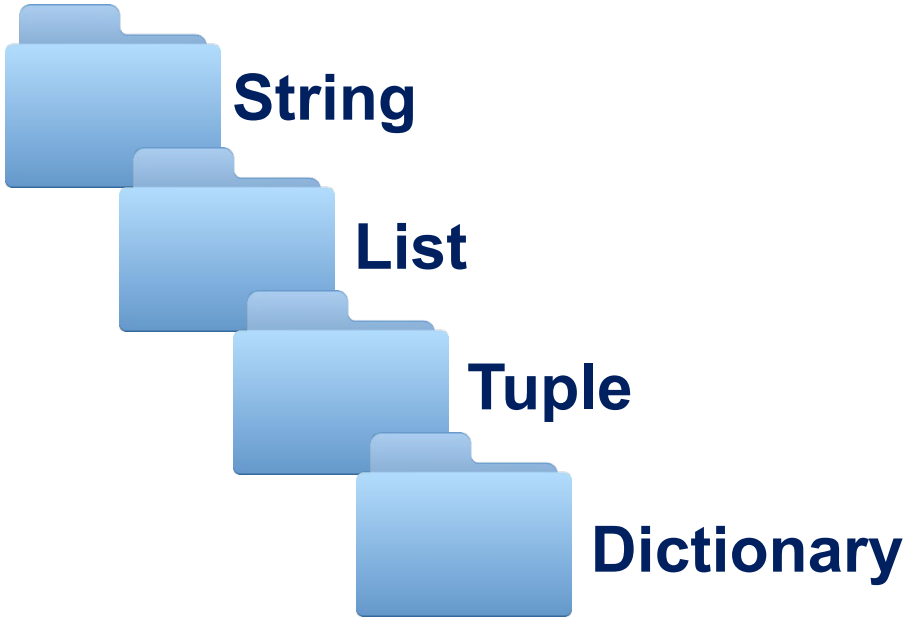
Composite Data Types in Python

📋 Summary Table

Data Type	Real-Life Analogy	Value Type
Integer	Number of people in a room	Whole numbers
Float	Weight on a scale, money	Decimal numbers
Boolean	Light switch, yes/no question	True / False

Review Outline

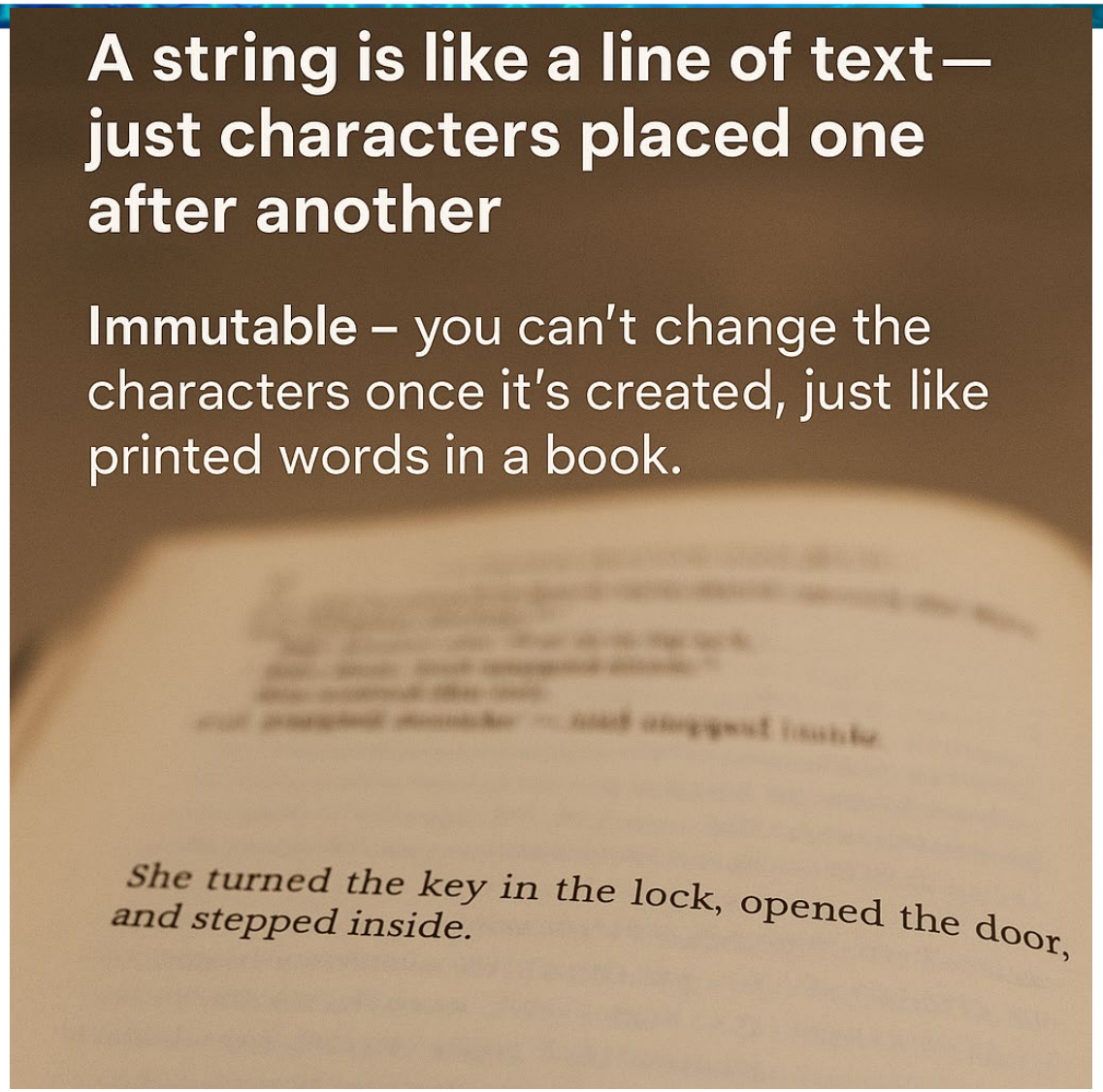


Summary Table:

Data Structure	Real-Life Analogy	Key Traits
String	Sentence or word	Ordered, immutable
List	Shopping list	Ordered, mutable
Tuple	Address (fixed items)	Ordered, immutable
Dictionary	Phonebook / ID card	Key-value pairs, mutable

Python Strings

Real-life analogy: A sentence or a word in a book



https://blog.mandarinportal.com/wp-content/uploads/2013/12/IMG_4227.jpg

Index

```
myStr = "Hello World"
```

Characters	H	e	l	l	o		W	o	r	l	d
Indices	0	1	2	3	4	5	6	7	8	9	10
	–11	–10	–9	–8	–7	–6	–5	–4	–3	–2	–1

We can use [ ] to access particular characters in a string.

```
print(myStr[10])
print(myStr[-1])
```

will print 'd'

```
print(myStr[11])
```

Error

Slicing

```
myStr = "Hello World"
```

Syntax: **[start : finish : step]**

specifies the step size to jump along the sequence

Default Value:

- **start**: beginning
- **finish**: end
- **step**: 1

```
newStr = myStr[:]          # To copy a string
print(newStr)              # Will print 'Hello World'

print(myStr[::-1])         # To reverse a string      # Will print 'dlroW olleH'
```

```
print(myStr[::-1])
```

Traditional Chinese Sign Reading From Right To Left

Modern Chinese Sign Reading From Left To Right

https://mandarinportal.com/reading-left-right-right-left-chinese/

slido

Please download and install the Slido app on all computers you use

What is the output of the following Python program?

```
myStr = “Hello World”
print(myStr[2:-4:2])
```

ⓘ Start presenting to display the poll results on this slide.

Q1: What is the output of the following Python program?

```
myStr = "Hello World"
print(myStr[2:-4:2])
```

A. 'el '
B. 'el o'
✓ C. 'loW'
D. 'loWr'
E. 'lo W'

Characters	H	e	l	l	o		W	o	r	l	d
Indices	0	1	2	3	4	5	6	7	8	9	10
	–11	–10	–9	–8	–7	–6	–5	–4	–3	–2	–1

Basic Operations

```
opStr = "Basic"
```

Length of a string: `len()`

e.g. `len(opStr)` → **5**

Concatenate strings: `+`

e.g. `opStr + " operations"` → **'Basic operations'**

Repeat String: `*`

e.g. `opStr * 3` → **'BasicBasicBasic'**

Membership Operation

Is one string contained in another?

- Operator: `in`
- `a in b`: True if string `a` is contained in string `b`

```
myStr = "abcdefg"
```

`'c' in myStr` → **true**

`'cde' in myStr` → **true**

`'cef' in myStr` → **false**

`myStr in myStr` → **true**

slido

Please download and install the Slido app on all computers you use

What is the output of the following Python program?

ⓘ Start presenting to display the poll results on this slide.

Q2: What is the output of the following Python program?

```
str1 = "ababc"
str2 = "ab"

if str2 * len(str2) in str1:
    print("case1")

elif str2 in str1:
    print("case2")

else:
    print("case3")
```

A. 'case1'
   'case2'

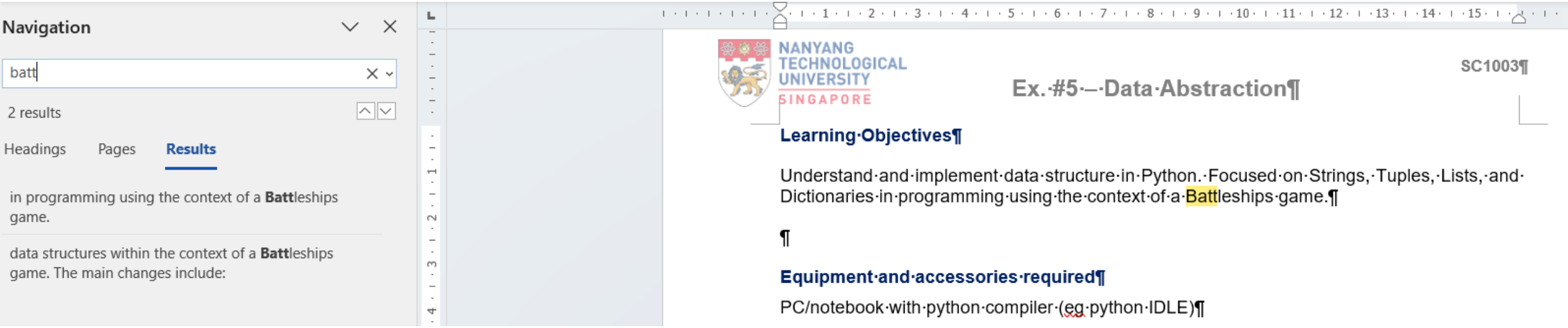
B. 'case2'

✓ C. 'case1'

D. 'case3'

E. 'case2'
   'case3'

Python String find() Method



https://docs.python.org/3/library/string.html

String Method: `find()`

`find()` is another string method.

```
myStr = "Find in a string"
myStr.find('d') → 3
```

- Input: a single character or a string
- Output: the index of the character/string (first seen from left to right)
- If the character/string is not found, –1 is returned

slido

Please download and install the Slido app on all computers you use

What is the output of the following Python program?

```
str1 = "couple“
str2 = “t”
newStr = str1[::str1.find(str2)]
print(newStr)
```

ⓘ Start presenting to display the poll results on this slide.

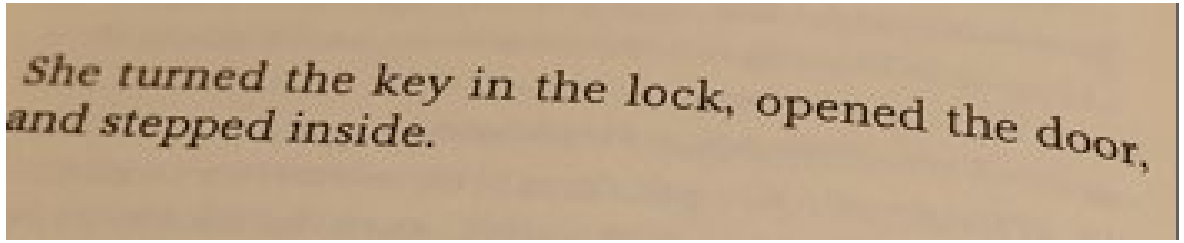
Q3: What is the output of the following Python program?

```
str1 = "couple"
str2 = "t"

newStr = str1[::str1.find(str2)]

print(newStr)
```

A. 'couple'
B. ''
C. 'coupl'
✓ D. 'elpuoc'
E. 'elpuo'

Strings are Immutable

- Strings are immutable, i.e., you cannot change one once you make it.
  - `aStr = 'spam'`
  - `aStr[1] = 1` → Error



- However, you can use it to make another string (copy it, slice it, etc.).
  - `newStr = aStr[:1] + 'l' + aStr[2:]`
  - `newStr` → **'slam'**
  - `aStr` → **'spam'**

Python Lists

Real-life analogy: A shopping list

- A list is like your grocery list—you can **add**, **remove**, or **reorder** items freely.
- It's **mutable** and **ordered**, and can hold any type of data.

```
shopping_list = ["milk", "eggs", "bread"]
shopping_list.append("butter")

shopping_list[2] = "whole wheat bread"

print(shopping_list)
```

['milk', 'eggs', 'whole wheat bread', 'butter']

Creating a List

- As with all data structures, lists have a **constructor**.
- **Constructors** have the same name as the data structures.

`l = list()`	Creates an empty list
`l = list(arg)`	Takes an iterable data structure as an argument and add each item of **`arg`** to the constructed list **`l`**

- **Shortcut**: use of **square brackets `[]`** to indicate explicit items. → `l = […]`

```
aList = list('abc')    # ['a','b','c']
newList = [1, 3.14159, 'a', True]
```

Operations on Lists

- **concatenate**: `+` (only for lists – not string + list)
- **repeat**: `*`
- **indexing**: the `[ ]` operator, e.g., `lst[3]` → 4th item in the list
- **slicing**: `[:]`
- **membership**: the `in` operator
- **length**: the `len()` function

List Methods

Lists are mutable — changes like .append(), .remove(), and item assignment affect the original list.You don’t need return unless you're creating a new list.

```
myList[0] = 'a'        #index assignment
myList.append(e)       // e: element to append
myList.extend(L)       // L: a list
myList.pop(i)          // i: index (default: -1)
myList.insert(i,e)
myList.remove(e)
myList.sort()
myList.reverse()
```

Think of a list as a whiteboard.
When you pass the whiteboard to a friend (the function), they can write directly on it.
After they return it, the new words are already there — no need to ask them to return a "new whiteboard."

```
list1 = ['d', 'c']
list2 = [2,9]
list1.reverse()
list2.reverse()
list1.extend(list2)
print(list1)
print(list2)
```

Python Tutor - Python Online Compiler with Visual AI Help

list of lists

A **list of lists** is like a table or matrix — a list where each element is itself a list.

Real-Life Analogy: Class Attendance Sheet

Each row is a list of student names in a group, and the entire sheet is a list of those rows.

```
class_groups = [
    ["Alice", "Ben", "Charlie"],
    ["Diana", "Ethan", "Fiona"],
    ["George", "Hannah", "Ian"]
]
```

- class_groups[0] gives ["Alice", "Ben", "Charlie"]
- class_groups[1][2] gives "Fiona" (2nd row, 3rd student)

Another Example: 2D MatrixpythonCopyEdit

```
matrix = [
    [1, 2, 3],
    [4, 5, 6],
    [7, 8, 9]
]
```

matrix[0][0] →	1
matrix[1][2] →	6
matrix[2] →	[7, 8, 9]

Use Cases:

- Grids in games (like Battleship or Sudoku)
- Seating arrangements
- Storing coordinates
- Tabular data

slido

Please download and install the Slido app on all computers you use

What is the output of the following Python program?

ⓘ Start presenting to display the poll results on this slide.

Q5: What is the output of the following Python program?

```
list1 = [1, "Python", [3, 4], True]

if 3 in list1:
    list2 = list1[2] * len(list1[2])
    print(list2)

elif [3, 4] in list1:
    print(list1[2][1])

else:
    print(list1[2])
```

A. [3,4,3,4]
B. [3, 4]
C. [4]
✓ D. 4
E. 3
F. [3]

List Comprehension

List comprehension: syntactic structure for concise construction of lists

mark the comprehension with [ ]

```
[ n for n in range(1,5) ]
```

what we collect

Iteration condition
Note that we iterate over a set of values and collect some (in this case all) of them.

[1, 2, 3, 4]

slido

Please download and install the Slido app on all computers you use

What is the output of the following Python program?

```
list1 = ['d', 'c', 'A', 3]
list2 = ['A', 'b', '3']
result = [item for item in list1 if item in list2]
print(result)
```

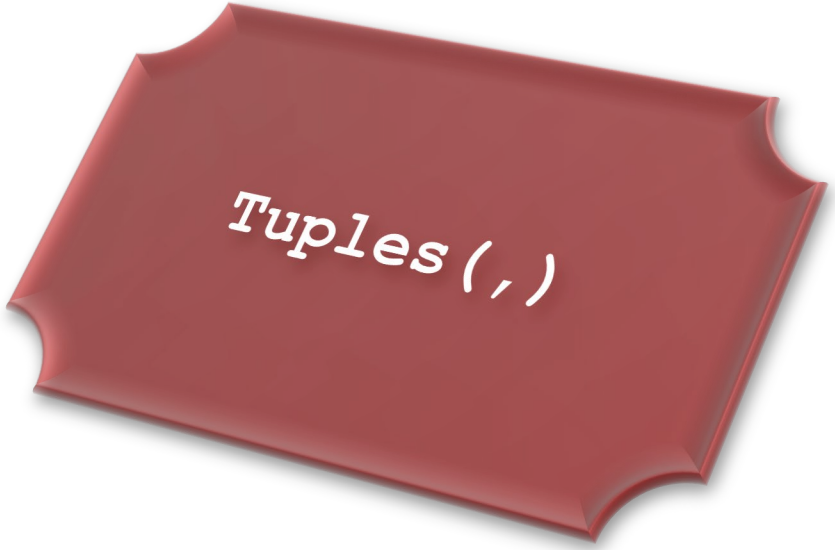
ⓘ Start presenting to display the poll results on this slide.

Q7: What is the output of the following Python program?

```
list1 = ['d', 'c', 'A', 3]
list2 = ['A', 'b', '3']

result = [item for item in list1 if item in list2]

print(result)
```

A. ['3']
B. 'A'
C. [3]
✓ D. ['A']
E. ['A',3]
F. []

Python Tuples



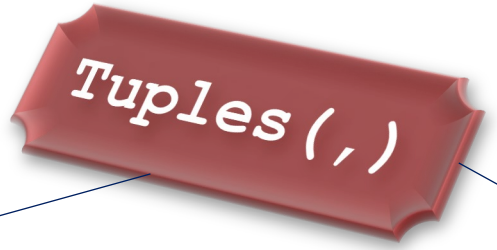
Real-life Analogy: Your birth record

- A **tuple** is like a fixed set of information:
  - (name, date of birth, gender, parent's name)
- It's **structured**, **ordered**, and **unchangeable**, just like what's written on official documents.

```
person_info = ("Jamie Tan", "2000-01-01", "Female",
"Tan Mei Lin")
```

- This stays constant. A tuple is like a read-only container—once it's created, you cannot modify, add, or remove its elements.

Tuples



Tuples are **immutable** lists.

Why Immutable Lists?

- Provides a data structure with some integrity and some permanency
- To avoid accidentally changing one

They are designated with `(,)`.

Example:

```
myTuple = (1, 'a', 3.14, True)
```

Lists vs. Tuples

Everything that works for a list works for a tuple **except** methods that modify the tuple.

What works?

- **indexing**
- **slicing**
- `len()`
- `print()`

What doesn't work?

Mutable methods

- `append()`
- `extend()`
- `remove(), etc.`

slido

Please download and install the Slido app on all computers you use

What is the output of the following Python program?

```
myTuple = (4, 2, 3, [6, 5])
myTuple[0] = 7
print(myTuple)
```

ⓘ Start presenting to display the poll results on this slide.

Q8: What is the output of the following Python program?

```
myTuple = (4, 2, 3, [6, 5])

myTuple[0] = 7

print(myTuple)
```

A. (4,2,3,[6,5])
B. (7,2,3,[6,5])
C. []
✓ D. Error

slido

Please download and install the Slido app on all computers you use

What is the output of the following Python program?

```
tuple1 = (3, 2, 6, ['a', 'b'])
tuple2 = tuple1[::-2]
print(tuple2,tuple2[0][0])
```

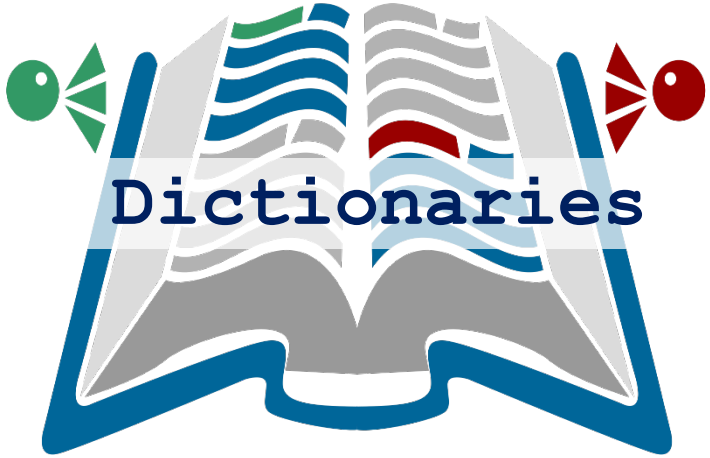
ⓘ Start presenting to display the poll results on this slide.

Q9: What is the output of the following Python program?

```
tuple1 = (3, 2, 6, ['a', 'b'])
tuple2 = tuple1[::-2]
print(tuple2,tuple2[0][0])
```

✓ A.(['a', 'b'], 2) a
B.(3, 2, 6, ['a', 'b’]) 3
C.(3, 6) 3
D.(['a', 'b’], 6, 2, 3) a
E.Error
F.None of the options

Python Dictionary



Real-life analogy: A phonebook or student ID card

Real-life analogy: A phonebook or student ID card

- A dictionary stores **key-value pairs**, like a contact name and their phone number.
- It's **unordered (before Python 3.7)**, **mutable**, and allows fast lookups.

```
student = {"name": "Alice", "age": 20, "major": "CS"}
```

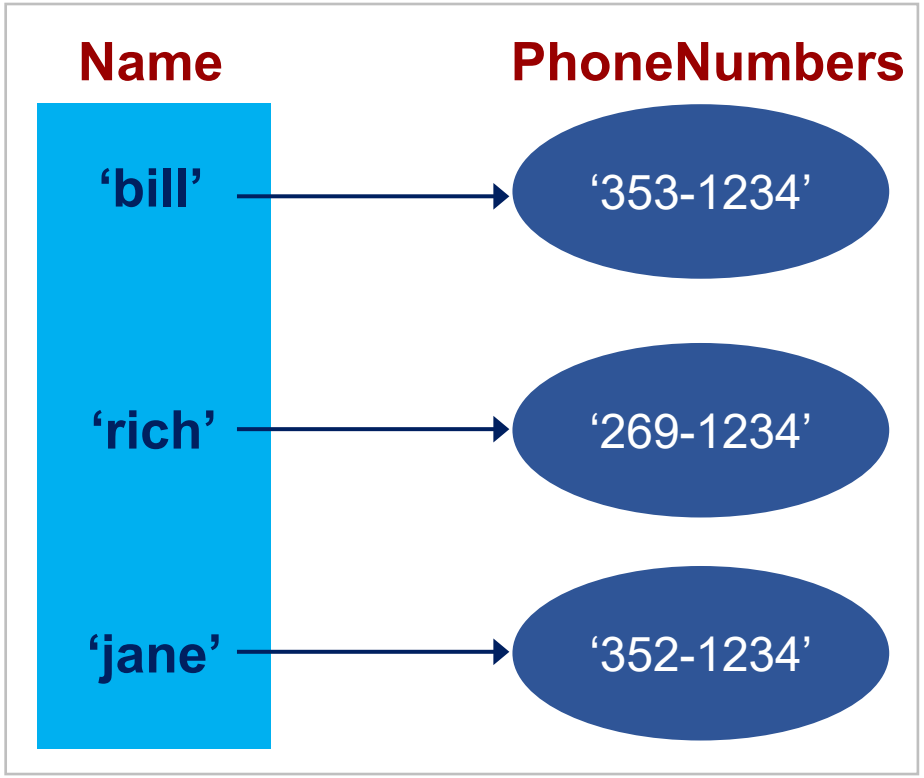
You can look up "name" to get "Alice", just like using a student ID card to find personal info.

Python Dictionary

{ } marker: used to create a dictionary

: marker: used to create **key:value** pairs

```
contacts = {'bill': '353-1234',
            'rich': '269-1234',
            'jane': '352-1234'}

print(contacts)  →  {'jane': '352-1234',
                     'bill': '353-1234',
                     'rich': '269-1234'}
```



slido

Please download and install the Slido app on all computers you use

What is the output of the following Python program?

ⓘ Start presenting to display the poll results on this slide.

Q10: What is the output of the following Python program?

```
contacts = {
    'bill': '353-1234',
    'rich': '269-1234',
    'jane': '352-1234'
}

print(len(contacts))
```

✓ A. 3
B. 6
C. Error

What are Keys and Values?

```
Key:value
```

Must be immutable

Strings, Integers, Tuples

Allowed

NOT Allowed

Lists

Can be anything

Use Keys to access the values

Methods on Dictionaries

`myDict.items()` → return all the **key:value** pairs

`myDict.keys()` → return all the keys

`myDict.values()` → return all the values

`myDict.clear()` → empty the dictionary

`myDict.update(yourDict)` → for each key in `yourDict`, update `myDict` with that **key:value** pair

slido

Please download and install the Slido app on all computers you use

What is the output of the following Python program?

ⓘ Start presenting to display the poll results on this slide.

Q11: What is the output of the following Python program?

```
tuple1 = (1, 2, [1], [1, 2])

dict1 = {
    'a': [1],
    'b': [2],
}

for key, value in dict1.items():
    if value in tuple1:
        dict1[key] = 'hit'

print(dict1)
```

A. {'a':[1], 'b':[2]}
B. {'a':'hit', 'b':'hit'}
C. {'hit':[1], 'hit':[2]}
D. {'hit':[1], 'b':[2]}
✓ E. {'a':'hit', 'b':[2]}
F. Error

dict1.update(dict2)

- The update() method updates dict1 by **adding or modifying** the key-value pairs from dict2.
- How update() works:
  - If a key in dict2 exists in dict1, the value for that key in dict1 will be **overwritten** by the value from dict2.
  - If a key in dict2 does **not** exist in dict1, it will be **added** to dict1.

slido

Please download and install the Slido app on all computers you use

What is the output of the following Python program?

ⓘ Start presenting to display the poll results on this slide.

Quiz 1 (Sep. 27)

Format:

- The quiz will consist of 30 multiple-choice questions (MCQs). Single-answer MCQs (default) / Multiple-correct-answer MCQs (if specified).

Sample of Multiple-correct-answer MCQs

Which of the following functions **ARE NOT**
(Select all that apply)

- ☐ A.
- ☐ B.
- ☐ C.
- ☐ D.

- The duration is 1 hour.

Instructions:

Please refer to the course announcements and information for full details.

- The quiz will be conducted **in-person** in the labs.
- Make sure to bring a **pen**.
- This is a **closed-book** quiz—no external resources (books, notes, etc.) are allowed.
- Each student will have an **individual timer** that starts when the quiz begins.

- You may either click the **submit** button when you finish, or the quiz will be **automatically submitted** when your time is up.
- **Important:** Students arriving more than **10 minutes late** will not be allowed to join the quiz.

Grading & Feedback:

- Your scores will be available to view in the Grade Centre by the end of **October 4**.

- Feedback, including quiz statistics such as the average, median, and more, will be posted on the course site.

Good luck and study well!

CCDS STUDENT-LED TUTORIAL GROUP (STG)

Semester 1AY 25/26 Class Schedule

SC1003/CE1103/CZ1103 INTRO TO COMP THINKING AND PROGRAMMING
The ARC LHN-TR+21
Every Wed, 6:30-8:30 PM
Peer Tutor: Hyun Bin Kim

SC1004/CE1104/CZ1104 LINEAR ALGEBRA FOR COMPUTING
The ARC LHN-TR+25
Every Wed, 6:30-8:30 PM
Peer Tutor: Luar Shui Yan

SC2001/CE2101/CZ2101 ALGORITHM DESIGN AND ANALYSIS
The ARC LHN-TR+24
Every Friday, 6:30-8:30 PM
Peer Tutor: Savanur Akash

SC2005/CE2005/CZ2005 OPERATING SYSTEMS
The ARC LHN-TR+10
Every Friday, 6:30-8:30 PM
Peer Tutor: Bui Gia Nhat Minh

SC1004/CE1104/CZ1104 LINEAR ALGEBRA FOR COMPUTING
NS TR+6
Every Tue, 6:30-8:30 PM
Peer Tutor: Kim Hyun Bin

SC1005/CE1105/CZ1105 DIGITAL LOGIC
The ARC LHN-TR+24
Every Wed, 6:30-8:30 PM
Peer Tutor: Vivek Shrey

SC1005/CE1105/CZ1105 DIGITAL LOGIC
NS TR+7
Every Thu, 6:30-8:30 PM
Peer Tutor: Jeremy Philip

IMPORTANT

- Classes will start from **Teaching Week 5 onwards!**
- Classes will be closed if the attendance is **less than 5 students.**

Register before turning up! **FREE!**

tinyurl.com/ccdsstudentcare

CCDS Student Affairs Office in Collaboration with SCDS Club
Exclusive to CCDS Students ONLY!