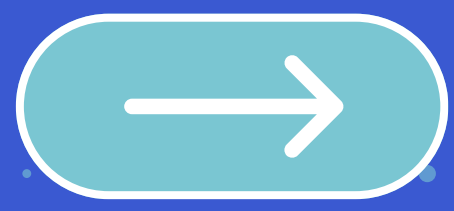
The Ultimate Python Interview Prep Guide (100 Q&A)

by Amr AbdElkarem

PROGRAMMINGVALLEY.COM



What’s the difference between Shallow Copy and Deep Copy in Python?

Shallow Copy

- Creates a new object
- Copies only references, not nested objects
- Changes in copy affect the original

```
import copy
original = [[1, 2], [3, 4]]
shallow = copy.copy(original)
shallow[0][0] = 9  # also changes
original
```

Deep Copy

- Creates a completely independent object
- Recursively copies nested items
- Changes in copy don’t affect original

```
import copy
original = [[1, 2], [3, 4]]
deep = copy.deepcopy(original)
deep[0][0] = 9  # original stays the
same
```

How is Multithreading achieved in Python?

- Use the threading module
- Threads run concurrently but not true parallel (GIL)
- Best for I/O tasks (files, network, DB)
- Synchronize with Lock to avoid race conditions

```python
import threading

def task():
    print("Running")

t = threading.Thread(target=task)
t.start()
t.join()
```

Discuss Django architecture

- Model: handles database and data logic
- View: connects Model and Template, maps to URL
- Template: front-end presentation
- Django: delivers the final page to the user

What advantage does NumPy array have over a Nested list?

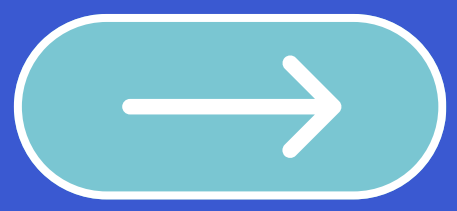
- Faster performance
- More memory efficient
- Supports vectorized operations
- Rich set of math functions
- Easy handling of multidimensional data

What are Pickling and Unpickling?

- Pickling: convert Python object → byte stream (serialization)
- Unpickling: convert byte stream → Python object (deserialization)
- Example use: save a trained ML model with pickle, then reload it later

How is Memory managed in Python?

- Python uses a private heap space for all objects
- Memory manager handles allocation, sharing, caching, segmentation
- Heap access is controlled by the Python interpreter, not the user



Are arguments in Python passed by value or by reference?

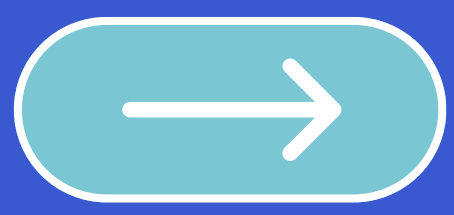
- Python uses pass-by-reference
- Changes inside a function affect the original object

How do you generate Random numbers in Python?

- Use random module:
    - randint(1,10) → random integer
    - uniform(0,1) → random float
- randrange(1,10,2) → step-based random number
- Use NumPy:
    - np.random.rand(3) → array of random floats

What does the // operator do in Python?

- / → normal division, returns float (e.g., 5 / 2 = 2.5)
- // → floor division, returns integer quotient (e.g., 5 // 2 = 2)



What does the is operator do in Python?

- Checks if two variables refer to the same object in memory
- Compares identity, not value

```python
a = [1,2,3]
b = a
c = [1,2,3]

print(a is b)  # True
print(a is c)  # False
```

What is the purpose of the pass statement in Python?

- Used as a placeholder where code is syntactically required but no action is needed
- Common in empty loops, functions, or classes

```python
for ch in "Si mplilea rn":
    if ch == " ":
        pass
    else:
        print(ch, end="")
```

How to check if all characters in a string are alphanumeric in Python?

- Use .isalnum() method
- Returns True if all characters are letters or numbers

```python
s = "Hello123"
print(s.isalnum())  # True
```

How do you merge elements in a sequence in Python?

- Use the .join() method with a separator
- Works with lists, tuples, or any string sequence

```python
words = ['Hello', 'World', 'Python']
merged = ' '.join(words)
print(merged)  # Hello World Python
```

How do you remove all leading whitespace in a string in Python?

- Use .lstrip()
- Removes spaces (or specified chars) only from the start of the string

```
text = "   Hello, World!"
print(text.lstrip())  # Hello, World!
```

How do you replace all occurrences of a substring in Python?

- Use .replace(old, new)
- Returns a new string with all matches replaced

```
text = "Hello World, World is great!"
print(text.replace("World", "Python"))
# Hello Python, Python is great!
```

What is the difference between del and remove() on lists?

- del list[start:end] → deletes elements by index/range
- list.remove(x) → deletes first occurrence of value

```
lis = ['a','b','c','d']
del lis[1:3]       # ['a','d']

lis = ['a','b','b','d']
lis.remove('b')    # ['a','b','d']
```

How do you display the contents of a text file in reverse order in Python?

- Reverse lines:

```
with open('file.txt') as f:
    for line in reversed(f.readlines()):
        print(line.strip())
```

- Reverse characters:

```
with open('file.txt') as f:
    print(f.read()[::-1])
```

What's the difference between append() and extend() in Python?

- append(x) → adds a single element at the end

```python
lst = [1,2,3]
lst.append(4)      # [1,2,3,4]
```

- extend(iterable) → adds multiple elements from another iterable

```python
lst = [1,2,3]
lst.extend([4,5,6])  # [1,2,3,4,5,6]
```

What is the output of this code and why?

```python
def addToList(val, list=[]):
    list.append(val)
    return list

list1 = addToList(1)
list2 = addToList(123, [])
list3 = addToList('a')
```

Output:

- list1 = [1, 'a']
- list2 = [123]
- list3 = [1, 'a']

Reason: Default list [] is created once, so calls without a new list share the same object. Passing [] explicitly creates a fresh list.

What is the difference between a list and a tuple in Python?

- List
  - Mutable → can change after creation
  - Defined with [ ]
  - Slower, more methods (append, remove)
  - Best for dynamic data
- Tuple
  - Immutable → cannot change after creation
  - Defined with ( )
  - Faster, fewer methods
  - Best for fixed data

```
lst = [1,2,3]
tpl = (1,2,3)
```

What is a docstring in Python?

- A string used for documentation of modules, classes, functions, and methods
- Accessed with .__doc__ or help()

```python
def add(a, b):
    """This function adds two
numbers."""
    return a + b

print(add.__doc__)    # This function
adds two numbers.
```

How do you use print() without a newline in Python?

- Use the end parameter to control line ending
- Default: end="\n" → newline
- Change to end="" or custom string

```
print("Hello", end="")
print("World")    # HelloWorld
```

How do you use the split() function in Python?

- Splits a string into a list of substrings
- Default separator → whitespace
- Can specify custom separator and max splits

```
text = "Hello World Python"
print(text.split())       # ['Hello',
'World', 'Python']
print("a,b,c".split(",")) # ['a', 'b',
'c']
```

Is Python object-oriented or functional?

- Both → Python is a multi-paradigm language
- OOP features: classes, objects, inheritance, polymorphism
- Functional features: first-class functions, lambdas, higher-order functions, list comprehensions
- You can mix both approaches as needed

How do you write a function that takes a variable number of arguments in Python?

- Use *args → variable positional arguments
- Use **kwargs → variable keyword arguments

```python
def my_function(*args, **kwargs):
    pass
```

```python
my_function(1, 2, name="Alice", age=25)
```

What are *args and **kwargs in Python?

- *args → collects extra positional arguments into a tuple

```python
def f(*args):
    print(args)

f(1,2,3)  # (1, 2, 3)
```

- **kwargs → collects extra keyword arguments into a dict

```python
def f(**kwargs):
    print(kwargs)

f(a=1,b=2)  # {'a': 1, 'b': 2}
```

“In Python, functions are first-class objects.” What does this mean?

- Functions are treated like any other object
- You can:
  - Assign them to variables
  - Pass them as arguments
  - Return them from other functions
  - Store them in data structures

```python
def greet(): print("Hi")
f = greet
f()  # Hi
```

What is a NumPy array?

- A fast, memory-efficient, grid-like data structure
- Handles large multi-dimensional arrays and matrices
- Stores elements of the same data type
- Supports vectorized operations for numerical computing

```python
import numpy as np
arr = np.array([1, 2, 3, 4])
```

What is the difference between Matrices and Arrays in Python?

- Matrix
  - 2D data structure (from linear algebra)
  - Supports matrix-specific math operations
- Array
  - Sequence of elements of the same type
  - Can be 1D, 2D, or multi-dimensional
  - A matrix is essentially a 2D array

How do you get indices of n maximum values in a NumPy array?

- Use np.argsort() and slice the last n indices

```python
import numpy as np
arr = np.array([1,3,2,7,5])
n = 2
indices = np.argsort(arr)[-n:]
print(indices)  # [4, 3]
```

How do you split a dataset into train_set and test_set in Python?

- Use train_test_split() from scikit-learn
- Control split with test_size

```
from sklearn.model_selection import
train_test_split

res_set = [1,2,3,4,5,6,7,8,9,10]
train_set, test_set =
train_test_split(res_set, test_size=0.2,
random_state=42)

print(train_set)  # 80%
print(test_set)   # 20%
```

How do you import a Decision Tree Classifier in scikit-learn?

1. from sklearn.decision_tree import DecisionTreeClassifier
2. from sklearn.ensemble import DecisionTreeClassifier
3. from sklearn.tree import DecisionTreeClassifier
4. None of these

Answer: 3. from sklearn.tree import DecisionTreeClassifier

How can you access a public Google Spreadsheet CSV in Python?

- Share sheet → “Anyone with link can view”
- Replace /edit with /export?format=csv in URL
- Use pandas to load

```
import pandas as pd

url =
"https://docs.google.com/spreadsheets/d/·
format=csv"
df = pd.read_csv(url)
print(df)
```

Difference between df['Name'] and df.loc[:, 'Name']?

1. df['Name'] = view, df.loc[:, 'Name'] = copy
2. df['Name'] = copy, df.loc[:, 'Name'] = view
3. Both are copies
4. Both are views

Answer: 3 → Both are copies of the original DataFrame

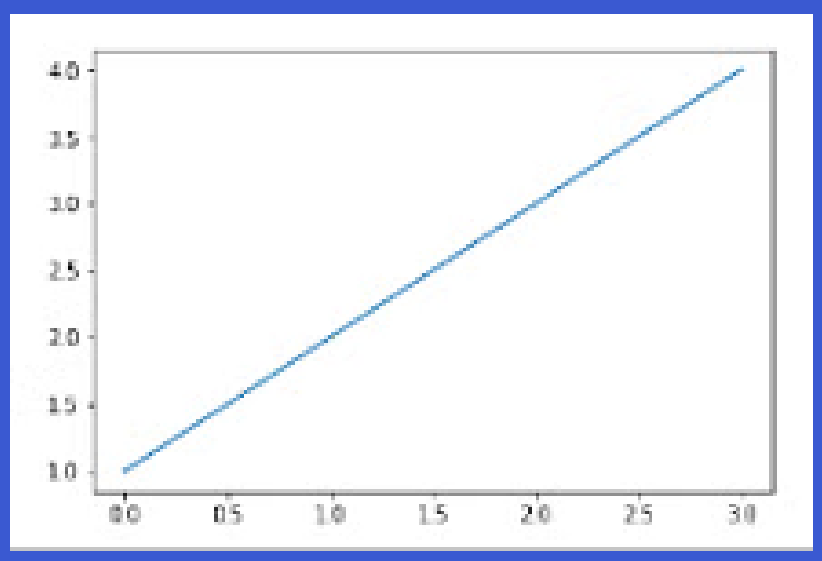
You get a UnicodeEncodeError reading “temp.csv” with pandas. Which fixes it?

1. pd.read_csv("temp.csv", compression='gzip')
2. pd.read_csv("temp.csv", dialect='str')
3. pd.read_csv("temp.csv", encoding='utf-8')
4. None of these

Answer: 3 → Use encoding='utf-8'

How do you set a line width in the plot given below?

```
import matplotlib.pyplot as plt
plt.plot([1,2,3,4])
plt.show()
```



1. In line two, write plt.plot([1,2,3,4], width=3)
2. In line two, write plt.plot([1,2,3,4], line_width=3
3. In line two, write plt.plot([1,2,3,4], lw=3)
4. None of these

Answer: 3. In line two, write plt.plot([1,2,3,4], lw=3).

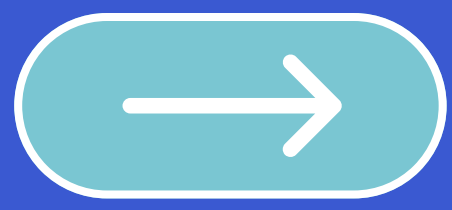
How do you reset the index of a DataFrame to a given list?

- df.reset_index(new_index,)
- df.reindex(new_index,)
- df.reindex_like(new_index,)
- None of these

Answer: 3 → df.reindex_like(new_index,)

What is the difference between range() and xrange() in Python?

- Python 2
  - range() → returns a full list (high memory use)
  - xrange() → returns an iterator (lazy, memory-efficient)
- Python 3
  - xrange() removed
  - range() now behaves like old xrange()



How can you check if a pandas DataFrame is empty?

- Use the .empty attribute
- Returns True if no rows/columns, else False

```python
import pandas as pd
df = pd.DataFrame()
print(df.empty)  # True
```

How do you sort a NumPy array by the (N-1)th (last) column?

- Use slicing with argsort()

```python
import numpy as np

arr = np.array([[1,5,3],
                [4,2,9],
                [7,8,6]])

sorted_arr = arr[arr[:, -1].argsort()]
print(sorted_arr)
```

How do you create a Pandas Series from a list, NumPy array, and dictionary?

- Use the .empty attribute
- Returns True if no rows/columns, else False

```python
import pandas as pd
import numpy as np

# From list
pd.Series([1,2,3,4])

# From NumPy array
pd.Series(np.array([10,20,30,40]))

# From dictionary
pd.Series({'a':1, 'b':2, 'c':3})
```

Each creates a 1D labeled array.

How do you get items not common to both Series A and B in pandas?

- Using symmetric_difference:

```
import pandas as pd
A = pd.Series([1,2,3,4,5])
B = pd.Series([4,5,6,7,8])

res =
pd.Series(list(set(A).symmetric_differen
print(res)  # [1,2,3,6,7,8]
```

- Using ~isin() + append():

```
# Items not in both A and B
result =
A[~A.isin(B)].append(B[~B.isin(A)])
print(result)
```

How do you keep only the top 2 most frequent values in a Series and replace others with 'Other'?

```python
import pandas as pd, numpy as np
ser =
pd.Series(np.random.randint(1,5,12))

top2 = ser.value_counts().index[:2]
ser[~ser.isin(top2)] = 'Other'
print(ser)
```

Result: Only top 2 frequent values remain, rest → "Other"

How do you find positions of numbers that are multiples of 3 in a Series?

```python
import pandas as pd, numpy as np
ser =
pd.Series(np.random.randint(1,10,7))
print(ser)

positions = np.argwhere(ser % 3 == 0)
print(positions)
```

Output: Indices where values are divisible by 3.

How do you reverse the rows of a DataFrame in pandas?

```python
import pandas as pd, numpy as np
df =
pd.DataFrame(np.arange(25).reshape(5,-1))

print(df.iloc[::-1, :])
```

Explanation: iloc[::-1, :] reverses row order.

If you split your data into train/test sets, can you still overfit?

Yes.

- Happens if you repeatedly tune/retrain based on test set results
- Test set should be used only once for final evaluation
- Use a validation set or cross-validation to tune models safely

Which Python library is built on top of matplotlib and pandas for easier plotting?

Answer: Seaborn

- High-level data visualization library
- Simplifies creation of statistical and informative graphs

```
import seaborn as sns
sns.histplot([1,2,2,3,3,3,4])
```

What are the essential features of Python?

- Scripting language → no compilation needed before execution
- Dynamically typed → no need to declare variable types
- Supports object-oriented programming → classes, inheritance, composition

What type of language is Python?

- General-purpose programming language
- Also used for scripting

Why is Python an interpreted language?

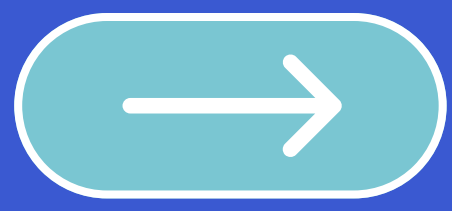
- Code is executed line by line at runtime
- Not compiled into machine code beforehand

What is PEP 8?

- Python Enhancement Proposal 8
- Official style guide for writing clean, readable Python code

What is a Python namespace?

- Container mapping names → objects
- Types:
  - Local → inside functions
  - Global → top-level of script/module
  - Built-in → standard functions/exceptions



What are decorators in Python?

- Functions that modify other functions' behavior
- Change functionality without altering the function's code

How do you use decorators in Python?

```python
def decorator(func):
    def wrapper():
        print("Before")
        func()
        print("After")
    return wrapper

@decorator
def greet():
    print("Hello")

greet()
```

What is the difference between .py and .pyc files in Python?

- .py → source code written by the developer
- .pyc → compiled bytecode, auto-generated when a module is imported
- Purpose: .pyc speeds up execution by skipping recompilation

What is slicing in Python?

- Technique to extract part of a sequence (list, tuple, string)
- Syntax: sequence[start:stop:step]
  - start → inclusive index
  - stop → exclusive index
  - step → optional, skip size

```
my_list = [1,2,3,4,5]
print(my_list[1:4])  # [2,3,4]
```

How to use the slicing operator in Python?

- Syntax: [start:end:step]
  - start → inclusive index
  - end → exclusive index
  - step → skip size (default = 1)
- Negative index → counts from end

```
s = "Python"
print(s[1:5])     # ytho
print(s[::-1])    # nohtyP
```

What are keywords in Python?

- Reserved words with predefined meaning
- Cannot be used as variable or function names
- Python has 33 keywords (examples):

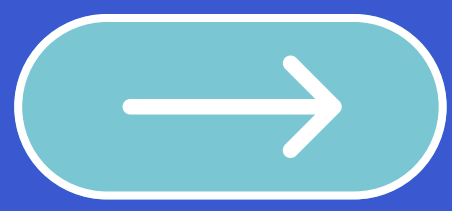
False, True, None, and, or, not, if, elif, else, while, for, break, continue, return, def, class, try, except, finally, with, as, import, from, global, nonlocal, assert, lambda, yield, raise, del, pass, in, is

How do we combine DataFrames in Pandas?

- Concatenate vertically (stack rows)
- Concatenate horizontally (stack columns)
- Merge into a single column

Key features of Python 3.9.0

- New modules: zoneinfo, graphlib
- Improved: asyncio, ast
- PEG parser replaces LL1
- New string methods: remove prefixes/suffixes
- Better assignment idioms, signal handling, type hinting



How is memory managed in Python?

- Objects/data in private heap space (not user accessible)
- Garbage collector reclaims unused memory
- Core API exposes tools for memory management

What is PYTHONPATH?

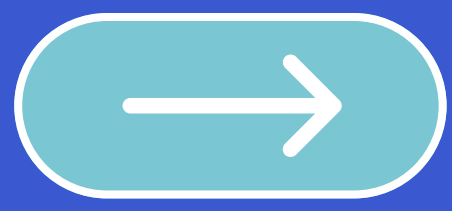
- Environment variable checked during import
- Tells interpreter where to look for modules

Global vs Local variables

- Local → declared inside function, exists only there
- Global → declared outside function, accessible by all functions

Is Python case-sensitive?

- Yes
- Variable, variable, VARIABLE → treated as different identifiers



How to install Python on Windows and set path

- Download from python.org
- Install and check with python in CMD
- Add new env variable PYTHON_HOME with install path
- Edit Path → add %PYTHON_HOME%

On Unix, how do you make a Python script executable?

The script file should start with #!/usr/bin/env python.

On Unix, how do you make a Python script executable?

- Represents the instance of a class
- Used to access attributes and methods inside the class
- Binds variables to the object
- Not a keyword (can be renamed, but conventionally self)

```
class Person:
    def __init__(self, name):
        self.name = name
```

What are literals in Python?

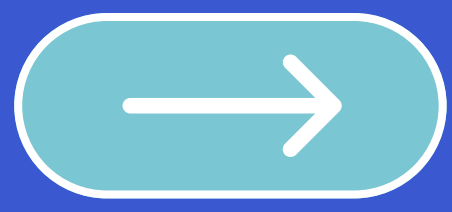
- Fixed values directly assigned to variables
- Represent constants in source code

Types of literals in Python

- String → "Hello" or '"Multiline"'
- Numeric → 10, 3.14, 1+2j
- Character → 'A' (actually a string of length 1)
- Boolean → True, False
- Collections → [ ], ( ), { }, {"a":1}

What are Python modules?

- A .py file containing code (functions, classes, variables)
- Common built-ins: json, datetime, random, math, sys, os



What is __init__?

- Constructor method in classes
- Called when a new object is created

```
class A:
    def __init__(self):
        print("Init called")
```

What is a Lambda function?

- Anonymous, single-expression function
- Syntax: lambda args: expr

```
add = lambda x,y: x+y
print(add(2,3))  # 5
```

Why use Lambda in Python?

- For short, throwaway functions
- Can be:
  - Assigned to variables
  - Used inside other functions (e.g., map, filter)

How do continue, break, and pass work in Python?

- continue → skips to next loop iteration

```python
for i in range(5):
    if i == 2: continue
    print(i)   # 0,1,3,4
```

- break → exits the loop

```python
for i in range(5):
    if i == 2: break
    print(i)   # 0,1
```

- pass → placeholder, does nothing

```python
for i in range(3):
    if i == 1: pass
    print(i)   # 0,1,2
```

What are Python iterators?

- Objects that let you traverse elements one by one
- Must implement:
  - __iter__() → returns iterator
  - __next__() → returns next item, raises StopIteration when done

```
it = iter([1,2,3])
print(next(it))  # 1
```

Difference between range and xrange

- Python 2
  - range() → returns a full list (high memory use)
  - xrange() → returns an iterator (lazy, memory-efficient)
- Python 3
  - xrange() removed
  - range() behaves like old xrange()

Built-in data types in Python

- Numeric → int, float
- Sequence → str, list, tuple
- Mapping → dict
- Set types → set
- Boolean → bool

What are generators in Python?

- Special functions that return an iterator
- Use yield instead of return
- Generate values lazily (one at a time, on demand)
- Memory-efficient for large datasets

```
def my_gen():
    yield 1; yield 2; yield 3

for v in my_gen():
    print(v)   # 1,2,3
```

How do you copy an object in Python?

- Shallow copy → copies object but not nested objects (references shared)

```python
import copy

original = [[1,2],[3,4]]
shallow = copy.copy(original)

shallow[0][0] = 9
print(original)  # [[9,2],[3,4]]
```

- Deep copy → creates a fully independent copy (nested objects included)

```python
import copy

original = [[1,2],[3,4]]
deep_copied = copy.deepcopy(original)

deep_copied[0][0] = 9
print(original)  # [[1,2],[3,4]]
```

In Python, are arguments passed by value or reference?

- Python uses pass-by-object-reference (a mix of both concepts)
- Mutable objects → changes inside function affect original
- Immutable objects → changes inside function don't affect original

```python
def appendNumber(arr):
    arr.append(4)

arr = [1,2,3]
appendNumber(arr)
print(arr)  # [1,2,3,4]
```

How do you delete a file in Python?

- Use os.remove(filename)

```python
import os
os.remove("file.txt")
```

Deletes the file from the system.

Explain join() and split() in Python

- split(delimiter) → splits string into list

```
s = "This is a string."
print(s.split(" "))  #
['This','is','a','string.']
```

- join(iterable) → joins list into string with delimiter

```
lst = ['This','is','a','string.']
print(" ".join(lst))  # This is a
string.
```

What are negative indexes in Python and why are they used?

- The indexes from the end of the list, tuple, or string are called negative indexes.
- Arr[-1] denotes the array's last element. Arr[]

How to capitalize the first letter of a string?

- Use .capitalize()

```python
print("python".capitalize())  # Python
```

How to convert a string to lowercase?

- Use .lower()

```python
print("HELLO".lower())  # hello
```

How to comment multiple lines in Python?

- Prefix each line with #
- Shortcut in editors: select lines + add #
- No official multiline comment syntax (triple quotes used as docstrings, not true comments)

Is indentation required in Python?

- Yes, indentation defines code blocks
- Replaces { } used in other languages

Purpose of not, is, in operators

- not → boolean negation
- is → identity check (same object)
- in → membership test

What are help() and dir() used for?

- dir() → lists attributes/symbols of an object/module
- help() → shows docstring and detailed help

Why isn’t all memory deallocated when Python exits?

- Circular references may remain
- Objects held in global namespaces not freed
- Some memory reserved by C libraries is not released

What is a dictionary in Python?

- Built-in mapping type → key-value pairs
- Keys are unique, values accessed via keys

```
d = {"a":1, "b":2}
```

How to use ternary operators in Python?

```python
x = 5
result = "Even" if x % 2 == 0 else "Odd"
```

Explain split(), sub(), subn() in re module

- split() → split string by regex pattern
- sub() → replace matches with new string
- subn() → same as sub but also returns count

What are negative indexes and why use them?

- Index from end of sequence (-1 last, -2 second last)
- Useful for reverse access or slicing

```python
s = "Python"
print(s[-1])  # n
```

What are Python packages?(), subn() in re module

- Collection of modules in a directory with __init__.py
- Organizes and reuses code

```
from mypackage import module1
```

Built-in types of Python

- Numeric: int, float, complex
- Sequence: str, list, tuple
- Set: set
- Mapping: dict
- Boolean: bool
- Built-in functions also available

Benefits of NumPy arrays over Python lists

- Faster, more memory efficient
- Support vectorized operations (elementwise math)
- Consistent data type (no per-element type checks)
- Rich library support: linear algebra, stats, FFT, search

Find this useful? like and share this post with your friends.

by Amr AbdElkarem
PROGRAMMINGVALLEY.COM

Save