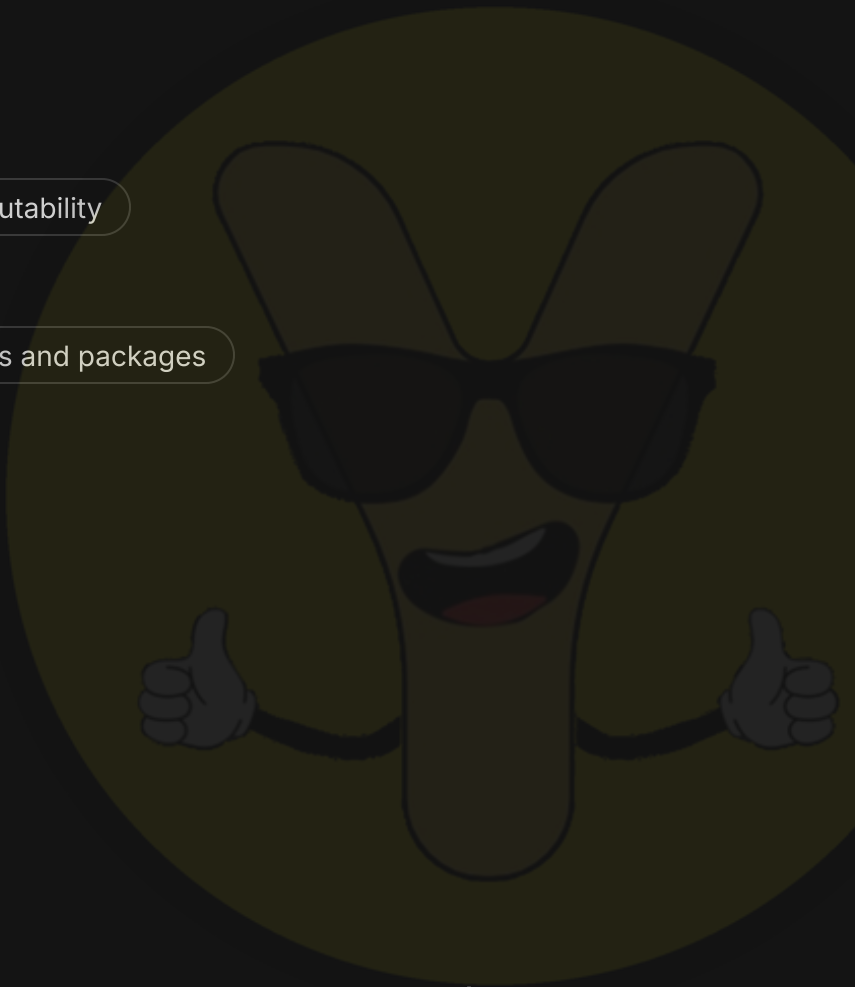
AKTU · 2ND YEAR · BCC302 / BCC402

Python Programming

Unit 2 — Data Structures and Functions

- Strings: operations, slicing, methods
- Lists: operations, slicing, methods
- Tuples: creation, operations, immutability
- Dictionaries: creation, operations, methods
- Sets: creation, operations, methods
- Functions: definition, calling, arguments, return values
- Scope of variables
- Lambda functions
- Modules and packages



DECODED BY

Chirag Jain

Founder, Ycotes · Gold Medalist in Artificial Intelligence, IIT Madras · taught 10,000+ students

theycotes.com

Introduction to Data Structures

DATA STRUCTURE

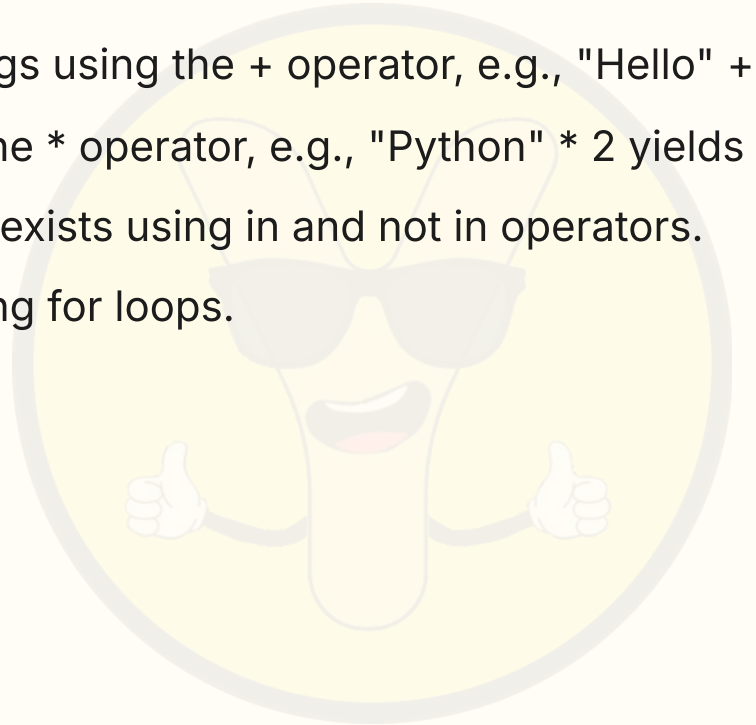
A data structure is a specialized format for organizing, processing, retrieving, and storing data efficiently in computer memory.

- Python provides built-in data structures: Strings, Lists, Tuples, Dictionaries, and Sets.
- They allow grouping related values and performing operations through built-in methods.
- Choosing the right structure optimizes memory and execution time for AKTU exam implementations.

Strings: Operations

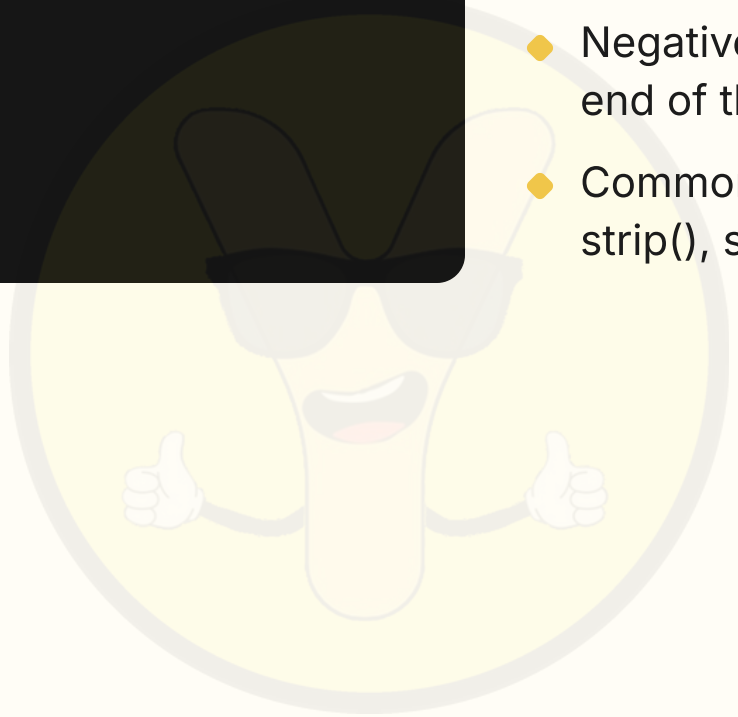
Strings are immutable sequences of Unicode characters enclosed in single, double, or triple quotes.

- Concatenation: Combines two strings using the + operator, e.g., "Hello" + "World" yields "HelloWorld".
- Repetition: Repeats a string using the * operator, e.g., "Python" * 2 yields "PythonPython".
- Membership: Checks if a substring exists using in and not in operators.
- Iteration: Traversing characters using for loops.



Strings: Slicing and Methods

```
s = "AKTU Lucknow"
print(s[0:4])      # AKTU
print(s[5:])       # Lucknow
print(s[::-1])     # Reverse string
print(s.lower())   # aktu lucknow
print(s.replace("Lucknow", "Delhi"))
```

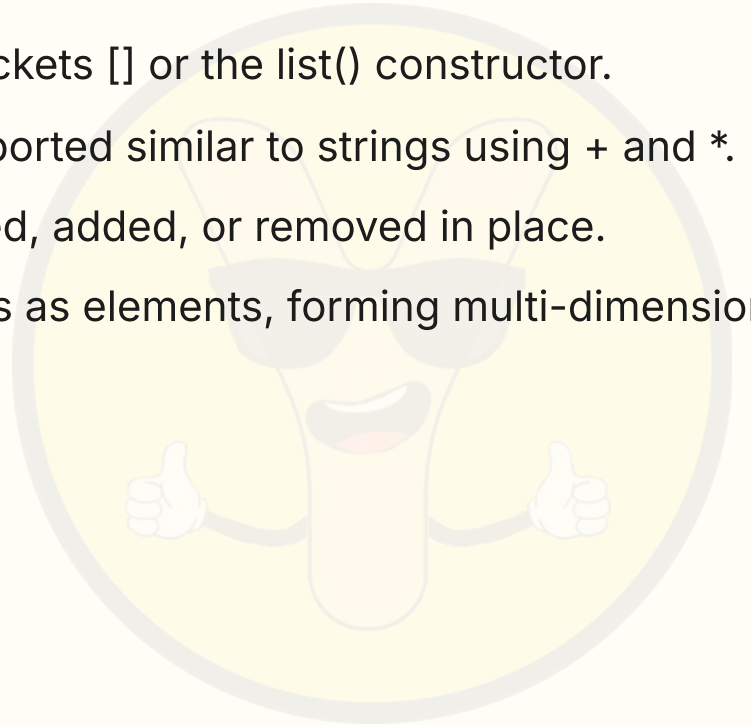


- Slicing syntax: string[start:end:step].
- Negative indices count backwards from the end of the string.
- Common methods include lower(), upper(), strip(), split(), find(), and replace().

Lists: Operations

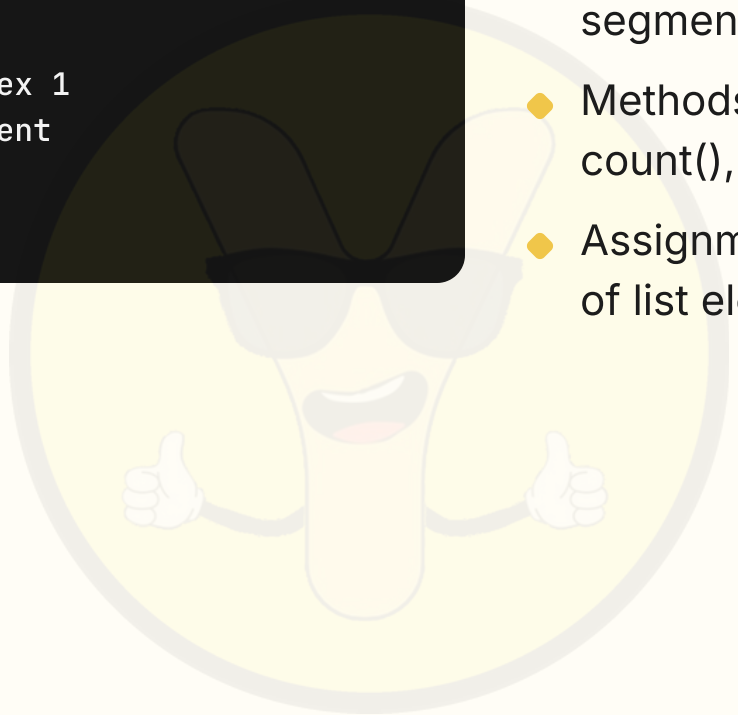
Lists are ordered, mutable collections that can hold heterogeneous data types.

- Creation: Defined using square brackets [] or the list() constructor.
- Concatenation and Repetition: Supported similar to strings using + and *.
- Mutability: Elements can be modified, added, or removed in place.
- Nesting: Lists can contain other lists as elements, forming multi-dimensional structures.



Lists: Slicing and Methods

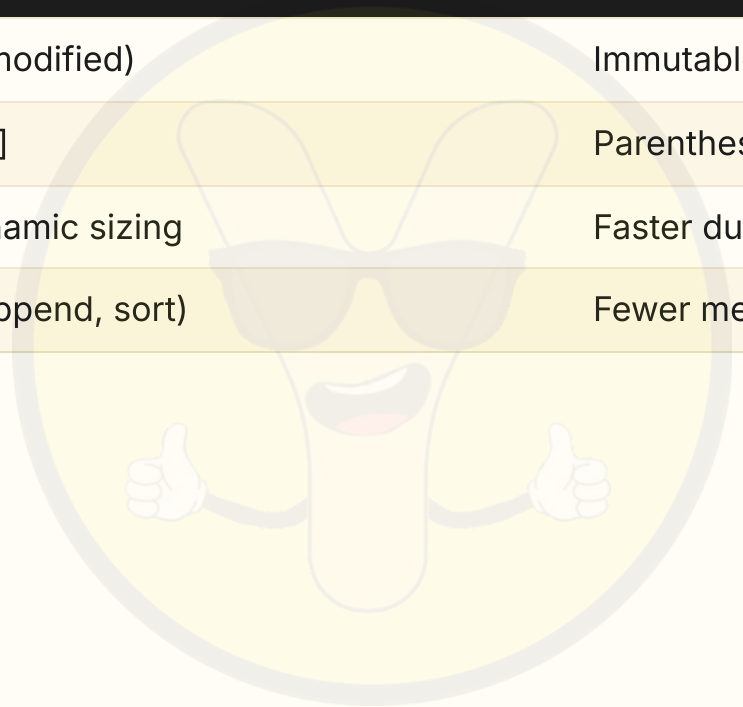
```
lst = [10, 20, 30, 40, 50]
lst.append(60)      # Adds 60 at end
lst.insert(1, 15)   # Inserts 15 at index 1
lst.pop()           # Removes last element
lst.sort()          # Sorts in-place
print(lst[1:4])     # Slicing sublist
```



- List slicing allows extracting or replacing sub-segments efficiently.
- Methods like extend(), remove(), index(), count(), and reverse() are heavily used.
- Assignment via slice allows bulk modification of list elements.

Lists vs Tuples

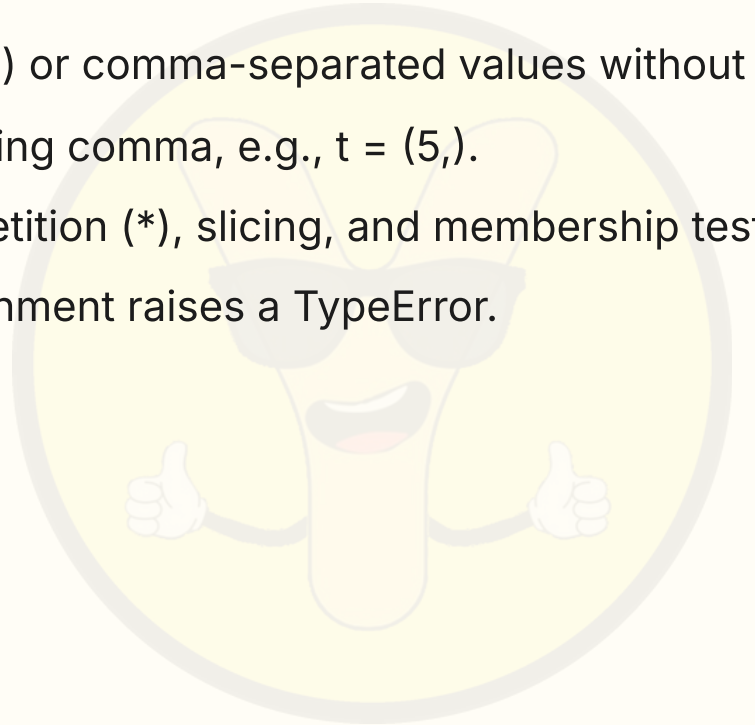
Feature	Lists	Tuples
Mutability	Mutable (can be modified)	Immutable (cannot be changed)
Syntax	Square brackets []	Parentheses ()
Performance	Slower due to dynamic sizing	Faster due to fixed size
Methods	Many methods (append, sort)	Fewer methods (count, index)



Tuples: Creation and Operations

Tuples are immutable ordered sequences, ideal for protecting read-only data.

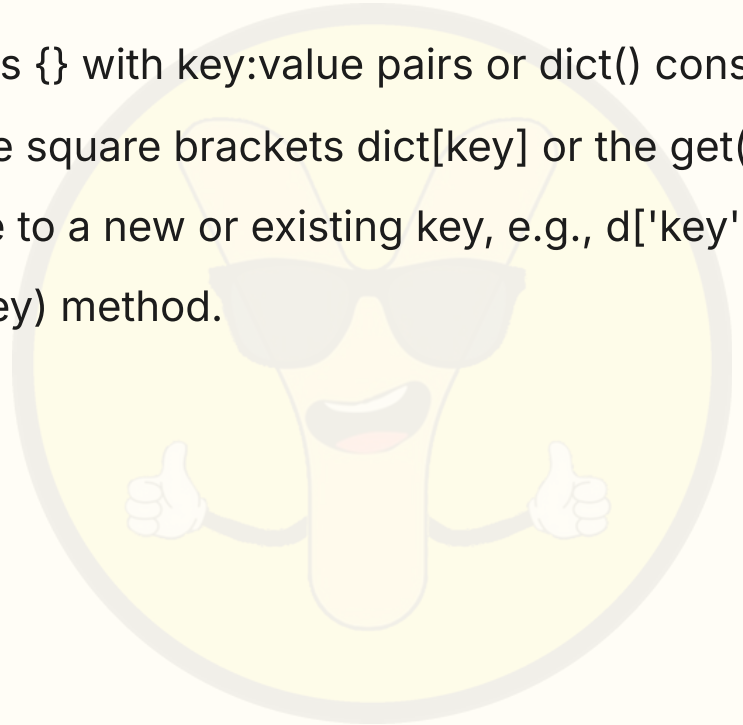
- Creation: Enclosed in parentheses () or comma-separated values without brackets.
- Single element tuple requires a trailing comma, e.g., t = (5,).
- Operations: Concatenation (+), repetition (*), slicing, and membership testing (in).
- Immutability: Attempting item assignment raises a TypeError.



Dictionaries: Creation and Operations

Dictionaries store key-value pairs where keys must be unique and immutable.

- Creation: Defined using curly braces {} with key:value pairs or dict() constructor.
- Accessing values: Using keys inside square brackets dict[key] or the get() method.
- Adding/Updating: Assigning a value to a new or existing key, e.g., d['key'] = value.
- Deletion: Using del d[key] or pop(key) method.



Dictionaries: Methods

```
student = {"name": "Amit", "roll": 101, "marks": 85}
print(student.keys())    # dict_keys(['name', 'roll',
'marks'])
print(student.values())  # dict_values(['Amit', 101, 85])
print(student.items())   # dict_items([...])
student.update({"marks": 90})
print(student.pop("roll"))
```

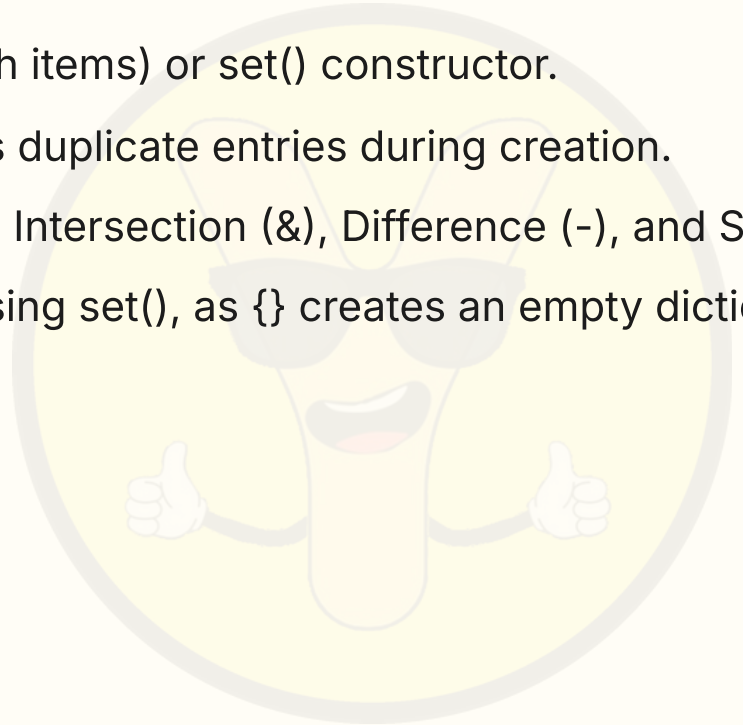


- keys(), values(), and items() return view objects reflecting dictionary changes.
- update() merges another dictionary or iterable of key-value pairs.
- setdefault() inserts key with a default value if the key is not already present.

Sets: Creation and Operations

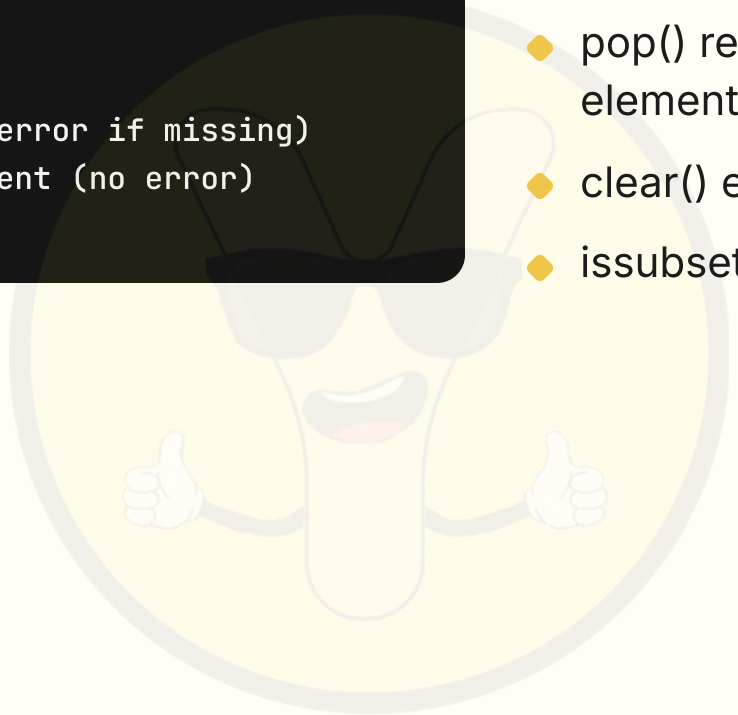
Sets are unordered collections of unique, immutable elements.

- Creation: Using curly braces {} (with items) or set() constructor.
- Uniqueness: Automatically removes duplicate entries during creation.
- Mathematical Operations: Union (|), Intersection (&), Difference (-), and Symmetric Difference (^).
- Note: Empty set must be created using set(), as {} creates an empty dictionary.



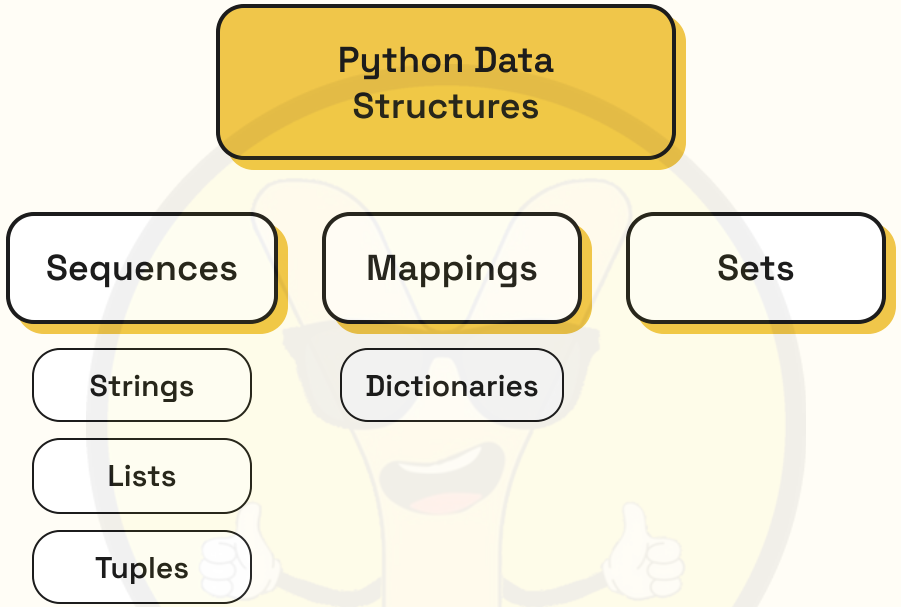
Sets: Methods

```
A = {1, 2, 3}
B = {3, 4, 5}
A.add(4)          # Adds element 4
A.remove(2)       # Removes 2 (raises error if missing)
A.discard(10)     # Removes 10 if present (no error)
print(A.union(B)) # {1, 3, 4, 5}
```



- add() inserts a single element into the set.
- pop() removes and returns an arbitrary set element.
- clear() empties the entire set.
- issubset() and issuperset() test set relations.

Python Data Structures Overview

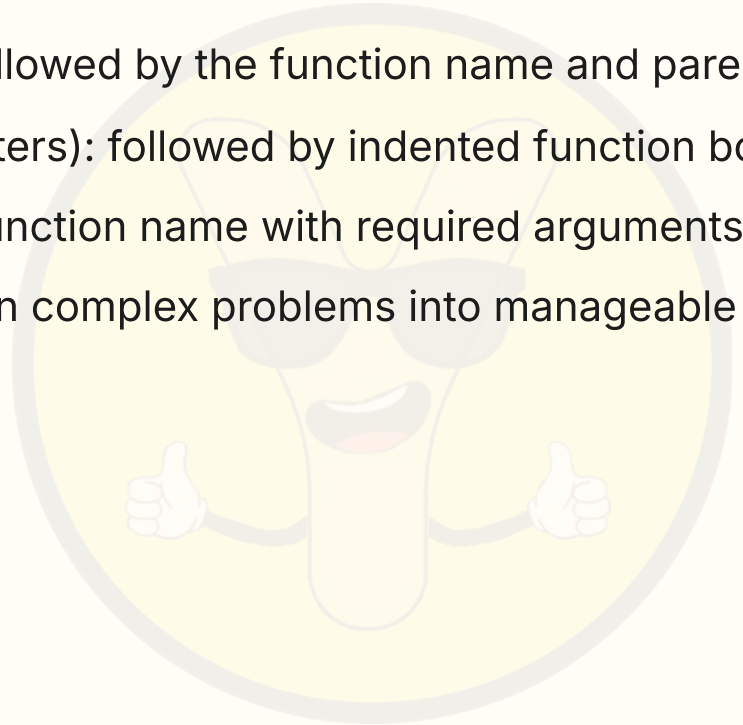


Hierarchical classification of built-in Python data structures.

Functions: Definition and Calling

A function is a reusable block of organized code designed to perform a specific task.

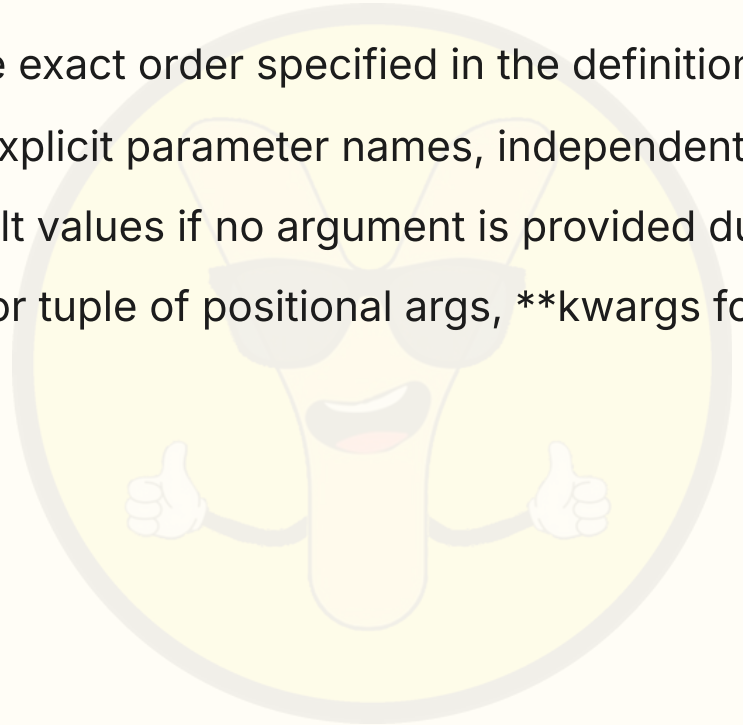
- Definition: Uses the def keyword followed by the function name and parentheses ().
- Syntax: def function_name(parameters): followed by indented function body.
- Calling: Executed by invoking the function name with required arguments.
- Modular Programming: Breaks down complex problems into manageable sub-tasks.



Functions: Arguments

Functions accept data via parameters defined in the function signature.

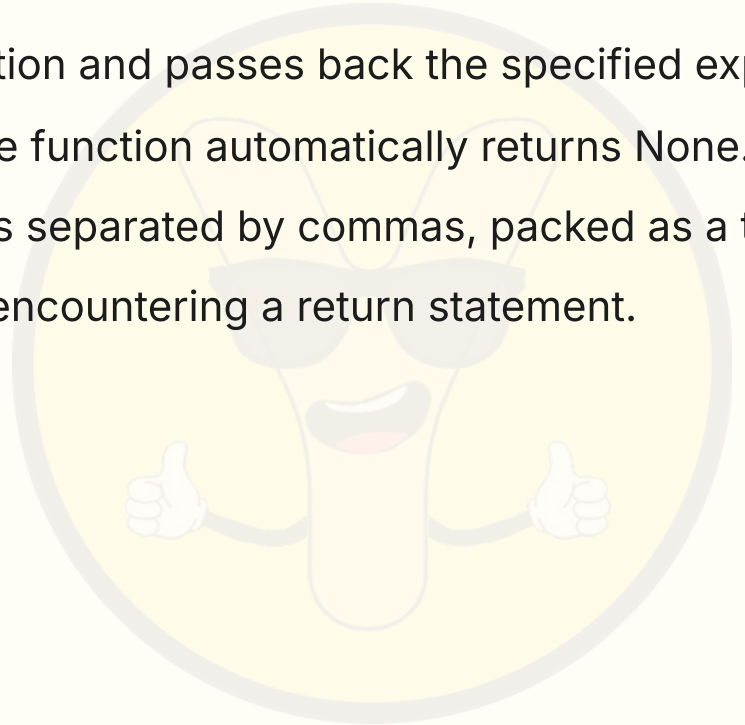
- Positional Arguments: Passed in the exact order specified in the definition.
- Keyword Arguments: Passed with explicit parameter names, independent of order.
- Default Arguments: Assigned default values if no argument is provided during call.
- Variable-length Arguments: *args for tuple of positional args, **kwargs for dictionary of keyword args.



Functions: Return Values

Functions communicate results back to the caller using the return statement.

- The return statement exits the function and passes back the specified expression.
- If no return statement is present, the function automatically returns None.
- Functions can return multiple values separated by commas, packed as a tuple.
- Execution stops immediately upon encountering a return statement.

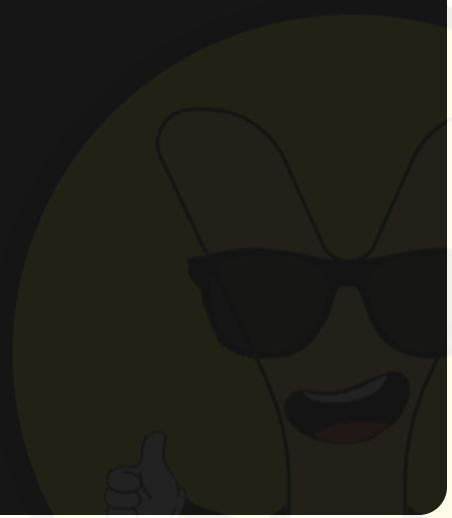


Scope of Variables

```
x = 10  # Global variable

30:
def modify():
    global x
    x = 20
    y = 5  # Local variable
    print(y)

modify()
print(x)  # Prints 20
```

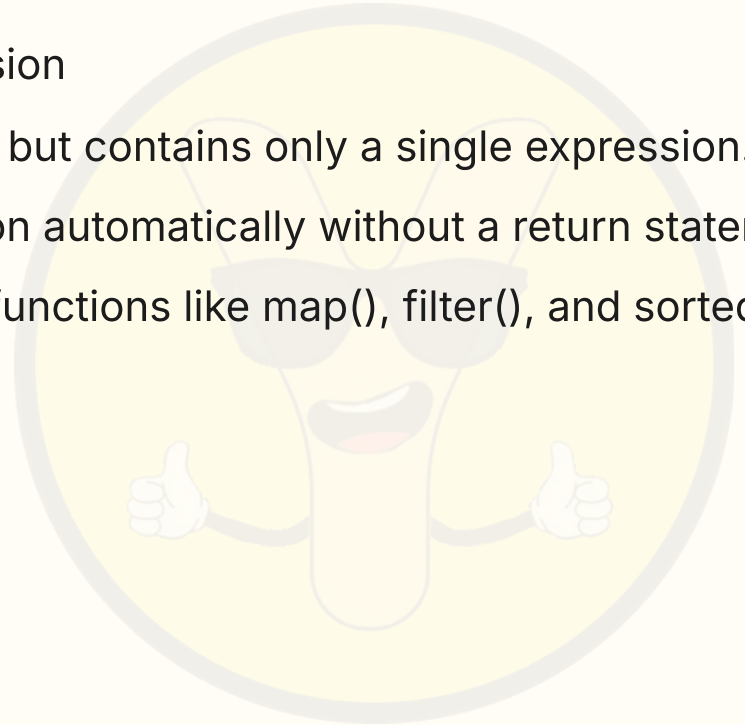


- Local Scope: Variables defined inside a function, accessible only within that function.
- Global Scope: Variables defined outside all functions, accessible module-wide.
- The global keyword allows modifying a global variable inside a local scope.

Lambda Functions

Lambda functions are small, anonymous functions defined using the lambda keyword.

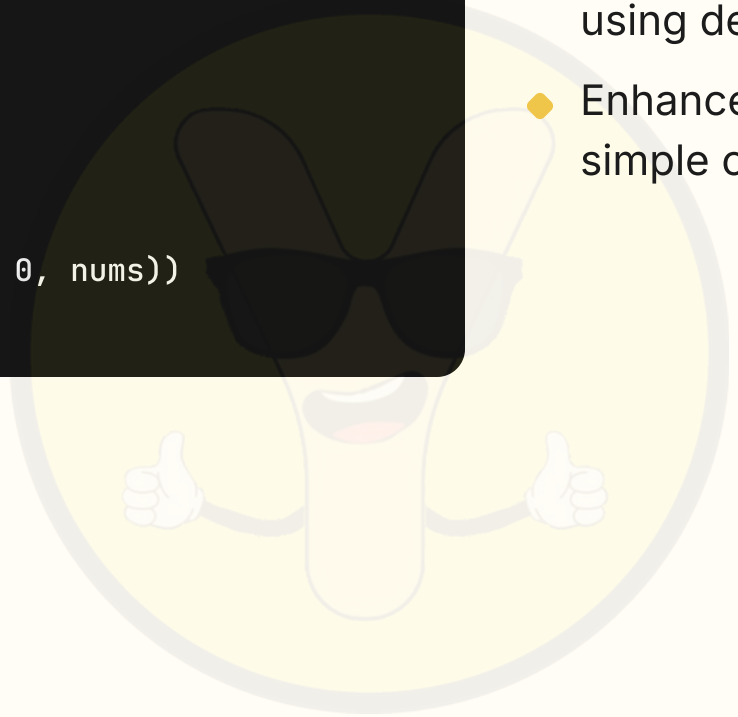
- Syntax: lambda arguments: expression
- Can take any number of arguments but contains only a single expression.
- Evaluates and returns the expression automatically without a return statement.
- Frequently used with higher-order functions like map(), filter(), and sorted().



Lambda Function Example

```
# Lambda to find square
square = lambda x: x ** 2
print(square(5))  # Output: 25

# Using lambda with filter
nums = [1, 2, 3, 4, 5]
evens = list(filter(lambda x: x % 2 == 0, nums))
print(evens)      # Output: [2, 4]
```

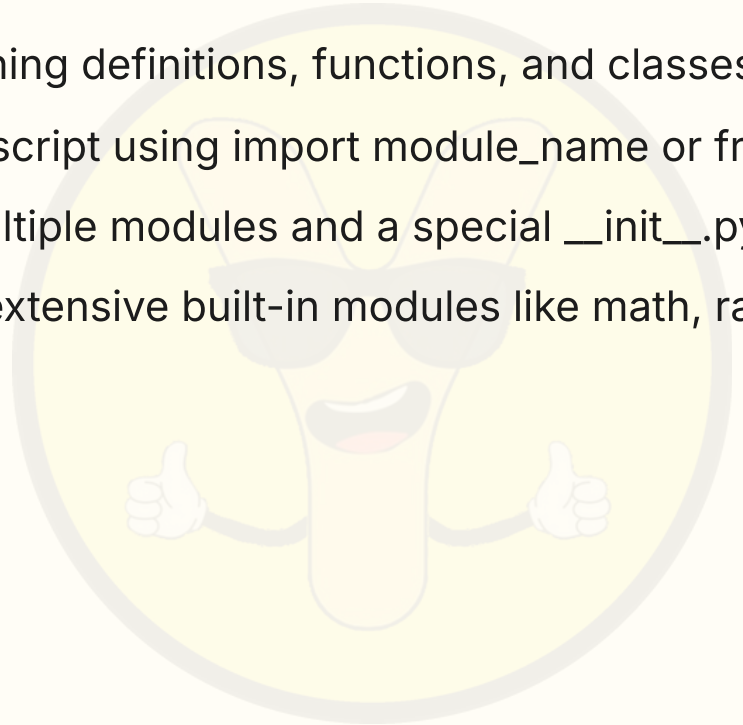


- Eliminates the need to define a formal function using def for short operations.
- Enhances code readability when passing simple operations as arguments.

Modules and Packages

Modules and packages promote code reusability and modular architectural design.

- Module: A single Python file containing definitions, functions, and classes (extension .py).
- Importing: Bringing modules into a script using import module_name or from module import item.
- Package: A directory containing multiple modules and a special __init__.py file.
- Standard Library: Python includes extensive built-in modules like math, random, and os.



Exam corner

- **Q1** Explain mutability with examples, differentiating between lists and tuples.
- **Q2** Write a Python program to demonstrate string slicing and common string methods.
- **Q3** What are dictionaries? Explain key-value mapping and list at least five dictionary methods.
- **Q4** Discuss different types of function arguments with suitable code snippets.
- **Q5** Explain variable scope (Local vs Global) and the use of the global keyword.
- **Q6** What are lambda functions? Write a Python script using lambda with filter() or map().
- **Q7** Explain the concept of Python modules and packages with an example import statement.

Unit 2 done. You are exam-ready.

Revise the Exam corner once more, then open Unit 3 on theycotes.com. Every AKTU subject, unit by unit, free. Notes by Chirag Jain, Ycotes.

