UNIVERSIDAD NACIONAL DE ASUNCIÓN
FACULTAD DE INGENIERÍA

Facultad de Ingeniería
Universidad Nacional de Asunción

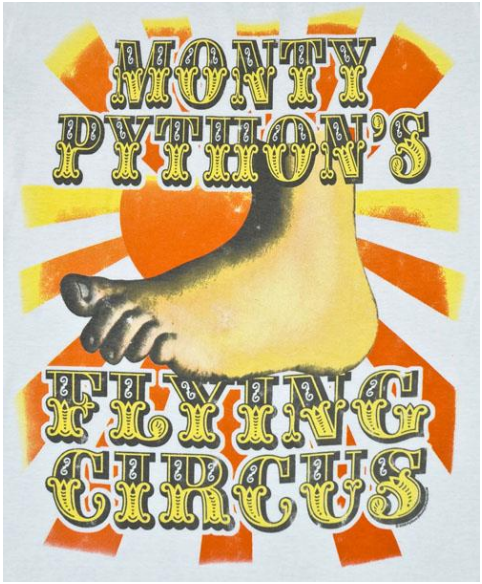
Cátedra de Fundamentos de Programación

Introducción a Python
Elementos básicos de un programa

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

Python

Guido van Rossum



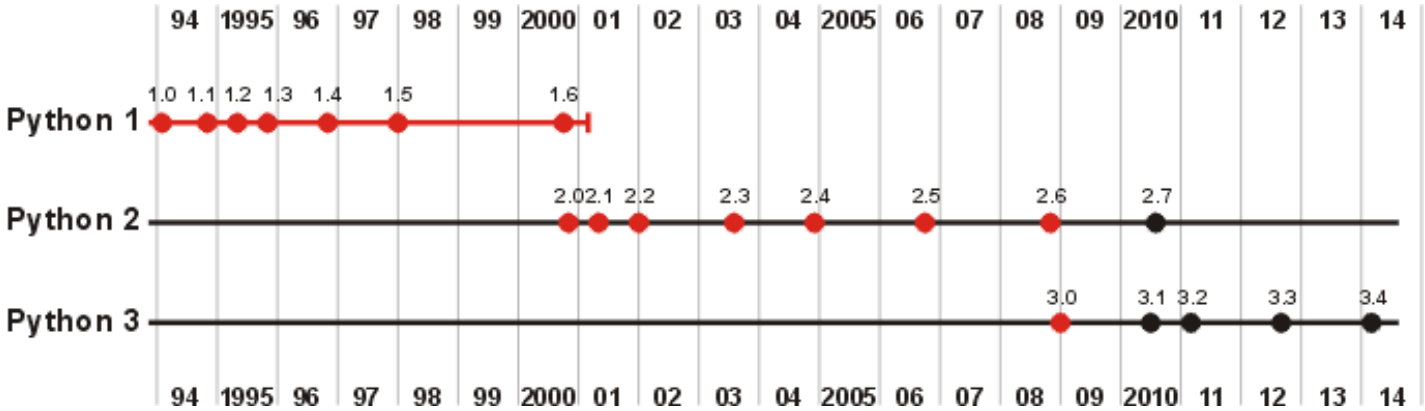
https://es.wikipedia.org/wiki/Historia_de_Python

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*.

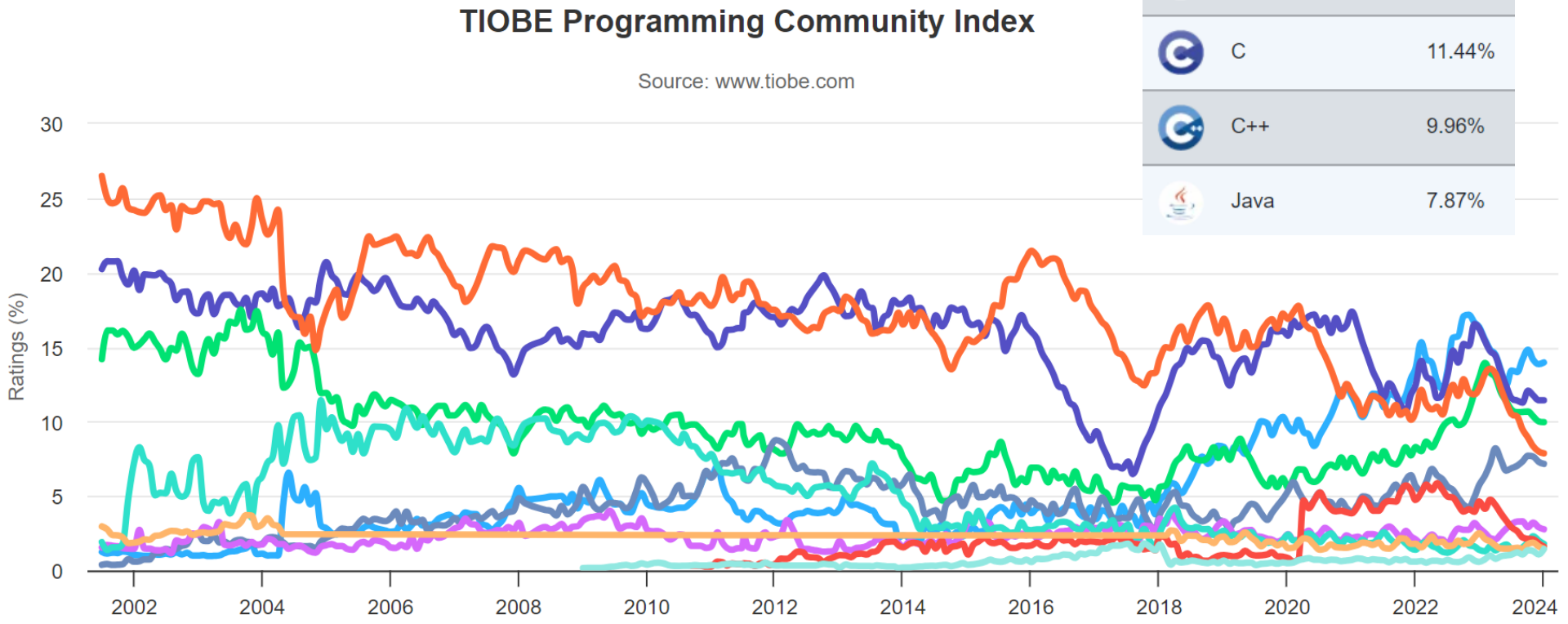


Adaptación de slides de "Python for Genomic Data Science", Coursera.
https://es.wikipedia.org/wiki/Historia_de_Python

¿Por qué aprender Python?

Su popularidad

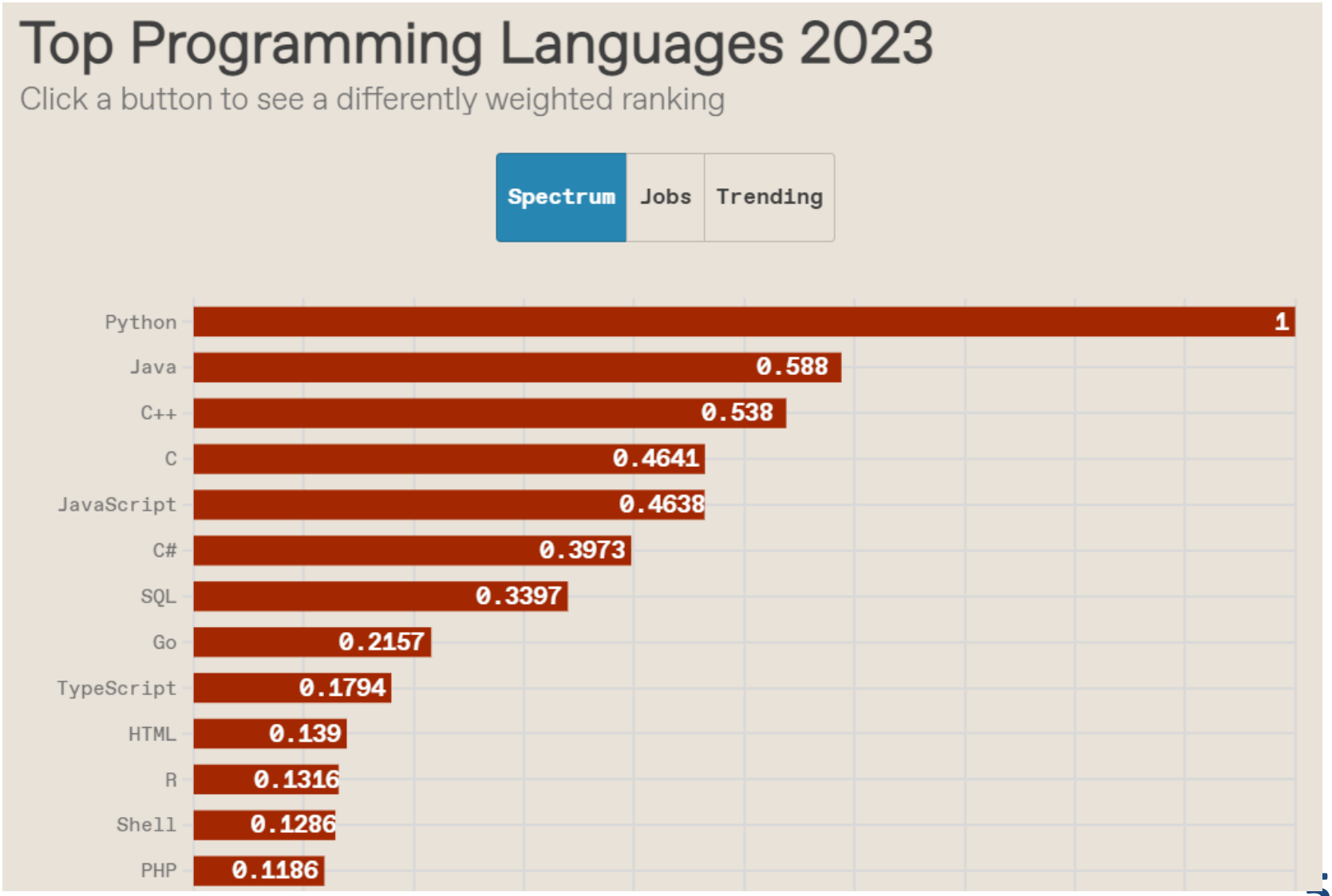
Programming Language	Ratings
Python	13.97%
C	11.44%
C++	9.96%
Java	7.87%



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)

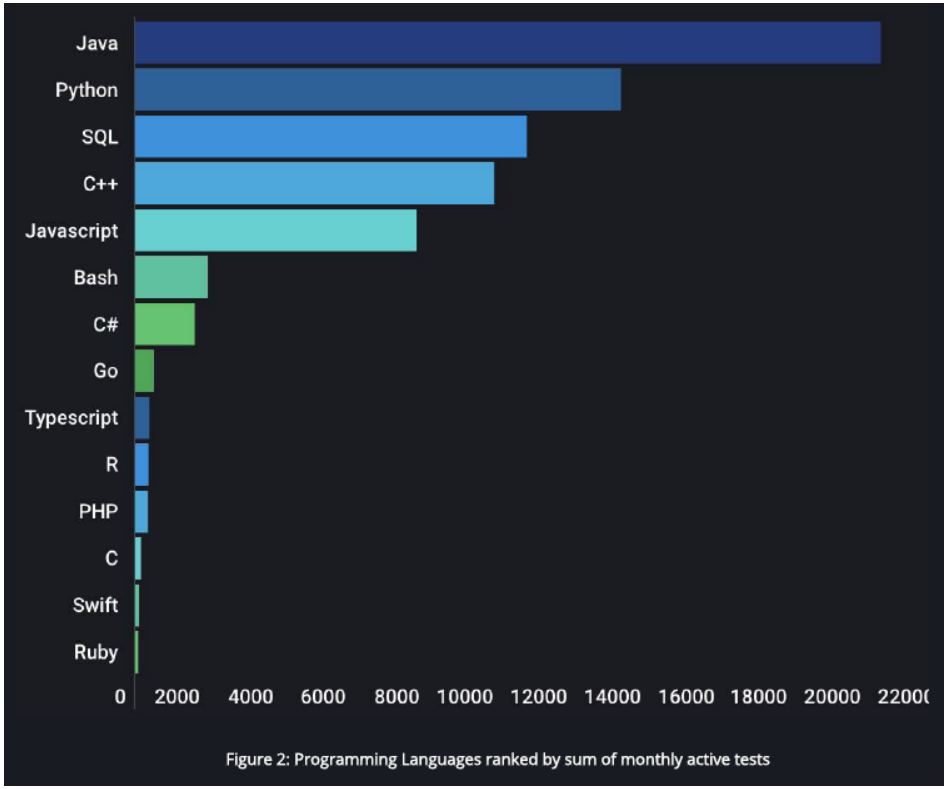


Figure 2: Programming Languages ranked by sum of monthly active tests

Lenguajes preferidos por los programadores

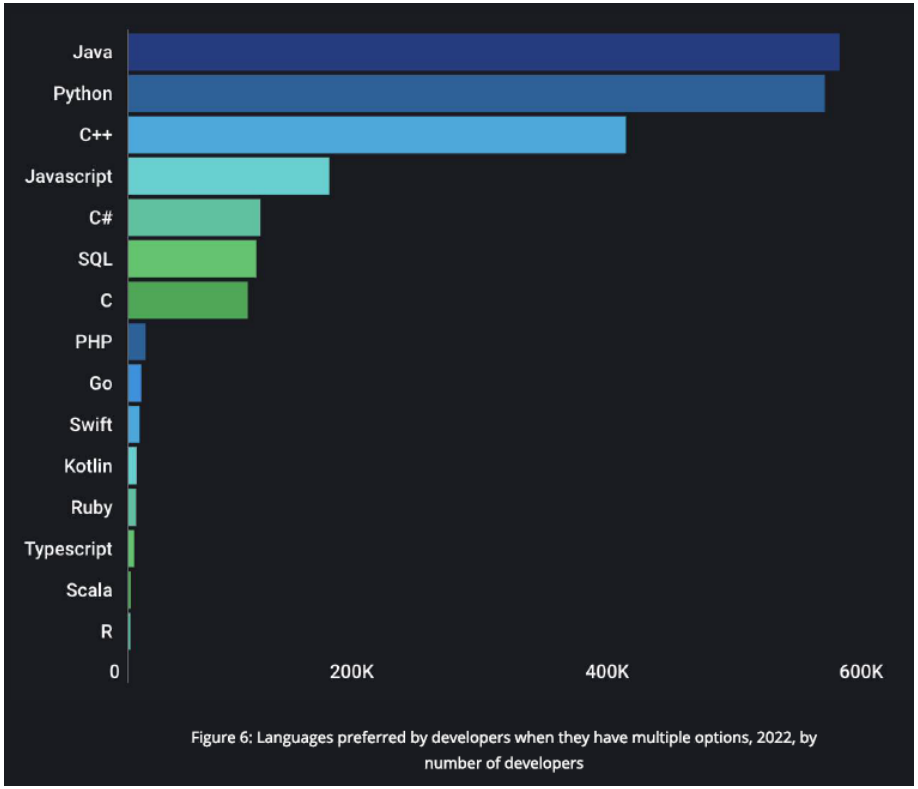


Figure 6: Languages preferred by developers when they have multiple options, 2022, by number of developers

https://www.hackerrank.com/research/developer-skills/2023

HackerRank Developer Skills Report 2023

Comunidad de 21 millones de miembros

Habilidades solicitadas

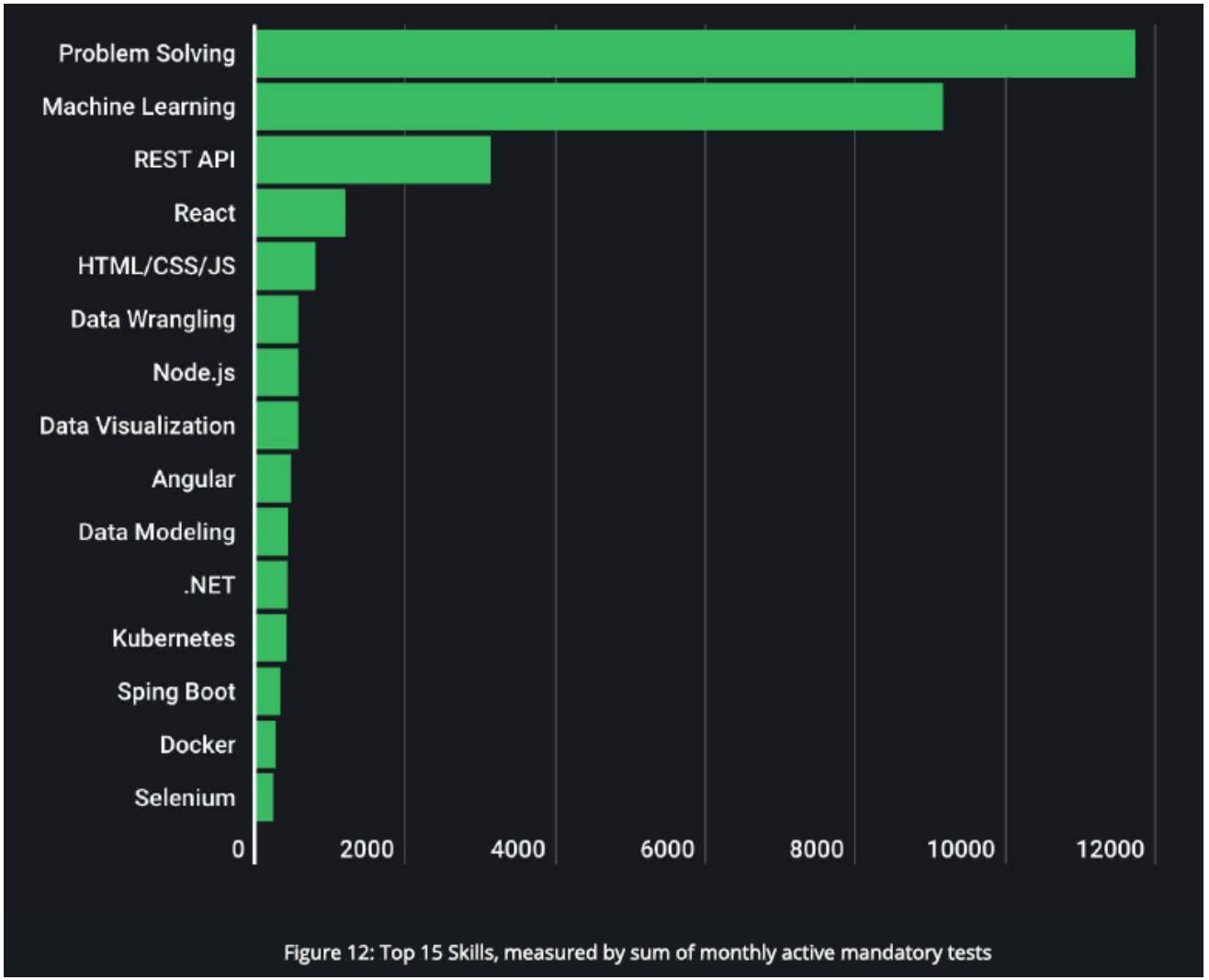
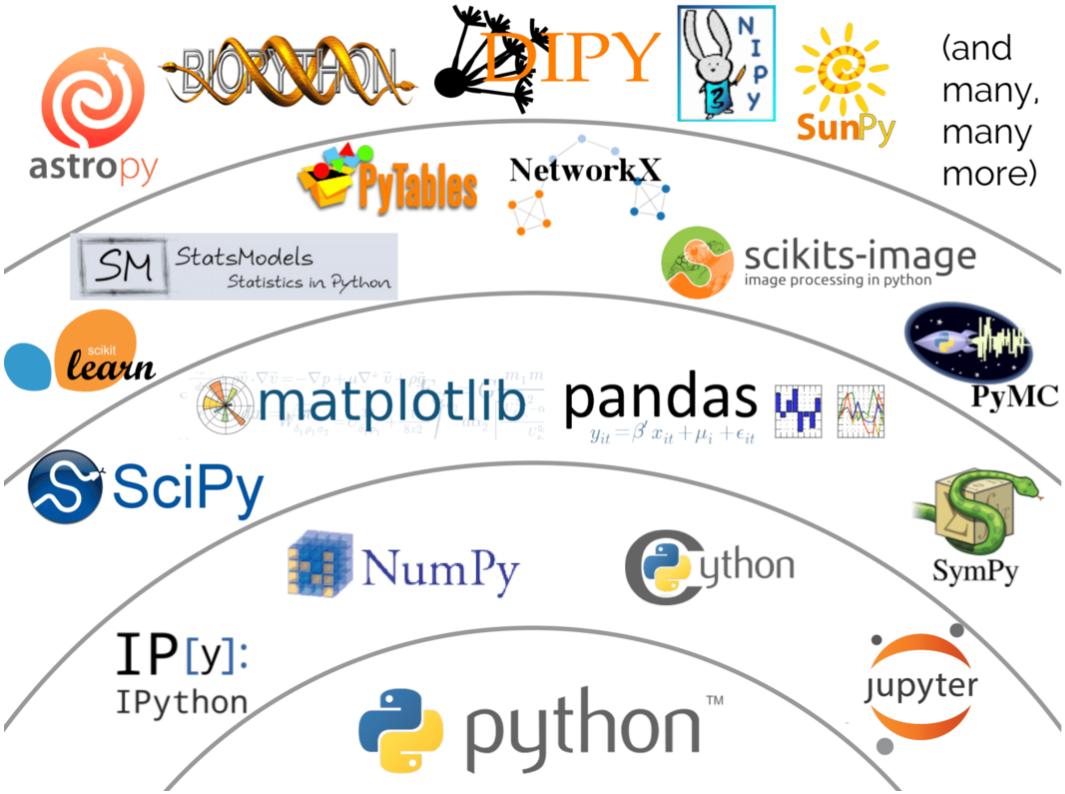


Figure 12: Top 15 Skills, measured by sum of monthly active mandatory tests

https://www.hackerrank.com/research/developer-skills/2023

Librerias 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!

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

```cpp
#include <iostream>

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

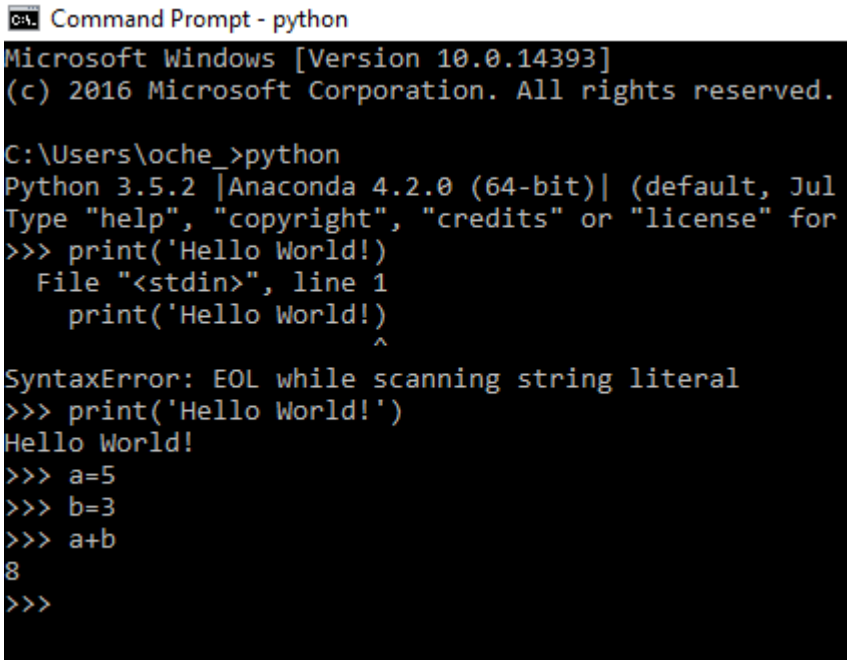

```java
public class HelloWorld {
	public static void main(String[] args) {
		System.out.println("Hello, World");
	}
}
```


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

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

Python - Características

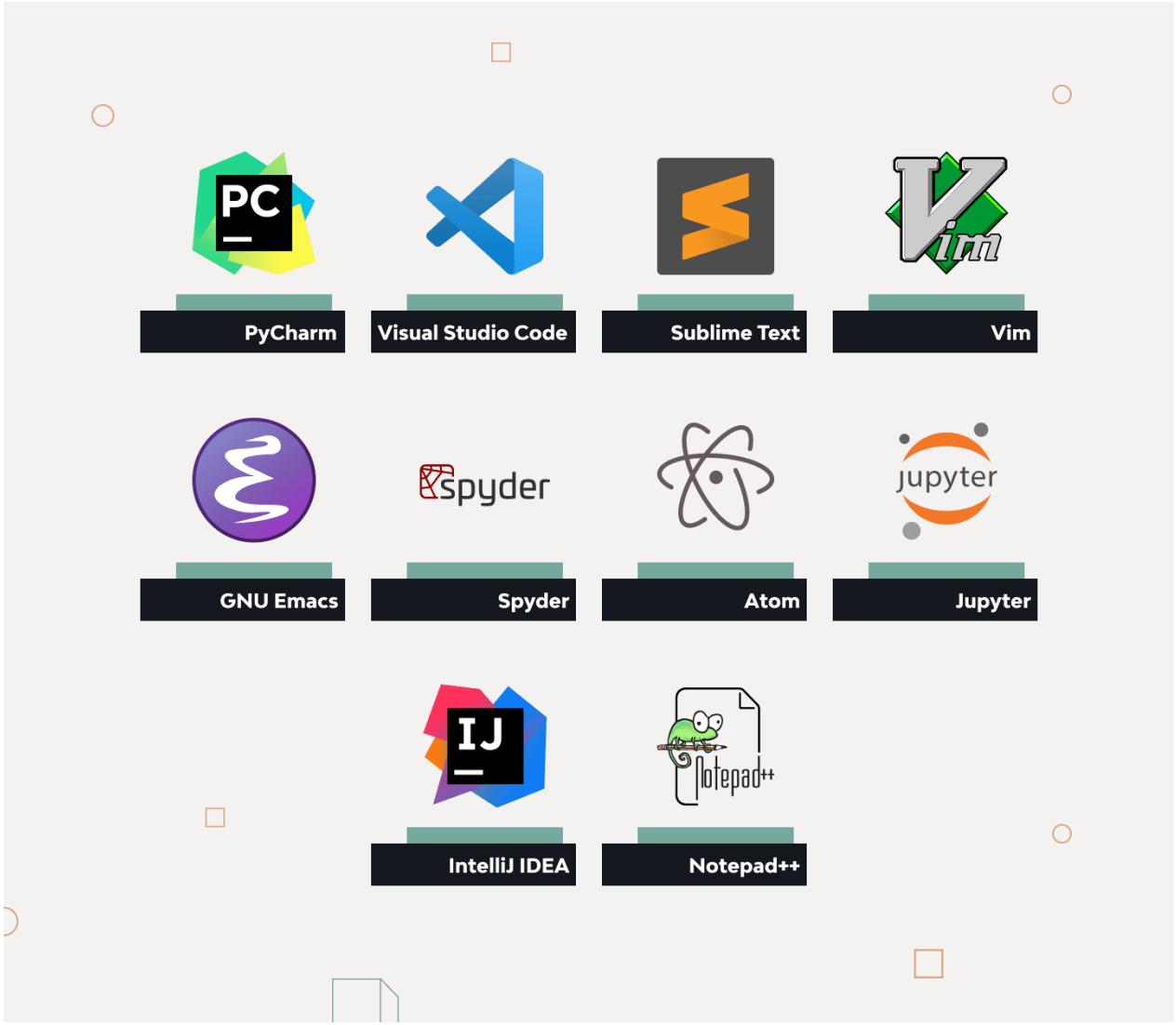
- Es simple de usar!
- Es interactivo
- Es portable
- Es extensible
- Tiene una gran librería estándar
- Es escalable



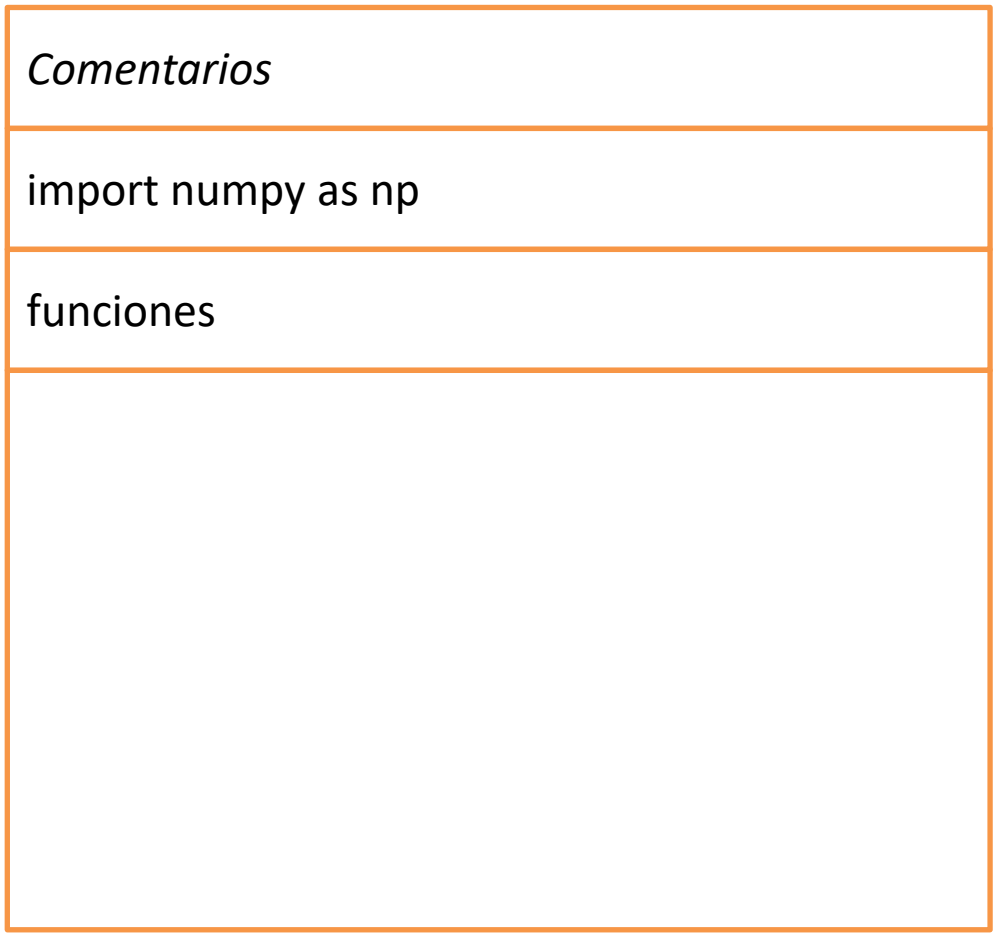
Python - Características

- 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




```
# 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, PI, 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.`

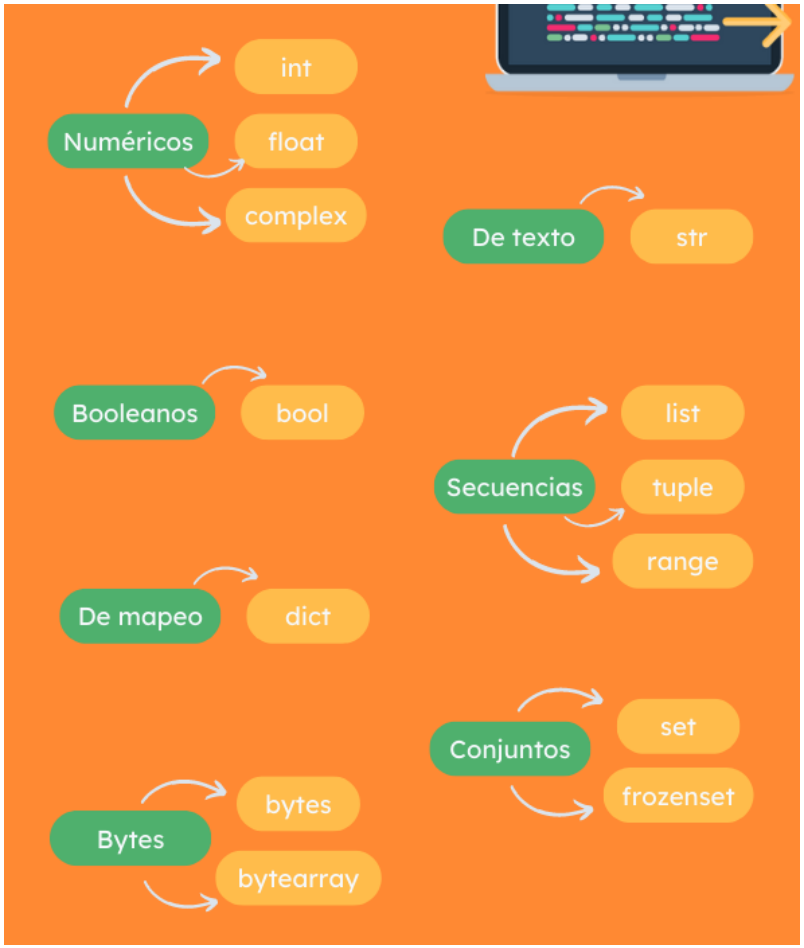
- **Lógico** (`bool` o `boolean`) : representa datos que pueden tomar dos valores posibles: verdadero (`True`) o falso (`False`).

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

Ejemplos de tipos de datos

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

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>)

https://docs.python.org/es/3/reference/datamodel.html

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: %

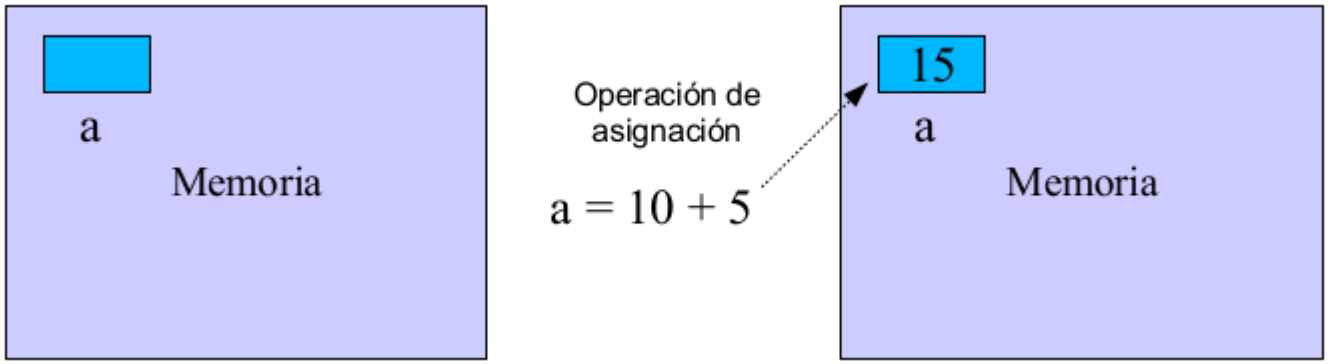
```
>> edad = 20
>> nombre = “Luis”
>> a = 10.4 % 3.01
>> a = 10
>> b = -a
>> edad += 10
```

- **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

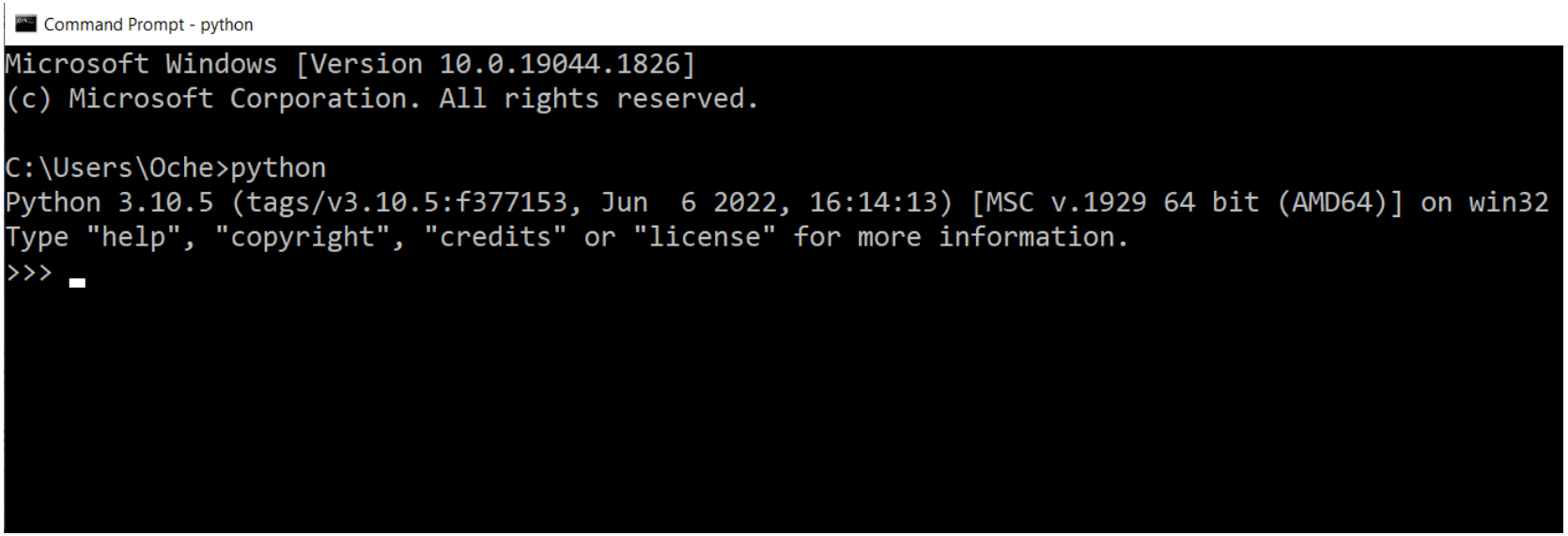
Orden de Precedencia	Operador	Significado
1	`**`	Potenciación
2	`-,+` (unarios)	Cambio de signo - Identidad
3	`*, /, %,//`	Multiplicación – División (real) – Módulo (Resto) – División (entera)
4	`+,-`	Suma – Resta
5	`<=, <, >, >=`	Operadores relacionales de comparación
6	`==, !=`	Operadores relacionales 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.



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
```

En la calculadora Python sería:

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

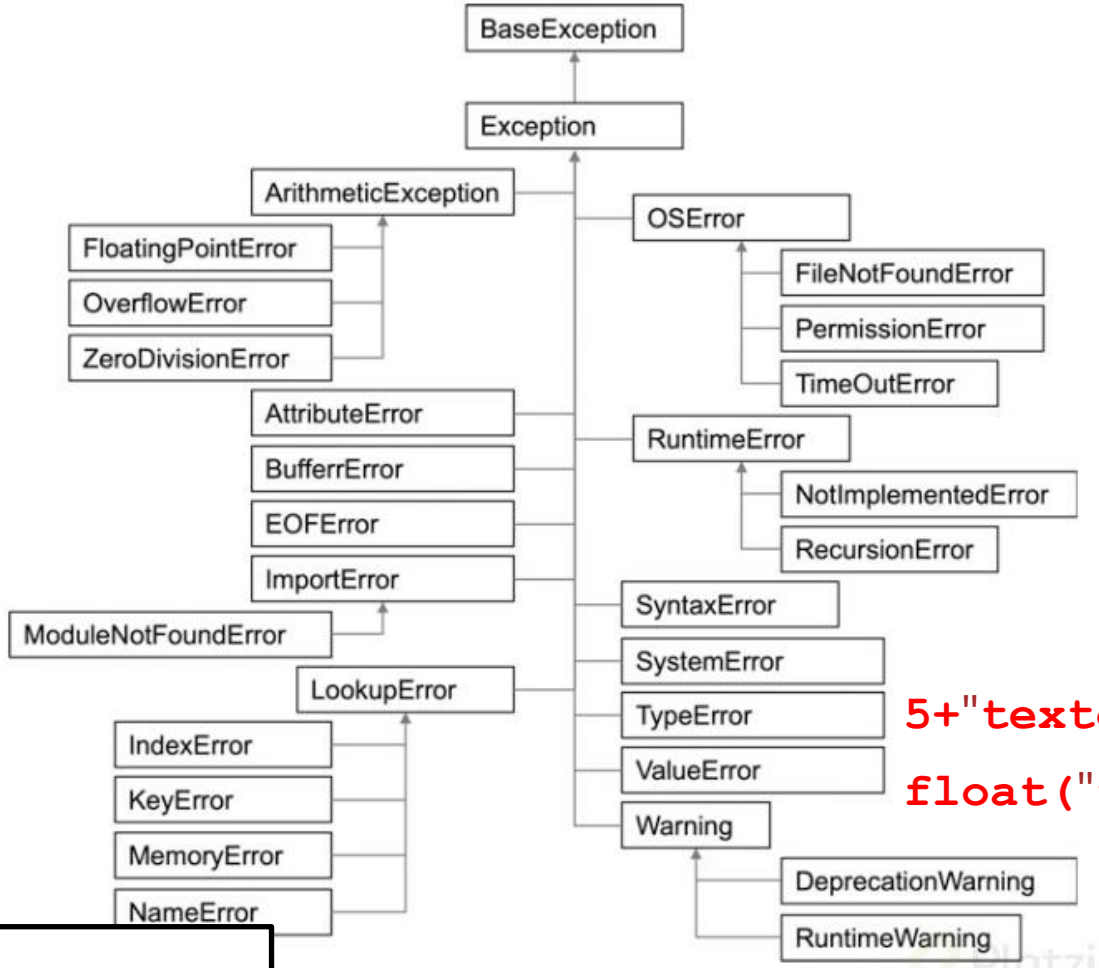
Evaluar la expresión

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

Escribir las siguientes expresiones algebraicas como expresiones algorítmicas:

$\sqrt{b^2 - 4ac}$	$\frac{x^2 + y^2}{z^2}$	$\frac{3x + 2y}{2z}$	$\frac{a + b}{c - d}$
$4x^2 - 2x + 7$	$\frac{x + y - \frac{3x}{5}}{x}$	$\frac{a}{bc}$	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:
  print("Otro tipo de error")
```

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  (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

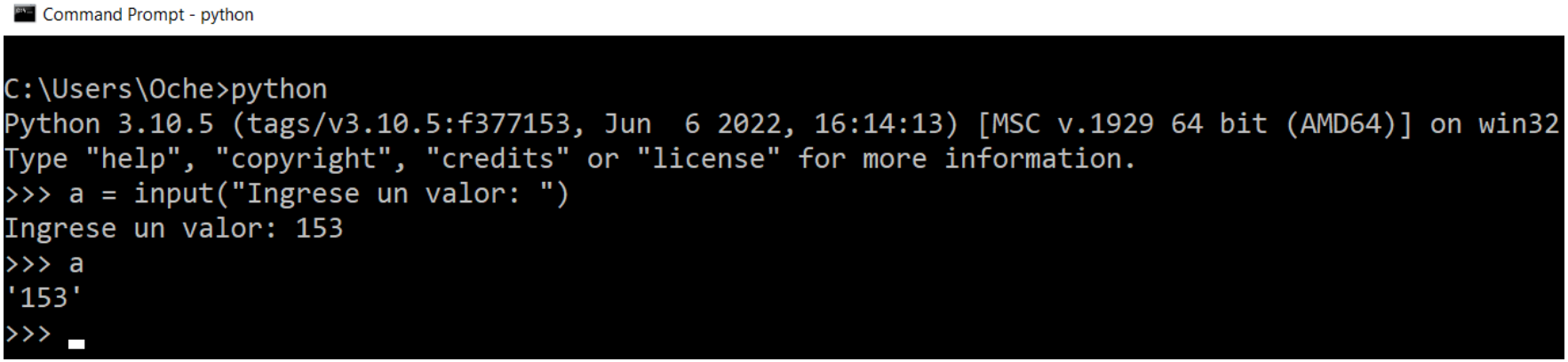
Python permite la entrada de datos mediante la función `input()`:

```
<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 __
```



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("Ingrese un valor: ")
Ingrese 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 %

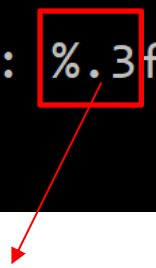
Más info en: https://docs.python.org/es/3/tutorial/inputoutput.html#fancier-output-formatting

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

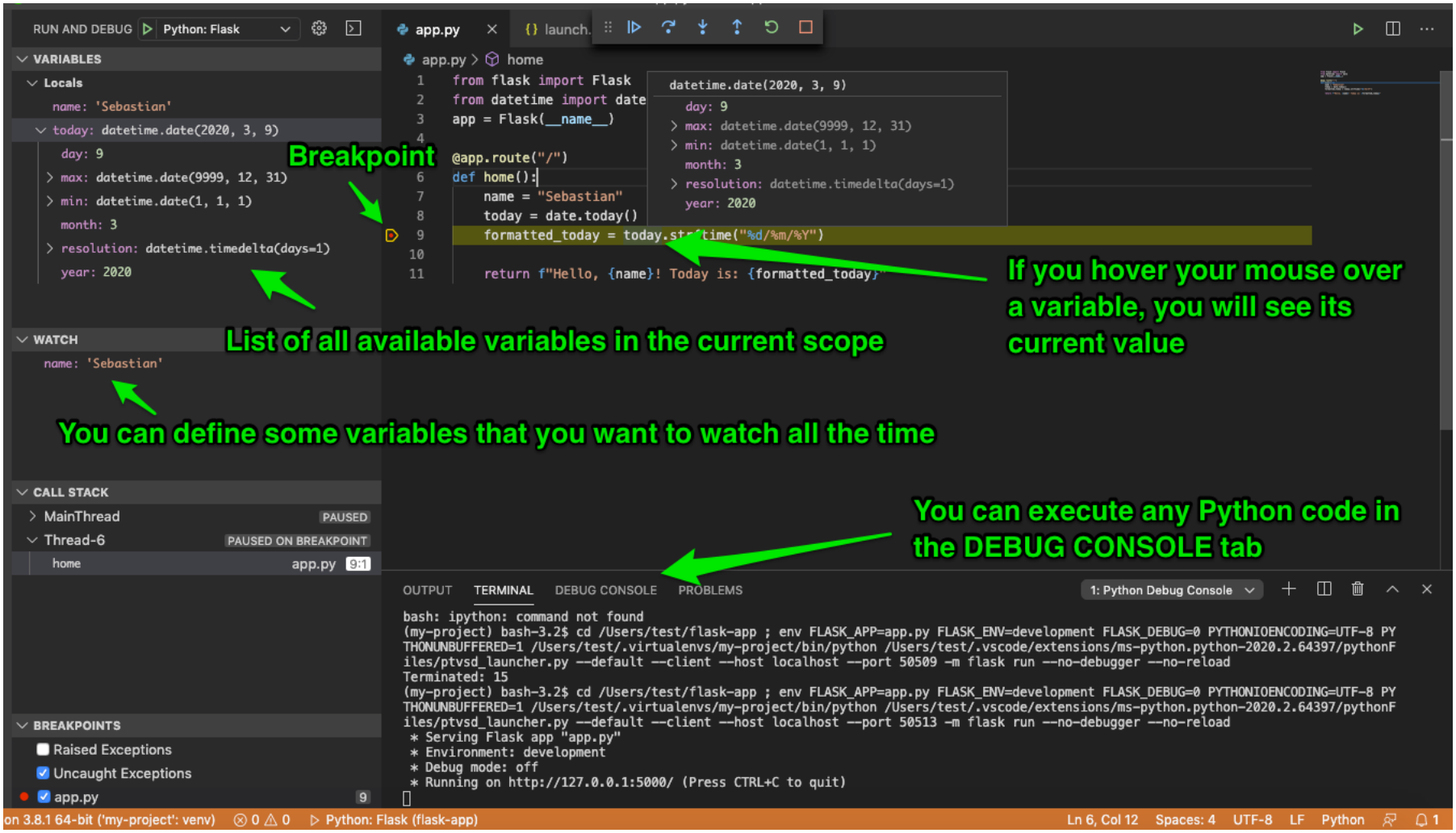
Ayuda interactiva en Python

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

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


RUN AND DEBUG
Python: Flask
VARIABLES
Locals
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)
year: 2020
WATCH
name: 'Sebastian'
CALL STACK
MainThread PAUSED
Thread-6 PAUSED ON BREAKPOINT
home app.py 9:1
BREAKPOINTS
Raised Exceptions
Uncaught Exceptions
app.py 9
app.py
launch.
app.py > home
1 from flask import Flask
2 from datetime import date
3 app = Flask(__name__)
4
5 @app.route("/")
6 def home():
7 name = "Sebastian"
8 today = date.today()
9 formatted_today = today.strftime("%d/%m/%Y")
10
11 return f"Hello, {name}! Today is: {formatted_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)
year: 2020
Breakpoint
List of all available variables in the current scope
If you hover your mouse over a variable, you will see its current value
You can define some variables that you want to watch all the time
You can execute any Python code in the DEBUG CONSOLE tab
OUTPUT TERMINAL DEBUG CONSOLE PROBLEMS
1: Python Debug Console
bash: ipython: command not found
(my-project) bash-3.2$ cd /Users/test/flask-app ; env FLASK_APP=app.py FLASK_ENV=development FLASK_DEBUG=0 PYTHONIOENCODING=UTF-8 PYTHONUNBUFFERED=1 /Users/test/.virtualenvs/my-project/bin/python /Users/test/.vscode/extensions/ms-python.python-2020.2.64397/pythonFiles/ptvsd_launcher.py --default --client --host localhost --port 50509 -m flask run --no-debugger --no-reload
Terminated: 15
(my-project) bash-3.2$ cd /Users/test/flask-app ; env FLASK_APP=app.py FLASK_ENV=development FLASK_DEBUG=0 PYTHONIOENCODING=UTF-8 PYTHONUNBUFFERED=1 /Users/test/.virtualenvs/my-project/bin/python /Users/test/.vscode/extensions/ms-python.python-2020.2.64397/pythonFiles/ptvsd_launcher.py --default --client --host localhost --port 50513 -m flask run --no-debugger --no-reload
* Serving Flask app "app.py"
* Environment: development
* Debug mode: off
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
on 3.8.1 64-bit ('my-project': venv)
Python: Flask (flask-app)
Ln 6, Col 12 Spaces: 4 UTF-8 LF Python

Cadenas o Texto(str)

- Es un tipo secuencialen Python (elementos accesibles por un índice)
- Es inmutable y esta delimitado por comillassimples 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:

$$°\mathrm{F} = \frac{9}{5} \cdot °\mathrm{C} + 32$$

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 digito de la centena.
- Calcula el promedio de tres números ingresados por el usuario.
- Calcula la longitud de la hipotenusa de un triángulo rectángulo dado sus catetos.
- Dada una lista de palabras, utiliza 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.

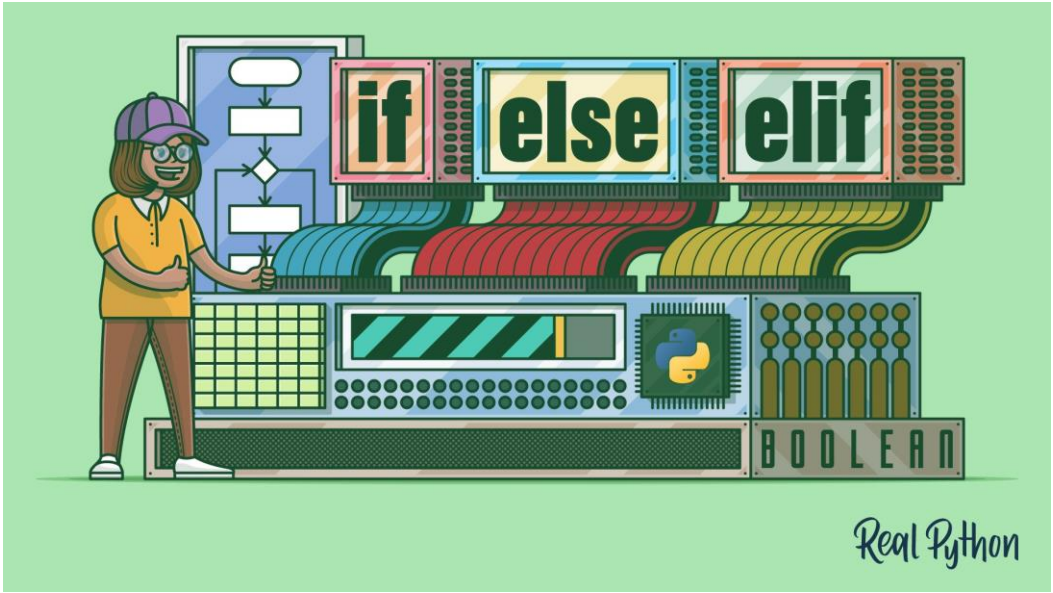
Gracias por la atención

UNIVERSIDAD NACIONAL DE ASUNCIÓN
FACULTAD DE INGENIERÍA

Cursos Básicos
Primer Ciclo 2026

Fundamentos de Programación

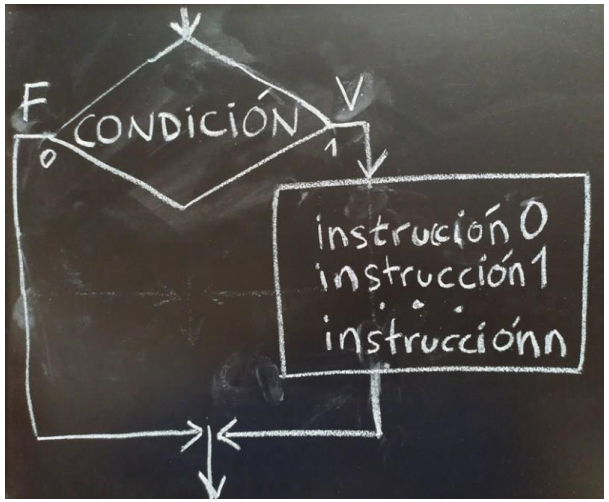
Estructuras de Selección



https://realpython.com/python-conditional-statements/

¿Qué veremos hoy?

- Necesidad de estructuras de selección.
- Elementos de la programación estructurada.
- Estructuras de selección:
  - Representación
  - Operadores lógicos y relacionales.
  - Expresiones booleanas.
  - Estructuras de selección simples, alternativas y anidadas.



Necesidad de estructuras selectivas

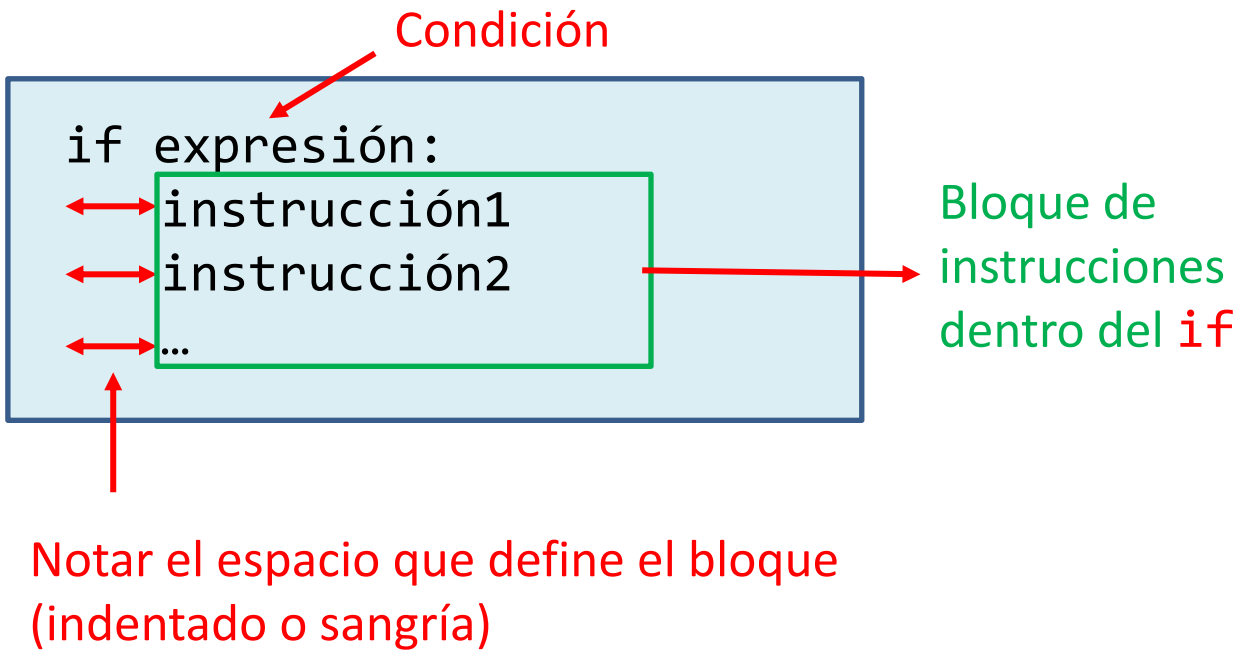
Problema: mostrar la calificación de un alumno a partir del puntaje obtenido en el examen, de acuerdo a la siguiente tabla:

Puntaje	Calificación
Entre 90 y 100	5
Entre 80 y 89	4
Entre 70 y 79	3
Entre 60 y 69	2
Menos de 60	1

Uso de sangría o *indentación* en Python

A diferencia de otros lenguajes de programación, Python no emplea llaves { } para delimitar *bloques* de código. En cambio, emplea la **sangría** o ***indentación***; que consiste en agregar espacios o tabulaciones para mover instrucciones a la derecha.

Un bloque de código dentro de una estructura siempre estará a su derecha.



Expresiones booleanas

- La **condición** en la estructura `if` es llamada una **expresión Booleana**.

- Las expresiones booleanas pueden ser verdaderas (**True**) o falsas (**False**).

```
>>> 0<1
True
>>> 5%2==0
False
```

- Estas expresiones son formadas con la ayuda de operadores **relacionales** y **lógicos**. Más adelante veremos otros operadores, como los de **identidad** y **membresía**.

Expresión dentro del `if`

Expresiones relacionales

Son expresiones que permiten determinar la relación que existe entre dos operandos a través de un operador relacional. El mismo permite comparar los valores de dos operandos de igual naturaleza. El resultado de evaluar una expresión relacional siempre es verdadero (`True`) o falso (`False`).

Operadores relacionales

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

Ejemplos:

- `10 > 20` ==> `False`
- `20 == 10` ==> `False`
- `5 == 5` ==> `True`
- `'a' == 'a'` ==> `True`
- `1!=2` ==> `True`
- `3+5 <= 7` ==> `False`

Estructura selectiva if

Ejemplo 1: (con una expresión relacional simple)

Imprimir el mensaje “Es un nro. negativo” cuando un número entero leído por teclado es negativo.

```
n = int(input("Ingrese un numero: "))
if n<0:
    print("Es un numero negativo")
```

```
========== RESTART:
Ingrese un numero: -6
Es un numero negativo
>>>
========== RESTART:
Ingrese un numero: 3
>>>
```

Expresión dentro del `if`

Obs: Tener en cuenta el orden de precedencia de los operadores!!

Expresiones lógicas

Las expresiones lógicas consisten en variables booleanas, constantes booleanas (`True` o `False` en Python), expresiones relacionales y operadores booleanos (`and`, `or`, `not`).

Operador	Descripción
and	**True** si ambas condiciones son verdaderas
or	**True** si al menos una de las condiciones es verdadera
not	**True** si la condición es falsa

and	or	not
V and V ==> V	V or V ==> V	not V ==> F
F and V ==> F	F or V ==> V	not F ==> V
V and F ==> F	V or F ==> V	
F and F ==> F	F or F ==> F	

Estructura selectiva `if`

Ejemplo 2: (con una expresión lógica)

Imprimir el mensaje “Es un nro. mayor a 100 y es par” cuando un número leído por teclado es mayor a 100 y es par.

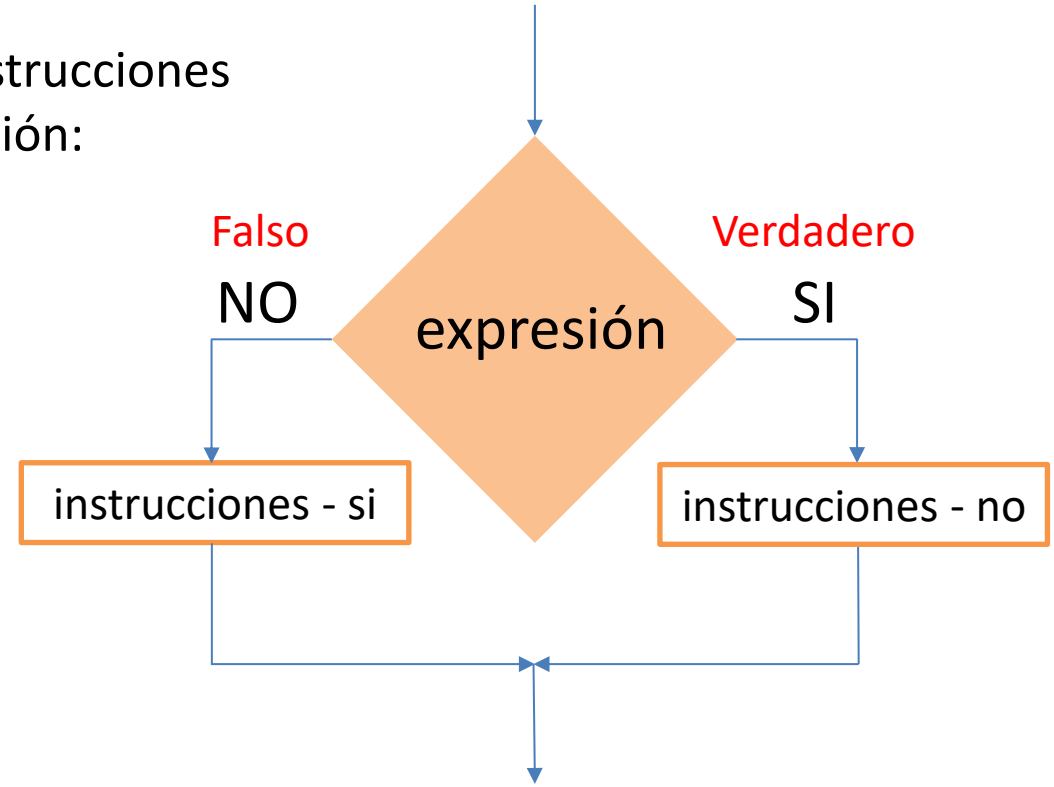
```
n = int(input("Ingrese un numero: "))
if n>100 and n%2==0:
    print("Es un nro. mayor a 100 y es par")
```

```
========== RESTART: C:
Ingrese un numero: 102
Es un nro. mayor a 100 y es par
>>>
========== RESTART: C:
Ingrese un numero: 103
>>>
========== RESTART: C:
Ingrese un numero: 96
>>>
```

Estructura selectiva `if-else`

En el caso de querer ejecutar instrucciones para cada resultado de la condición:

```
if expresión:
    instrucciones - si
else:
    instrucciones - no
```



En esta segunda forma tenemos dos posibilidades: si al evaluar la expresión el resultado es verdadero se ejecuta las `instrucciones-si`, pero si el resultado es falso se ejecuta las `instrucciones-no`. ***En cualquier caso, se ejecuta sólo uno de los dos grupos de instrucciones.***

Estructura selectiva `if-else`

Ejemplo 3:

Imprimir el mensaje “Es un nro. negativo” cuando un número entero leído por teclado es negativo, y “Es un nro. mayor o igual a cero” cuando no.

```
n = int(input("Ingrese un numero: "))
if n<0:
    print("Es un nro. negativo")
else:
    print("Es un nro. mayor o igual a cero")
```

```
========== RESTART: C:
Ingrese un numero: 5
Es un nro. mayor o igual a cero
>>>
========== RESTART: C:
Ingrese un numero: -2
Es un nro. negativo
>>>
```

Estructura selectiva anidada

Una estructura `if` o `if-else` puede estar dentro de otro `if` o `else`. En este caso, se tiene una estructura selectiva *anidada*. De esta manera, aumenta el número de caminos posibles.

Ejemplo 4-a:

Escribir un programa en Python que indique en pantalla si el número ingresado por teclado es negativo, positivo o neutro (cero).

```
n = int(input("Ingrese un numero: "))
if n<0:
    print("Es un nro. negativo")
else:
    if n>0:
        print("Es un nro. positivo")
    else:
        print("El nro. es cero")
```

Estructura selectiva anidada

Una estructura `if` o `if-else` puede estar dentro de otro `if` o `else`. En este caso, se tiene una estructura selectiva *anidada*. De esta manera, aumenta el número de caminos posibles.

Ejemplo 4-b:

Escribir un programa en Python que indique en pantalla si el número ingresado por teclado es negativo, positivo o neutro (cero).

```
n = int(input("Ingrese un numero: "))
if n>=0:
    if n>0:
        print("Es un nro. positivo")
    else:
        print("El nro. es cero")
else:
    print("Es un nro. negativo")
```

Estructura selectiva anidada

Ejemplo 5:

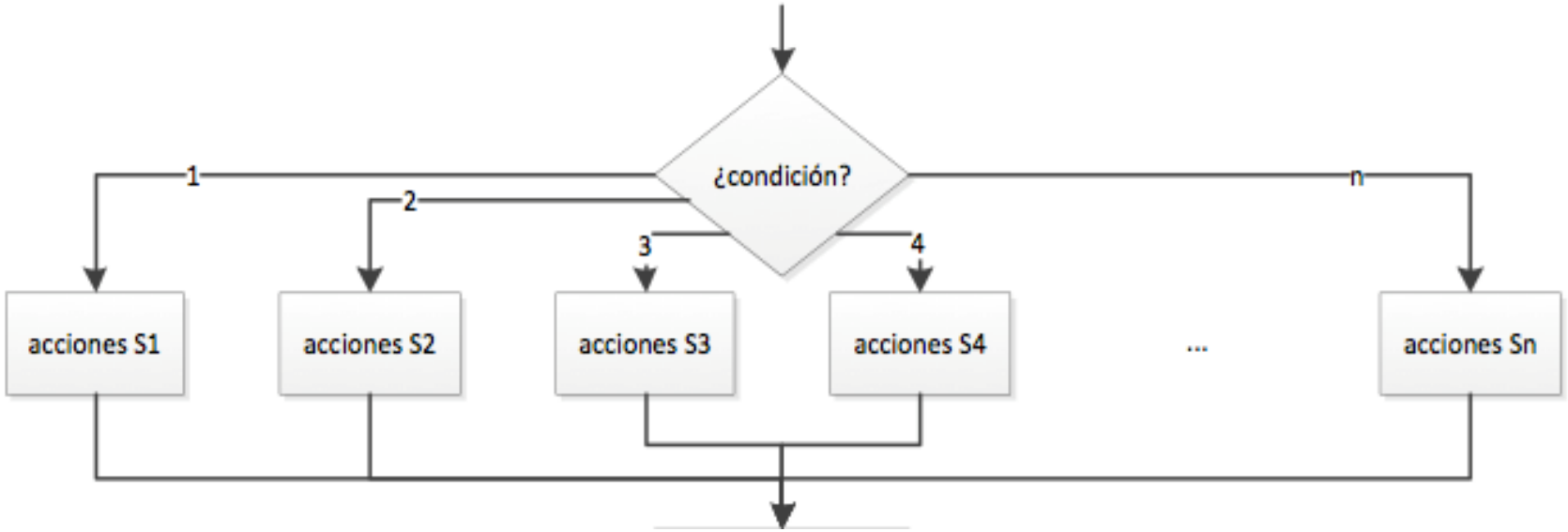
Escribir un programa en Python que lea 3 números por teclado e indique el valor central.

```
a = int(input("Primer numero: "))
b = int(input("Segundo numero: "))
c = int(input("Tercer numero: "))
if a<=b:
    if b<=c:
        central=b
    else: #b>c
        if a<=c:
            central=c
        else:
            central=a
else: #a>b
    if a<=c:
        central=a
    else: #a>c
        if b<=c:
            central=c
        else:
            central=b
print("Valor central:",central)
```

Estructura selectiva múltiple

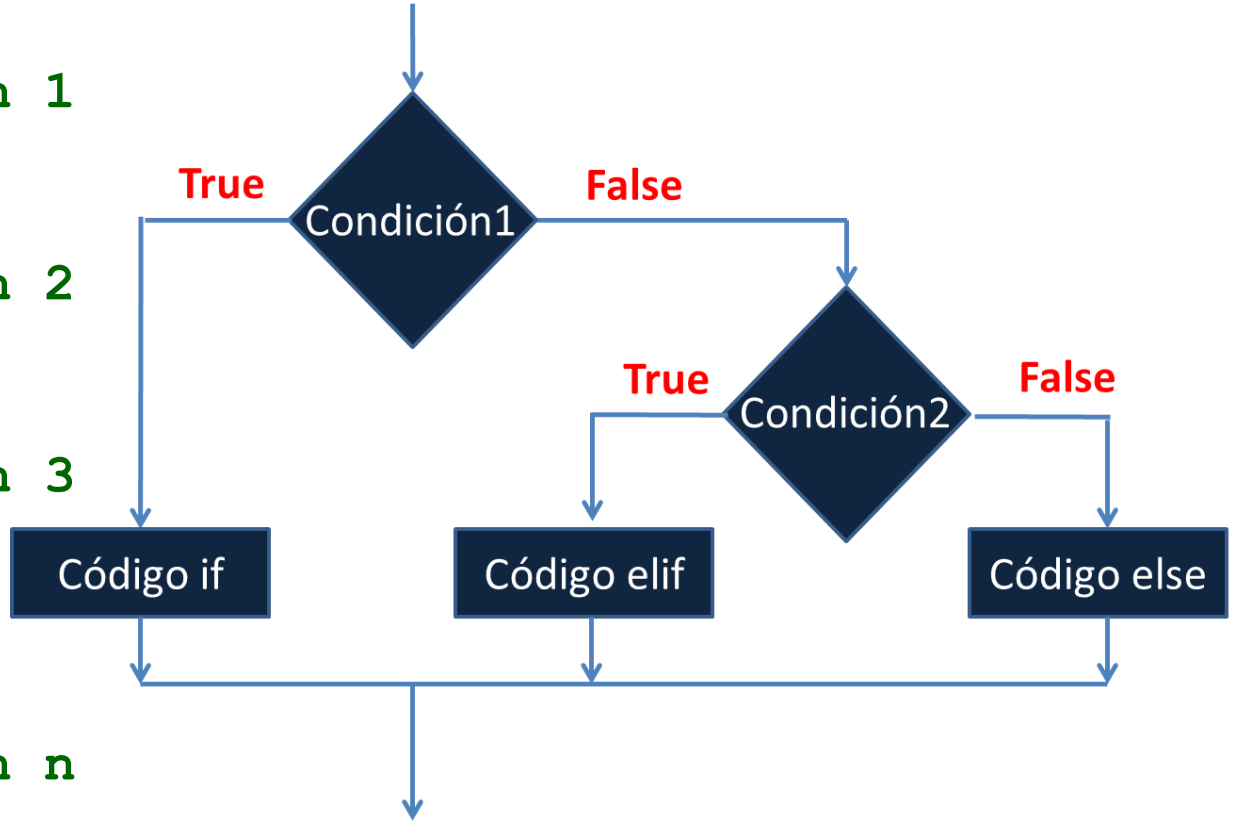
La estructura de decisión múltiple evaluará una expresión que podrá conducir a n caminos distintos, 1, 2, 3, …, n . Según se elija una de estas posibilidades en la condición, se realizará una de las n acciones.

Esta estructura también puede ser resuelta con estructuras simples o dobles anidadas. Sin embargo, si el número de alternativas es grande puede plantear serios problemas de escritura del algoritmo y naturalmente de legibilidad. Para facilitar esto, Python proporciona `elif`.



Estructura selectiva múltiple

```
if condición 1:
    realizar acción 1
elif condición 2:
    realizar acción 2
elif condición 3:
    realizar acción 3
...
elif condición n:
    realizar acción n
else: # ninguna de las anteriores condiciones se cumple
    hacer algo
```



Ejercicio 1

Mostrar la calificación de un alumno a partir del puntaje obtenido en el examen, de acuerdo a la siguiente tabla:

Puntaje	Calificación
Entre 90 y 100	5
Entre 80 y 89	4
Entre 70 y 79	3
Entre 60 y 69	2
Menos de 60	1

Ejercicio 1

Mostrar la calificación de un alumno a partir del puntaje obtenido en el examen, de acuerdo a la siguiente tabla:

```
puntaje = int(input("Ingrese el puntaje: "))
if puntaje>=90:
    calificacion=5
elif puntaje>=80:
    calificacion=4
elif puntaje>=70:
    calificacion=3
elif puntaje>=60:
    calificacion=2
else:
    calificacion=1
print("La calificacion es:",calificacion)
```

```
========== RESTART:
Ingrese el puntaje: 78
La calificacion es: 3
>>>
========== RESTART:
Ingrese el puntaje: 51
La calificacion es: 1
>>>
```

Operador ternario

Python permite el uso de operadores ternarios para expresar estructuras condicionales de forma más concisa:

```
resultado_si_verd if condicion else resultado_si_falso
```

Cabe resaltar que el operador ternario entrega un resultado en función a la condición establecida. Por ejemplo, la siguiente expresión:

```
minimo = a if a<b else b
```

Carga en `minimo` el menor valor entre `a` y `b`

Ejemplo 6:

¿Qué imprimirá el siguiente programa?

```
puntuacion=60
resultado = "Aprobado" if puntuacion>=70 else "Reprobado"
print(resultado)
```

Ejercicios propuestos

Ejercicio 1:

Dados dos valores a y b, indicar a través de un mensaje cuál de los 2 es el mayor (Ej: “El mayor es a”). En caso de que sean iguales, indicarlo a través de un mensaje.

Ejercicio 2:

Si los días LUN-DOM se ingresan de forma numérica (del 1 al 7), devolver el nombre del día correspondiente. Si el número es inválido, indicarlo con un mensaje.

Ejercicios propuestos

Ejercicio 3:

Escribir un programa que determine si un alumno tiene o no derecho a examen final en una materia. Un alumno tiene firma si la suma de los puntajes (1P: 24, 2P: 36, TPs/Lab: 10) es mayor o igual a 28. Si tiene derecho entonces imprimir el nombre y su puntaje total. Si no tiene derecho indicar que debe rendir el tercer parcial.

Se leen los puntajes de los dos parciales y el correspondiente a trabajos prácticos y/o laboratorios, y también el nombre del alumno.

Ejemplos:

- Si parc1=20, parc2=30, tps=10, y nombre="Julia"
  Se debe imprimir:
  Julia tiene firma con 60.

- Si parc1=10, parc2=5, tps=5, y nombre="Jose"
  Se debe imprimir:
  Jose no consiguió firma.

Ejercicios propuestos

Ejercicio 4:

Hacer un algoritmo que determine si tres valores ingresados pueden ser lados de un triángulo. Ninguno de sus lados puede ser superior o igual a la suma de los otros dos. Si los valores pueden ser lados de un triángulo, entonces calcular la superficie según la fórmula del semiperímetro.

Ejemplos:
- Si a=10, b=40 y c=100
Se imprime : “No pueden ser lados de un triangulo”.
- Si a=10, b=40 y c=35
Se imprime : “Pueden ser lados de un triangulo”.
Su superficie es:...

Nota: para obtener la raíz cuadrada de un número se utiliza la función `sqrt()` que está en `math`:

```
import math
…
y = math.sqrt(x) #calcula la raíz cuadrada de x y lo asigna a y
```

Ejercicios propuestos

Ejercicio 5:

Determinar si un año (ingresado por teclado) es bisiesto o no, teniendo en cuenta lo siguiente:

- Un año es bisiesto si es múltiplo de 4 pero no de 100, a no ser que lo sea también de 400.

Ejercicio 6 (Desafío):

Diseñar un programa en el que se ingresan tres variables: DIA, MES y ANHO (en forma númerica); y devuelva la fecha del día siguiente (en formato DIA/MES/ANHO). Se deben considerar los años bisiestos, cantidad de días de cada mes, etc. En caso de insertar números reales o fechas inválidas, indicar con un mensaje.

Gracias por la atención