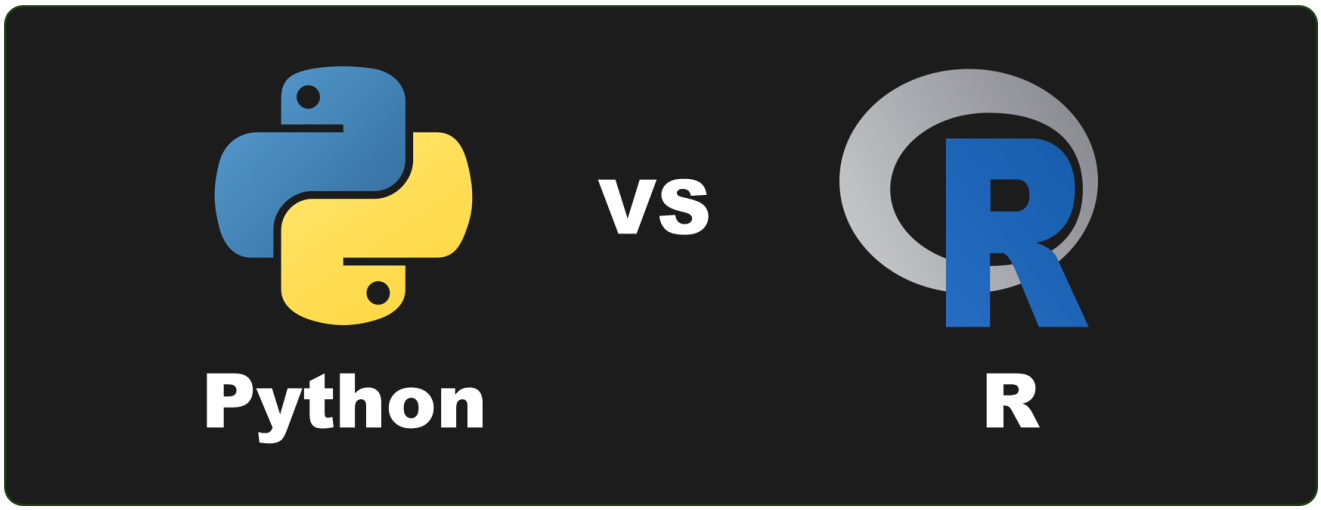
python and R

python and R

python vs R



R and Python are both popular open-source programming languages used for data analysis, statistical computing, and machine learning. However, they have different strengths and weaknesses.

R is commonly used for statistical analysis and data visualization. Python, on the other hand, is a general-purpose programming language that is widely used in data science.

Python and R are both open-source and free to use, which makes them accessible to a wider range of users.

SAS, on the other hand, is proprietary software and requires a license to use, which can be costly for some organizations.

Why do we have many programming languages?###

Technological Evolution: by looking at web development languages like HTML, CSS, and JavaScript and by looking at the evolution of c language showing us new programming languages emerging with the passing of time.

iversity and Specialization: Different programming languages are designed to address specific needs or applications, such as web development, data analysis, or mobile app development. Each language offers unique features and capabilities tailored to specific tasks

Developer Preferences: Developers choose programming languages based on personal tastes and the way they think.

Specific Use Cases: Programming languages are often optimized for specific types of programs. For example, NodeJS is designed for single-threaded web applications, while the R programming language specializes in statistical analysis

Performance and Efficiency:

The choice of language often depends on the trade-offs between convenience, safety, and speed required for a particular job

Legacy Code and Ecosystems:

Developers choose a language based on what they are familiar with.

code of some other languages

Assembly Program

```
.model small          ; tells the assembler to use the small memory model
.stack 100h           ; sets the stack size to 256 bytes (100h in hexadecimal)
.data                 ; declares the start of the data segment

S1 db "Helloworld$"   ; declares a string variable named S1 and initializes it with the value "Helloworld$". The
dollar sign is used to terminate the string.

.code                 ; declares the start of the code segment

main proc             ; declares the start of the main procedure
mov ax,@data          ; moves the address of the data segment to the AX register
mov ds,ax             ; moves the value of AX to the DS register, which sets the data segment register to point to
the data segment
lea dx,s1             ; loads the effective address of the string S1 into the DX register using the LEA (load
effective address) instruction
mov ah,9              ; moves the value 9 into the AH register, which tells the DOS system call to print a string to
the console
INT 21h               ; calls the DOS interrupt 21h, which performs the system call with the specified function
number and arguments
mov ah,4ch            ; moves the value 4Ch into the AH register, which tells the DOS system call to terminate the
program
INT 21h               ; calls the DOS interrupt 21h to terminate the program
main endp             ; declares the end of the main procedure
```

c and c++

```c
#include <stdio.h>
int main() {
printf("Hello, World!\n")
; return 0;
}
```

```
#include <iostream>
int main() {
std::cout << "Hello, World!" << std::endl; return 0;
}
```

Both programs achieve the same output of displaying "Hello, World!" on the screen. The main difference between the two is the use of `printf` in C for output and `std::cout` in C++ for output. Additionally, C++ requires the use of the `iostream` library for input and output operations.

```
// C program to illustrate the arithmatic operators

#include <stdio.h>
int main()
{

    int a = 25, b = 5;

    // using operators and printing results
    printf("a + b = %d\n", a + b);
    printf("a - b = %d\n", a - b);
    printf("a * b = %d\n", a * b);
    printf("a / b = %d\n", a / b);
    printf("a % b = %d\n", a % b);
    printf("+a = %d\n", +a);
    printf("-a = %d\n", -a);
    printf("a++ = %d\n", a++);
    printf("a-- = %d\n", a--);

    return 0;
}
//a < b  : 0
```

```
//a > b  : 1
//a <= b: 0
//a >= b: 1
//a == b: 0
//a != b : 1
```

pyhton and ruby

```
puts "Hello World"
```

```
print("hello world")
```

```
# Ruby program to show Operators Precedence
        a = 20;
        b = 10;
        c = 15;
        d = 5;
        e = 0

        # operators with the highest precedence
        # will operate first
        e = a + b * c / d;

                # step 1: 20 + (10 * 15) /5
                # step 2: 20 + (150 /5)
                # step 3:(20 + 30)

        puts"Value of a + b * c / d is : #{e}"
#Value of a + b * c / d is : 50
```

install python and vs code

python

vs code

[other website](install python and R)

start coding with python

#lets_strat

basic

- Syntax
- Comments
- Variables
- Data Types
  - Numbers
  - Casting
  - Strings
  - Booleans
- Operators
  - conditional
    - If...Else
    - While Loops
    - For Loops
- data structures
  - Lists
  - Tuples
  - Sets
  - Dictionaries

  advanced
- Object-oriented programming
- Functions

    - Lambda
- Classes/Objects
  - Inheritance
  - Iterators
  - Modules
  other
- JSON
- PIP
- Try...Except
- User Input
  libraries
  - NumPy
  - Pandas
  - Matplotlib
  - TensorFlow
  - PyTorch
  data science
  - network

web site for learn python

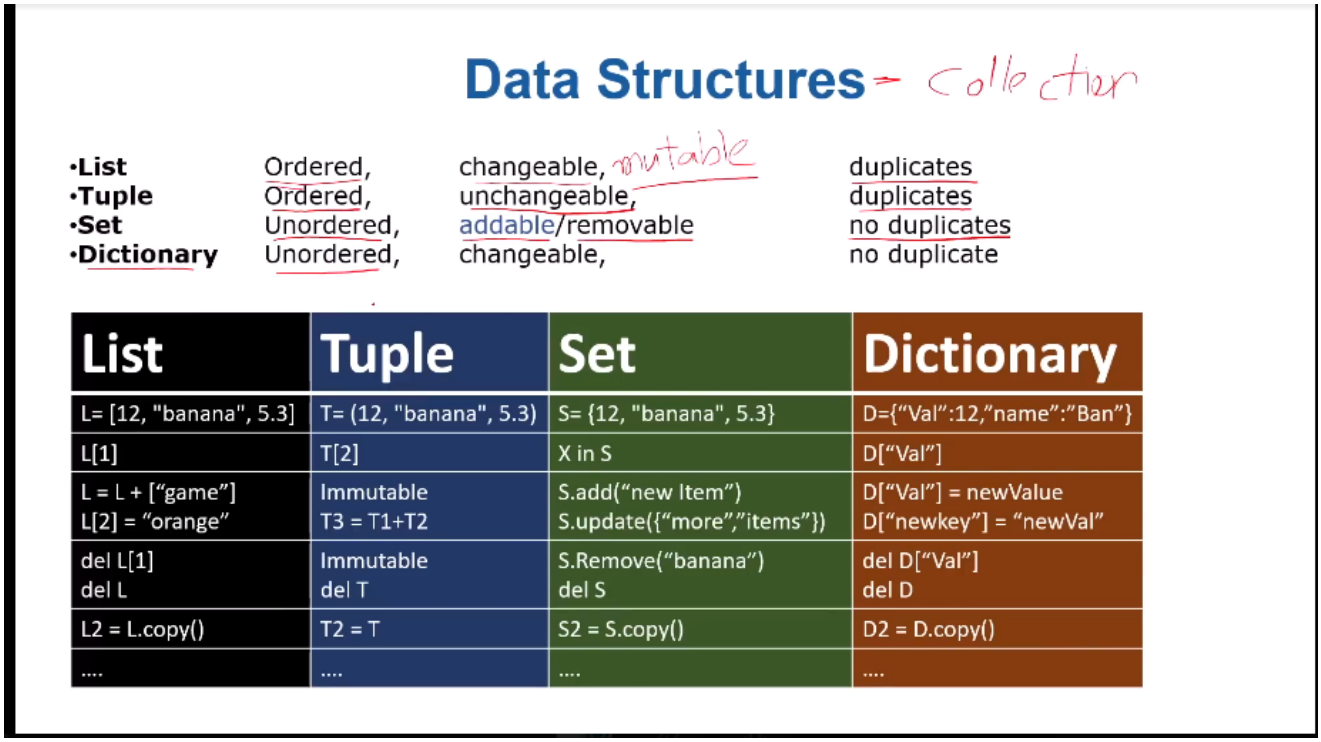
w3schools
programiz.com
geeksforgeeks
docs.python.org

other

Logical Operators

X	Y	X and Y	X or Y	not(X)	not(Y)
T	T	T	T	F	F
T	F	F	T	F	T
F	T	F	T	T	F
F	F	F	F	T	T

data-structures



Data Structures = Collection

•**List**	Ordered,	changeable, mutable	duplicates
•**Tuple**	Ordered,	unchangeable,	duplicates
•**Set**	Unordered,	addable/removable	no duplicates
•**Dictionary**	Unordered,	changeable,	no duplicate

List	Tuple	Set	Dictionary
L= [12, "banana", 5.3]	T= (12, "banana", 5.3)	S= {12, "banana", 5.3}	D={"Val":12,"name":"Ban"}
L[1]	T[2]	X in S	D["Val"]
L = L + ["game"] L[2] = "orange"	Immutable T3 = T1+T2	S.add("new Item") S.update({"more","items"})	D["Val"] = newValue D["newkey"] = "newVal"
del L[1] del L	Immutable del T	S.Remove("banana") del S	del D["Val"] del D
L2 = L.copy()	T2 = T	S2 = S.copy()	D2 = D.copy()
....	....	....	....