operaciones, constantes literales, variables, paréntesis y llamadas a funciones.

Por ejemplo:

```
>>> a + (b + 3) + c**2  
>>> apellido + “,” + nombre  
>>> (a - 2) < (b + 4)
```

- Los operadores son símbolos que usamos para indicar operaciones sobre los datos u operandos. Tienen significado de acuerdo al tipo de dato sobre los que operan.

Operadores más comunes

- **Aritméticos:**

- Suma: +
- Resta: -
- Multiplicación: *
- División: /
- División entera: //
- Resto: %

```
>> edad = 20
>> nombre = "Luis"
>> a = 10.4 % 3.01
>> a = 10
>> b = -a
>> edad += 10
```

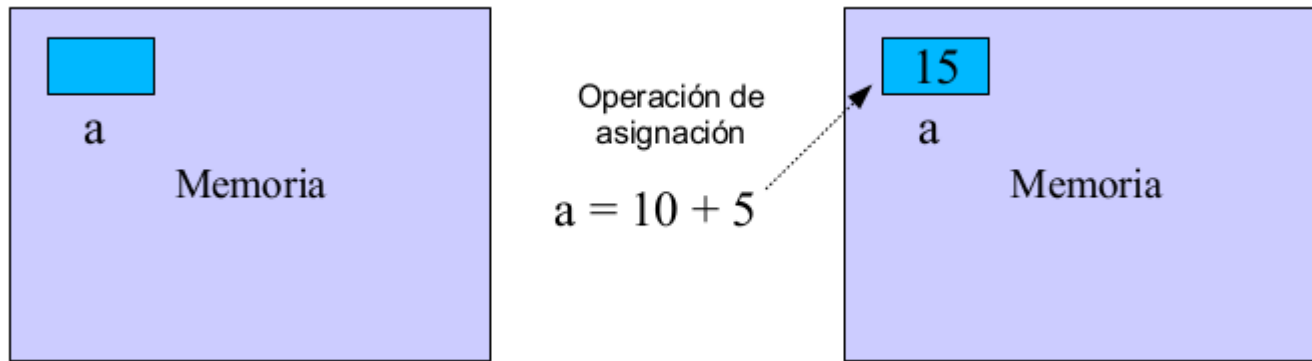
- **Asignación:**

- Operador de igualdad: = (puede emplearse de manera múltiple)
- Operadores +=, -=, *= y /= . Es lo mismo poner a+=1; que a=a+1; (y análogamente para los demás casos).

Operadores más comunes

Asignación

- Es la operación que permite acceder a cierta posición de memoria y cambiar el valor allí guardado.
- El operador de asignación está definido por el símbolo “=”. A la derecha debe aparecer una variable y a la izquierda una expresión.
- Ejemplo: $a = 10 + 5$. Indica que a la variable de nombre “a” se le asignará el resultado de evaluar $10 + 5$.



Operadores más comunes

- **Relacionales:**

- Igual que: ==
- Menor que: <
- Mayor que: >
- Menor o igual que: <=
- Mayor o igual que: >=
- Distinto que: !=

```
>> 5<4  
>> 10>6  
>> prueba = 100 > 2000 or 1 < 10  
>> 6==6
```

- **Lógicos:**

- Y: **and**
- O: **or**
- Operador negación lógica (**not**): Este operador devuelve **False** si se aplica a un valor **True**, y devuelve **True** si se aplica a un valor **False**. Su forma general es: **not** expresion

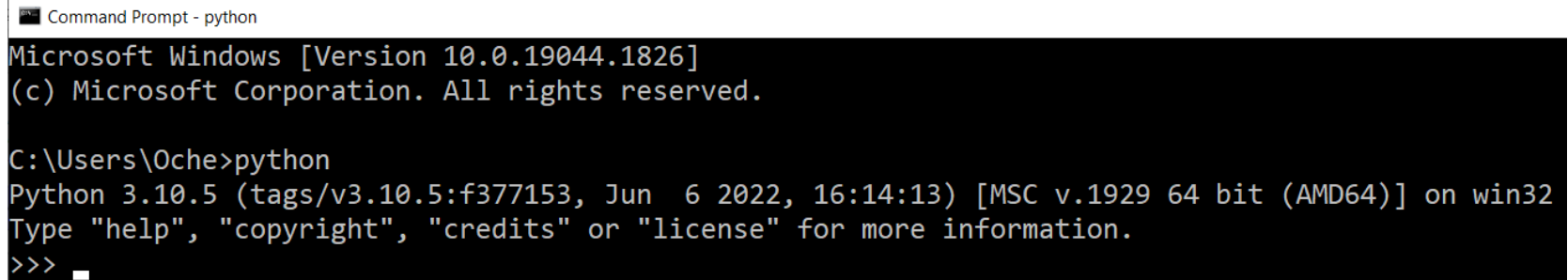
Precedencia de los operadores

Orden de Precedencia	Operador	Significado
1	**	Potenciación
2	- , + (unarios)	Cambio de signo - Identidad
3	*, / , %, //	Multiplicación – División (real) – Módulo (Resto) – División (entera)
4	+, -	Suma – Resta
5	<=, <, >, >=	Operadores relacionales de comparación
6	==, !=	Operadores relacionales de igualdad
7	not, or, and	Operadores lógicos
8	=, %=, /=, //=, -=, +=, *=, **=	Operadores de asignación

Todos los operadores en la misma línea tienen igual precedencia. Si aparecen en una expresión sin paréntesis, se evalúan de izquierda a derecha. El uso de paréntesis hace que las expresiones sean más legibles y altera el orden de precedencia. La abundancia de los mismos no tiene impacto negativo sobre la ejecución del programa.

La calculadora Python

- Python es un lenguaje de scripting (o guiones) interpretado (se ejecuta línea a línea)
- Cuando se ejecuta el intérprete se accede a un evaluador interactivo de expresiones Python.



```
Command Prompt - python
Microsoft Windows [Version 10.0.19044.1826]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Oche>python
Python 3.10.5 (tags/v3.10.5:f377153, Jun  6 2022, 16:14:13) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> █
```

Usando Python como calculadora

```
>>> 5+5
```

```
10
```

```
>>> 10.5-2*3
```

```
4.5
```

```
>>> 10**2    --> ** sirve para calcular potencias
```

```
100
```

```
>>> 17.0 // 3    --> // descarta decimales
```

```
5.0
```

```
>>> 17 % 3    --> % devuelve el resto de la división
```

```
2
```

```
>>> 5 * 3 + 2    --> precedencia de operadores
```

```
17
```

Tipos de datos – Uso de `type()`

Enteros:

```
>>> type(4)
<class 'int'>
```

Reales:

```
>>> type(1.5)
<class 'float'>
>>> 7/2
3.5
>>> float(12)
12.0
```

Complejos:

```
>>> type(3+2j)
<class 'complex'>
>>> (2+1j)**2
(3+4j)
```

Booleanos:

```
>>> type(True)
<class 'bool'>
```

Cadenas:

```
>>> type("Hola")
<class 'str'>
```


Evaluación de expresiones

Evaluar la siguiente expresión para $a=2$ y $b=5$:

$$3 * a - 4 * b / a ** 2 \longrightarrow$$

En la calculadora Python sería:

```
>>> a = 2
>>> b = 5
>>> 3 * a - 4 * b / a ** 2
1.0
```

Evaluar la expresión

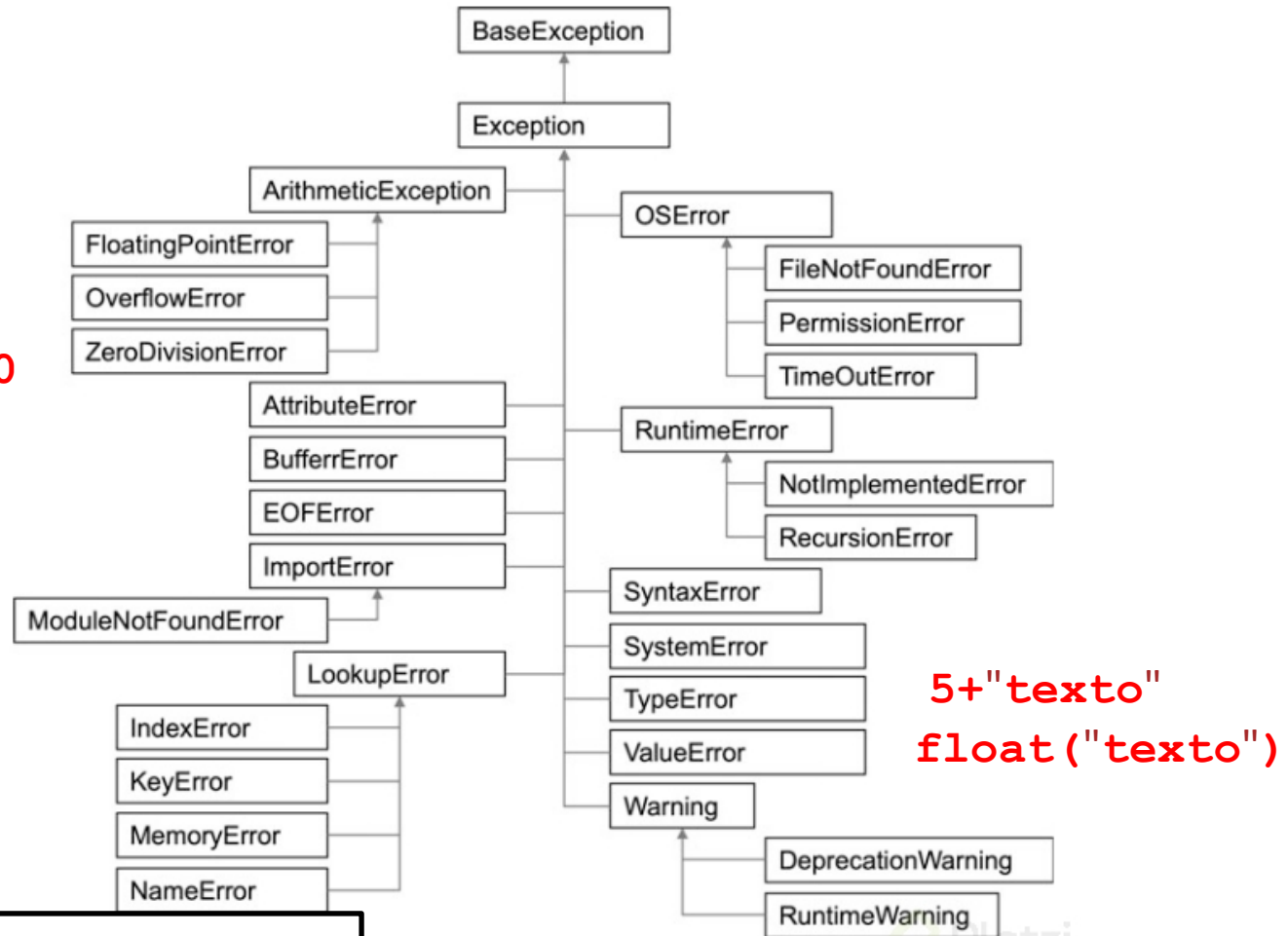
$$4 / 2 * 3 / 6 + 6 / 2 / 1 / 5 ** 2 / 4 * 2$$

Escribir las siguientes expresiones algebraicas como expresiones algorítmicas:

$\sqrt{b^2 - 4ac}$	$\frac{x^2 + y^2}{z^2}$	$\frac{3x + 2y}{2z}$	$\frac{a + b}{c - d}$
$4x^2 - 2x + 7$	$\frac{x + y - 3x}{x} \frac{5}{5}$	$\frac{a}{bc}$	xyz
$\frac{y_2 - y_1}{x_2 - x_1}$	$2 \pi r$	$\frac{4}{3} \pi r^3$	$(x_2 - x_1)^2 + (y_2 - y_1)^2$

Tipos de errores

5 % 0



5+"texto"
float("texto")

```
try:  
    print(x)  
except NameError:  
    print("La variable no está  
definida")
```

```
except:
```

Ejercicios de práctica

1) Escribir las siguientes expresiones algorítmicas como expresiones algebraicas

$$b^{**2} - 4 * a * c$$

$$3 * x^{**4} - 5 * x^{**3} + x * 12 - 17 \quad (b + d) / (c + 4)$$

$$(x^{**2} + y^{**2})^{** (1 / 2)}$$

2) Si el valor de $a=4$, $b=5$ y $c=1$, evaluar las siguientes expresiones:

$$b * a - b^{**2} / 4 * c$$

$$(a * b) / 3^{**2}$$

$$(((b + c) / 2 * a + 10) * 3 * b) - 6$$

3) Si el valor de a es 2, b es 3 y c es 2, evaluar la expresión:

$$a^{**b}^{**c}$$

Funciones internas

- Las operaciones que se requieren en los programas exigen en numerosas ocasiones, además de las operaciones aritméticas básicas ya tratadas, un número determinado de operadores especiales denominados *funciones internas* o incorporadas en el lenguaje.

Funciones disponibles en Python

- Python tiene una cantidad importante de funciones disponibles por defecto y puede extenderse mediante el uso de módulos específicos.
- Algunas funciones importantes (una lista completa puede obtener de <https://docs.python.org/3/library/functions.html>)
 `abs()`, `input()`, `print()`, `chr()`, `type()`, `max()`,
 `min()`, `range()`
- Y las funciones matemáticas en el módulo `math`:
 `cos()`, `sin()`, `ceil()`, `floor()`, `log()`, `pow()`,
 `trunc()`, `round()`, `factorial()`, `fmod()`.

Uso de funciones en Python

```
>>> abs(-10)      #función incorporada  
10
```

```
>>> import math    #utilizamos el módulo matemático
```

```
>>> math.pi        #constante pi, se coloca el nombre del  
3.14159265358979    # módulo antes
```

```
>>> math.cos(math.pi)  
-1.0
```

```
>>> math.e          #constante e  
2.718281828459
```

Entrada y salida de información

- Las operaciones de entrada permiten leer determinados valores y asignarlos a determinadas variables.
- Los datos de entrada se introducen al ordenador mediante dispositivos de entrada, siendo el teclado el utilizado por defecto.
- `a = int(input('Enter 1st number: '))`
- `b = int(input('Enter 2nd number: '))`
- La salida se realiza mediante dispositivos de salida, donde la pantalla es la utilizada por defecto.
- `print(f'Sum of {a} and {b} is {sum(a, b)}')`

Entrada de datos en Python

Python permite la entrada de datos mediante la función `input()`:

```
<var> = input(<msg>)
```

Donde `<var>` es la variable que recibirá la entrada y `<msg>` es un mensaje de información que aparecerá.

Ejemplo:

```
>>> a = input("Ingrese un valor")  
Ingrese un valor _
```

 Command Prompt - python

```
C:\Users\Oche>python  
Python 3.10.5 (tags/v3.10.5:f377153, Jun 6 2022, 16:14:13) [MSC v.1929 64 bit (AMD64)] on win32  
Type "help", "copyright", "credits" or "license" for more information.  
>>> a = input("Ingrese un valor: ")  
Ingrese un valor: 153  
>>> a  
'153'  
>>> _
```


Entrada de datos en Python

Como puede notarse, la función `input()` siempre retorna un valor de tipo cadena. Para nuestro ejemplo, podemos pasar a entero mediante `int()`

```
C:\Users\Oche>python
Python 3.10.5 (tags/v3.10.5:f377153,
Type "help", "copyright", "credits"
>>> a = input("Ingresa un valor: ")
Ingresa un valor: 153
>>> a
'153'
>>> type(a)
<class 'str'>
>>> a = int(a)
>>> a
153
>>> type(a)
<class 'int'>
>>>
```

Salida de datos en Python

Python puede mostrar los datos mediante la función `print()`:
`print(<expr1>, <expr2>, ..., <exprn>)`

Donde `<expr>` es una expresión

Ejemplo:

```
>>> a=10
>>> print("El valor ingresado es: ",a)
El valor ingresado es: 10
```

```
>>> a = 5
>>> b = 3
>>> print("El valor de",a,"+",b,"es:",a+b)
El valor de 5 + 3 es: 8
>>>
```

Salida de datos en Python

Por defecto, la función `print()` inserta un salto de línea al final (como consecuencia, los mensajes de dos `print()` consecutivos se mostrarán en líneas distintas). La opción `end` permite indicar la cadena al final del mensaje.

Por ejemplo, con las siguientes instrucciones:

```
print("Hola", end=" - ")  
print("Mundo")
```

Se mostrará:

```
Hola - Mundo
```

Salida de datos en Python

Python puede mostrar los datos mediante la función `print()`:
`print(<expr1>, <expr2>, ..., <exprn>)`

Donde `<expr>` es una expresión.

La opción `sep` permite indicar cuál será la cadena de separación entre las expresiones. Por ejemplo:

```
>>> a=5
>>> b=3
>>> c=1
>>> print(a,b,c,sep=" - ")
```

Tendrá como salida en pantalla:

```
5 - 3 - 1
```

Salida de datos con f-strings

Más info en: <https://docs.python.org/es/3/tutorial/inputoutput.html#fancier-output-formatting>

Python puede mostrar los datos mediante la función `print()`, pero utilizando un formato particular:

```
print(f'<cadena_con_formato>')
```

Donde en dicha cadena se especifica cada expresión a imprimir mediante `{<expr>}`. También es posible indicar un formato específico.

Ejemplos:

```
>>> a = 10
```

```
>>> print(f"El valor ingresado es: {a}")
```

```
El valor ingresado es: 10
```

```
>>> print(f"El cuadrado del valor ingresado es: {a**2}")
```

```
El cuadrado del valor ingresado es: 100
```

Salida de datos con f-strings

Más info en: <https://docs.python.org/es/3/tutorial/inputoutput.html#fancier-output-formatting>

Otro ejemplo se muestra a continuación:

```
>>> a=10
>>> b=5
>>> print(f"\nEl resultado de {a}+{b} es: {a+b}\n")

El resultado de 10+5 es: 15
```

Podemos limitar la cantidad de decimales que se muestran:

```
>>> import math
>>> print(f"El valor de pi es: {math.pi}")
El valor de pi es: 3.141592653589793
>>> print(f"El valor de pi con tres decimales es: {math.pi:.3f}")
El valor de pi con tres decimales es: 3.142
>>> _
```



Cantidad de decimales

Salida de datos con %

Más info en: <https://docs.python.org/es/3/tutorial/inputoutput.html#fancier-output-formatting>

Otra opción para mostrar datos con la función `print()` y el operador módulo (`%`), de la siguiente forma:

```
print("cadena_con_formato" % valores)
```

Donde las apariciones de `%` en la `cadena_con_formato` se reemplazan con cero o más elementos de `valores`.

Ejemplos:

```
>>> a = 10
>>> print("El valor ingresado es: %d" % a)
El valor ingresado es: 10
>>> print("El cuadrado del valor ingresado es: %d" % (a**2))
El cuadrado del valor ingresado es: 100
```

%d	-->	entero (int)
%f	-->	real (float)
%s	-->	cadena (string)

Salida de datos con %

Más info en: <https://docs.python.org/es/3/tutorial/inputoutput.html#fancier-output-formatting>

Otro ejemplo se muestra a continuación:

```
>>> a=5
>>> b=10
>>> print("El resultado de %d+%d es: %d" % (a,b,a+b))
El resultado de 5+10 es: 15
```

Podemos limitar la cantidad de decimales que se muestran:

```
>>> import math
>>> print("El valor de pi es: %f" % math.pi)
El valor de pi es: 3.141593
>>> print("El valor de pi es: %.3f" % math.pi)
El valor de pi es: 3.142
>>>
```



Cantidad de decimales

Ayuda interactiva en Python

Siempre puede obtener ayuda en la calculadora haciendo `help(<tema>)`

En el ejemplo mostrado se pide ayuda de la función `input()` y luego de `print()`

```
>>> help(input)
Help on built-in function input in module builtins:

input(prompt=None, /)
    Read a string from standard input.  The trailing newline is stripped.

    The prompt string, if given, is printed to standard output without a
    trailing newline before reading input.

    If the user hits EOF (*nix: Ctrl-D, Windows: Ctrl-Z+Return), raise EOFError.
    On *nix systems, readline is used if available.

>>> help(print)
Help on built-in function print in module builtins:

print(...)
    print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False)

    Prints the values to a stream, or to sys.stdout by default.
    Optional keyword arguments:
    file:  a file-like object (stream); defaults to the current sys.stdout.
    sep:   string inserted between values, default a space.
    end:   string appended after the last value, default a newline.
    flush: whether to forcibly flush the stream.
```

Depuración

The image shows the VS Code Python Debug Console interface with several annotations in green text and arrows:

- Breakpoint**: An arrow points to a yellow breakpoint icon on line 9 of `app.py`.
- Variables Panel**: The `VARIABLES` panel on the left shows the `Locals` scope with variables: `name: 'Sebastian'`, `today: datetime.date(2020, 3, 9)`, `day: 9`, `max: datetime.date(9999, 12, 31)`, `min: datetime.date(1, 1, 1)`, `month: 3`, `resolution: datetime.timedelta(days=1)`, and `year: 2020`.
- WATCH Panel**: The `WATCH` panel shows `name: 'Sebastian'`. An arrow points to it with the text: "You can define some variables that you want to watch all the time".
- CALL STACK Panel**: The `CALL STACK` panel shows the `home` function in `app.py` at line 9, which is `PAUSED ON BREAKPOINT`.
- DEBUG CONSOLE Panel**: The `DEBUG CONSOLE` panel at the bottom shows the output of the debug session, including the command to run the app and the Flask server output. An arrow points to it with the text: "You can execute any Python code in the DEBUG CONSOLE tab".
- Hover Panel**: A tooltip is visible over the `formatted_today` variable in the code, showing its value: `datetime.date(2020, 3, 9)`. An arrow points to it with the text: "If you hover your mouse over a variable, you will see its current value".

The code in the editor is as follows:

```
1 from flask import Flask
2 from datetime import date
3 app = Flask(__name__)
4
5
6 @app.route("/")
7 def home():
8     name = "Sebastian"
9     today = date.today()
10    formatted_today = today.strftime("%d/%m/%Y")
11    return f"Hello, {name}! Today is: {formatted_today}"
```

Cadenas o Texto(str)

- Es un tipo secuencial en Python (elementos accesibles por un índice)
- Es inmutable y está delimitado por comillas simples o dobles
- Cada elemento es de tipo UniCode (codificación de caracteres)
- Son comparables.

x = "Juan Perez"

y = 'Juan Perez'

z = """Lorem ipsum dolor sit amet,
consectetur adipiscing elit,
sed do eiusmod tempor incididunt
ut labore et dolore magna aliqua."""

z = " La Facultad de Ingeniería es la "mejor "
facultad "

z = " La Facultad de Ingeniería es la \"mejor \"
facultad "

x*5 # que imprimiría? ☺ tema de examen

Ejercicios

- Escribir un programa que evalúe la siguiente función

$$3 \cdot a - 4b/a^2$$

Dado: $a = 2$ y $b = 5$

- Escriba un programa que calcule la circunferencia de un círculo, dado su radio. Asuma $\pi = 3,14$.
- Dado un valor de Temperatura en grados Centígrados, imprima el valor en grados Fahrenheit:

$$^{\circ}\text{F} = \frac{9}{5} \cdot ^{\circ}\text{C} + 32$$

Ejercicios

- Escribir una variable `nombre_compañero` y asignar el nombre del compañero de al lado. Imprimir un saludo
- Escribe un programa que pida al usuario su nombre y lo salude por su nombre.
- Asignar el nombre y el apellido de tu compañera/o a las variables `nombre` y `apellido`. Luego, imprimir una frase, utilizando las variables. EJEMPLO: "Hola, Juan Pérez! Bienvenido a la segunda clase de Fundamentos de Programación"

Ejercicios

- Solicitar al usuario un número de 5 cifras e imprimir el dígito de la centena.
- Calcular el promedio de tres números ingresados por el usuario.
- Calcular la longitud de la hipotenusa de un triángulo rectángulo dado sus catetos.
- Dada una lista de palabras, utilizar `join()` para convertirlas en una oración completa, mostrando el resultado al usuario.

Ejercicios

- Pide al usuario que ingrese una palabra y determina si es un palíndromo (se lee igual de adelante hacia atrás que de atrás hacia adelante). El programa debe devolver True o False, sin usar estructuras de control condicional (if).
- Escribe un programa en Python que lea tres números (enteros o flotantes) desde la entrada del usuario y muestre la suma, el promedio, el número más alto y el más bajo.

Ejercicios

- Pide al usuario que introduzca un número de horas y conviértelo en segundos. Muestra el resultado.
- Escribe un programa que convierta dólares a euros. El usuario debe ingresar la cantidad en dólares y el programa debe mostrar la cantidad equivalente en euros. Considera una tasa de conversión fija.
- Solicita al usuario su peso en kilogramos y su altura en metros. Calcula y muestra su Índice de Masa Corporal (IMC).

$$IMC = \frac{Peso [Kg]}{Altura [m]^2}$$

Ejercicios

- Solicita al usuario el capital inicial, la tasa de interés anual (como un porcentaje) y el tiempo en años. Calcula y muestra el interés simple ganado.

$$I = P \cdot r \cdot t$$

Donde:

- I es el interés simple,
- P es el capital inicial (principal),
- r es la tasa de interés anual en forma decimal (por ejemplo, una tasa del 5% se expresaría como 0.05),
- t es el tiempo en años.

Gracias por la atención

