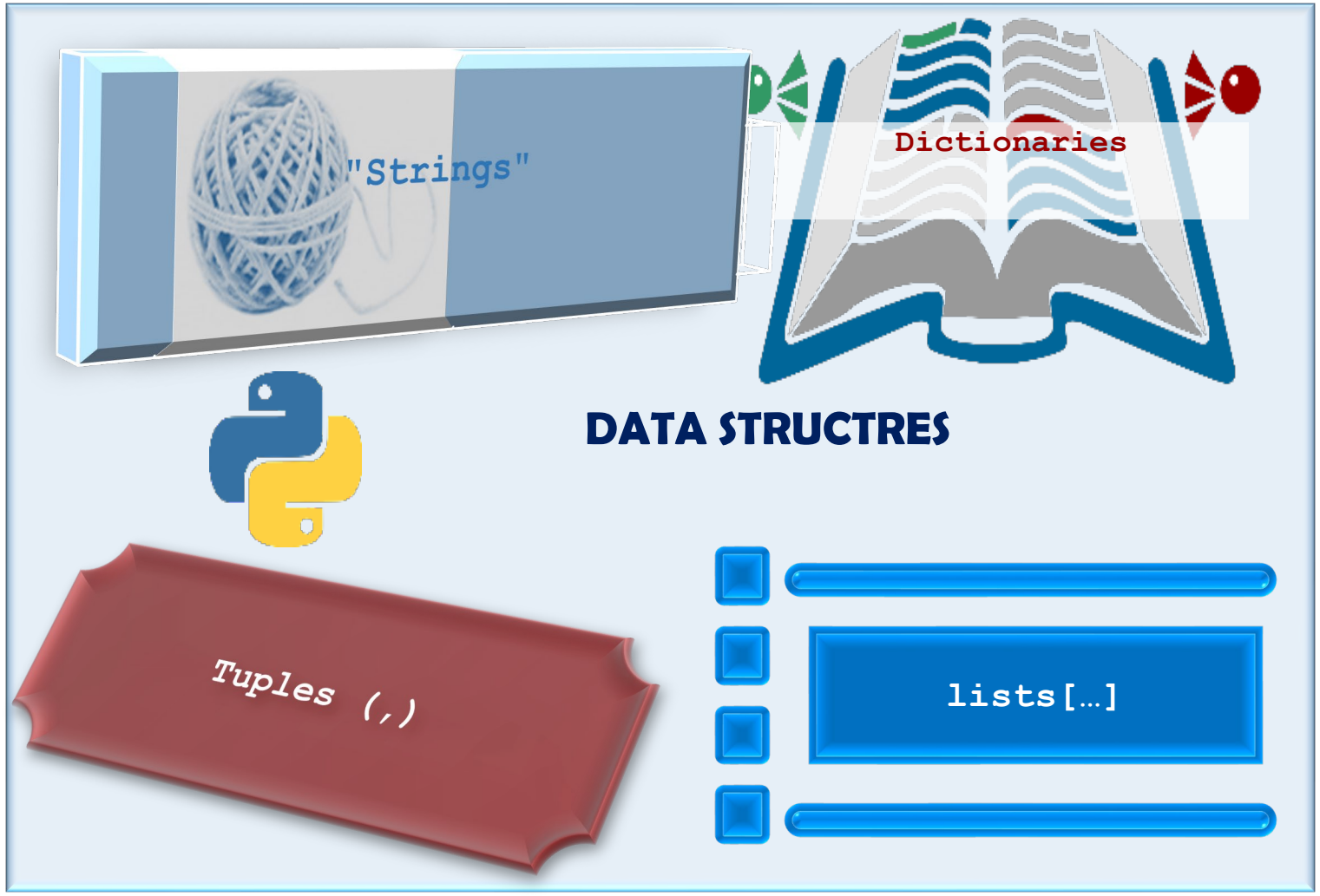
Discussion questions #5
String, Data structure (Data abstraction)


"Strings"
Dictionaries
DATA STRUCTRES
Tuples (,)
lists[…]

Q1 String operation

String functions

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

https://www.w3schools.com/python/python_ref_string.asp

partition()	Returns a tuple where the string is parted into three parts
replace()	Returns a string where a specified value is replaced with a specified value
rfind()	Searches the string for a specified value and returns the last position of where it was found
rindex()	Searches the string for a specified value and returns the last position of where it was found
rjust()	Returns a right justified version of the string
rpartition()	Returns a tuple where the string is parted into three parts
rsplit()	Splits the string at the specified separator, and returns a list
rstrip()	Returns a right trim version of the string
split()	Splits the string at the specified separator, and returns a list

String Method: `split()`

- The split() method returns a list of all the words in the string, using str as the separator (splits on all whitespace if left unspecified)
  - Default split-character: **white space**.
  - The string method, **`split()`**, returns a list.

```
splitLst = 'this is a test'.split()
print(splitLst)
```

['this', 'is', 'a', 'test']

```
splitLst = 'this,is,a,test'.split(',')
print(splitLst)
```

['this', 'is', 'a', 'test']

Definition and Usage

The `format()` method formats the specified value(s) and insert them inside the string's placeholder.

The placeholder is defined using curly brackets: `{}`. Read more about the placeholders in the Placeholder section below.

The `format()` method returns the formatted string.

Syntax

```
string.format(value1, value2...)
```

The values can be of any data type.

https://www.w3schools.com/python/ref_string_format.asp

Q1 **String operation**

Uppercase: HELLO, WORLD! WELCOME TO PYTHON PROGRAMMING.

Lowercase: hello, world! welcome to python programming.

Capitalized: Hello, world! welcome to python programming.

Position of 'World': 7

Replaced string: Hello, Universe! Welcome to Python programming.

List of words: ['Hello,', 'World!', 'Welcome', 'to', 'Python', 'programming.']

Joined string: Hello, World! Welcome to Python programming.

Q2 list methods

Python Lists

Python List is an **ordered sequence of items**.

Recall

We have already covered a type of sequence: **Strings**

- A string is a sequence of characters.

dad ≠ add

[255, 0,0] ≠ [0,0, 255]

Lists: Differences with Strings

- Lists can contain **a mixture of python objects (types)**; strings can **only hold characters**.

  E.g. `l = [1, 'bill', 1.2345, True]`

- Lists are **mutable**; their values can be changed, while strings are **immutable**.

- Lists are designated with `[ ]`, with elements separated by commas; strings use "".

Built-in List Functions & Methods

Sr.No.	Function with Description
1	cmp(list1, list2) Compares elements of both lists.
2	len(list) Gives the total length of the list.
3	max(list) Returns item from the list with max value.
4	min(list) Returns item from the list with min value.
5	list(seq) Converts a tuple into list.

Sr.No.	Methods with Description
1	list.append(obj) Appends object obj to list
2	list.count(obj) Returns count of how many times obj occurs in list
3	list.extend(seq) Appends the contents of seq to list
4	list.index(obj) Returns the lowest index in list that obj appears
5	list.insert(index, obj) Inserts object obj into list at offset index
6	list.pop(obj=list[-1]) Removes and returns last object or obj from list
7	list.remove(obj) Removes object obj from list
8	list.reverse() Reverses objects of list in place
9	list.sort([func]) Sorts objects of list, use compare func if given

https://www.tuto

Built-in Functions

Syntax

Following is the syntax for **len()** method –

```
len(list)
```

Parameters

list – This is a list for which number of elements to be counted.

Return Value

This method returns the number of elements in the list.

`min(lst)`	Minimum element in the list
`max(lst)`	Maximum element in the list
`sum(lst)`	Sum of the elements, numeric only

Q2 list methods

Recorded temperatures: [72, 68, 75, 70, 69, 74, 73]
[68, 69, 70, 72, 73, 74, 75]
After appending new temperature 71: [68, 69, 70, 72, 73, 74, 75, 71]
The maximum temperature recorded is: 75
The minimum temperature recorded is: 68
after apend pre_temperatures:[51, 53, 56, [68, 69, 70, 72, 73, 74, 75, 71]]
After appending temperatures for another week: [68, 69, 70, 72, 73, 74, 75, 71, 95, 97, 94]
The average temperature is: 78.0

Q3 list of lists, list slicing

Q3 list of lists, list slicing

['Meeting', 'Emails', 'Project Work']

['Testing', 'Meeting', 'Documentation']

Planning

Q4 List Comprehension

List Comprehension

[expression **for**-clause [*condition*]*]*

https://www.w3schools.com/python/python_lists_comprehension.asp

List comprehension offers a shorter syntax when you want to create a new list based on the values of an existing list.

- *Example:*
- *Based on a list of fruits, you want a new list, containing only the fruits with the letter "a" in the name.*

```
fruits = ["apple", "banana", "cherry", "kiwi", "mango"]
newlist = []

for x in fruits:
  if "a" in x:
    newlist.append(x)

print(newlist)
```

```
fruits = ["apple", "banana", "cherry", "kiwi", "mango"]

newlist = [x for x in fruits if "a" in x]

print(newlist)
```

The Syntax

```
newlist = [expression for item in iterable if condition == True]
```

The return value is a new list, leaving the old list unchanged.

Condition

The *condition* is like a filter that only accepts the items that valuate to `True`.

Example

Only accept items that are not "apple":

```
newlist = [x for x in fruits if x != "apple"]
```

Other Examples

```
[x + y for x in range(1,5) for y in range (1,4)]
```

It is as if we had done the following:

```
myList = [ ]
for x in range (1,5):
  for y in range (1,4):
    myList.append(x+y)
```

[2, 3, 4, 3, 4, 5, 4, 5, 6, 5, 6, 7]

slido

What is the output of the following code correct?

```
list1 = ["c", "Java", "Python", "C++"]
print([p.upper() for p in list1 if len(p)>=4])
```

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

```python
list1 = ["C", "Java", "Python", "C++"]

print([ p.upper()for p in list1 if len(p)>=4])
```

['JAVA', 'PYTHON']

Q4 Python list comprehension

- Write a Python program, in the fewest number of lines possible, which creates a list of all the square numbers: x^2 (where $1<=x<=100$) that are divisible by 3.

Q4 Python list comprehension

- Write a Python program, in the fewest number of lines possible, which creates a list of all the square numbers: x^2 (where 1<=x<=100) that are divisible by 3.

```
list1 =   [ x**2   for x in range(1,101) if x**2 % 3 == 0  ]
```

List comprehensions provide a concise way to create lists.

It consists of brackets containing an expression followed by a for clause, then zero or more for or if clauses. The expressions can be anything, meaning you can put in all kinds of objects in lists.

The result will be a new list resulting from evaluating the expression in the context of the for and if clauses which follow it.

The list comprehension always returns a result list.

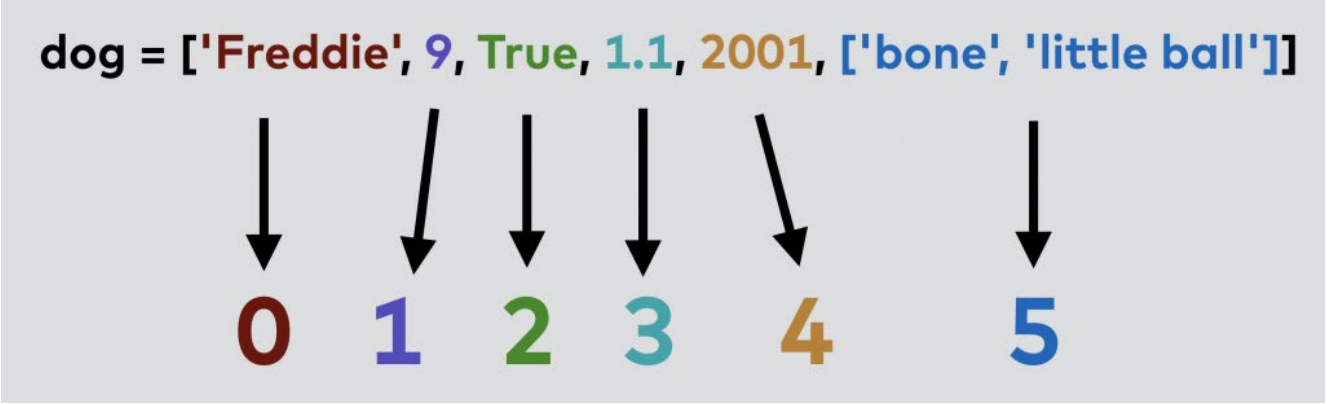
```
new_list = [expression(i)  for i in old_list]
new_list = [expression(i)  for i in old_list if filter(i)]
new_list = [expression(i, j)  for i in old_list1 for j in old_list2 if filter(i) if filter(j)]
```

Q5 Dictionary

List vs Dictionary

```
dog_name = 'Freddie'
age = 9
is_vaccinated = True
height = 1.1
birth_year = 2001
```

Freddie has two belongings: a bone and a little ball.



dog_dict = {'name': 'Freddie', 'age': 9, 'is_vaccinated': True, 'height': 1.1, 'birth_year': 2001, 'belongings': ['bone', 'little ball']}

In a dictionary you can attribute a unique key for each of these values, so you can understand better that what value stands for what.

https://data36.com/python-built-in-functions-methods-python-data-science-basics-3/

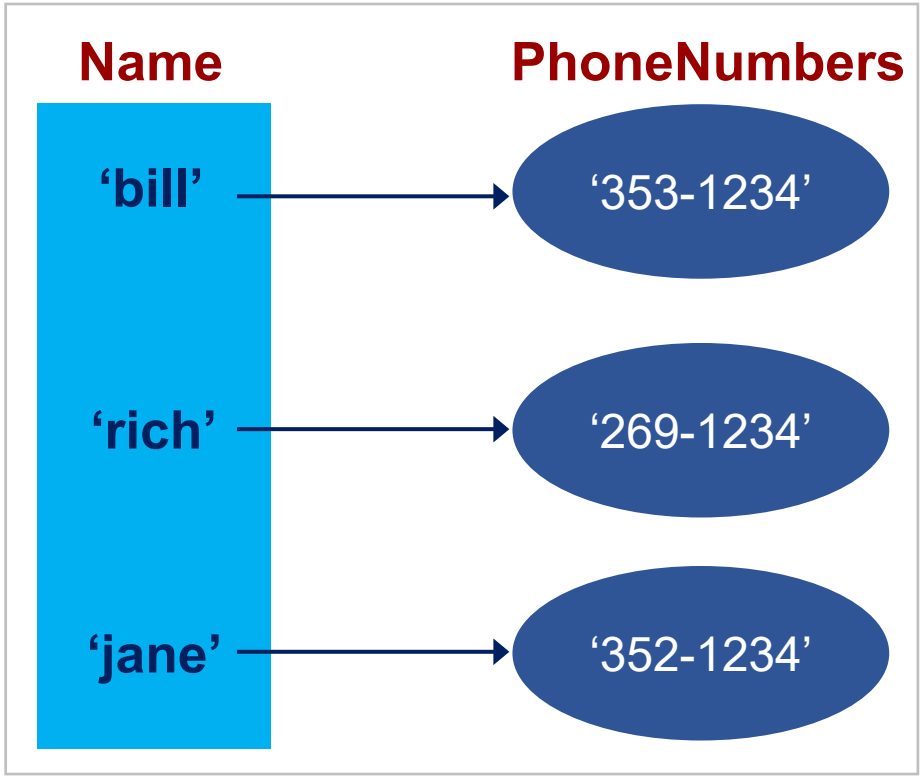
Python Dictionary

{ } 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

What is the output of the following code?

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

```
data = {'MMM':2.3, 'TSL':25, 'AIA':3.6}
for i in data:
    if i == 'TSL':
        print(data[i])
```

25

```
# Define a dictionary representing a collection of books in a library
library = {
    "978-0143127741": {
        "title": "To Kill a Mockingbird",
        "author": "Harper Lee",
        "year": 1960,
        "genre": "Fiction"
    },
    "978-0439023481": {
        "title": "The Hunger Games",
        "author": "Suzanne Collins",
        "year": 2008,
        "genre": "Dystopian"
    },
    "978-0307277671": {
        "title": "The Road",
        "author": "Cormac McCarthy",
        "year": 2006,
        "genre": "Post-apocalyptic"
    }
}
```

```
# Print the entire library dictionary
print("Library collection:")
for isbn, book in library.items():
    print(f"ISBN: {isbn}, Book: {book}")
```

```
Library collection:
ISBN: 978-0143127741, Book: {'title': 'To Kill a Mockingbird', 'author': 'Harper Lee', 'year': 1960, 'genre': 'Fiction'}
ISBN: 978-0439023481, Book: {'title': 'The Hunger Games', 'author': 'Suzanne Collins', 'year': 2008, 'genre': 'Dystopian'}
ISBN: 978-0307277671, Book: {'title': 'The Road', 'author': 'Cormac McCarthy', 'year': 2006, 'genre': 'Post-apocalyptic'}
```

The get() function in a dictionary is a method used to retrieve the value for a given key. It is particularly useful because, unlike directly accessing a key, it doesn't raise an error if the key is not found; instead, it returns None or a default value that you can specify.

```
my_dict = {"apple": 1, "banana": 2, "cherry": 3}
# Retrieve the value for the key "banana"
value = my_dict.get("banana")
print(value)  # Output: 2
# If the key does not exist, return None
value = my_dict.get("orange")
print(value)  # Output: None
# You can also specify a default value if the key is not found
value = my_dict.get("orange", 0)
print(value)  # Output: 0
```

```
# Access information about a specific book by its ISBN
isbn_to_lookup = "978-0143127741"
book_info = library.get(isbn_to_lookup, "Book not found")
print(f"\nDetails of the book with ISBN {isbn_to_lookup}:
{book_info}")
```

Details of the book with ISBN 978-0143127741: {'title': 'To Kill a Mockingbird', 'author': 'Harper Lee', 'year': 1960, 'genre': 'Fiction'}

```
# Add a new book to the library
new_book_isbn = "978-0553573404"
new_book = {
    "title": "A Game of Thrones",
    "author": "George R. R. Martin",
    "year": 1996,
    "genre": "Fantasy"
}
library[new_book_isbn] = new_book
print(f"\nAdded new book with ISBN {new_book_isbn}: {library[new_book_isbn]}")
```

```
Added new book with ISBN 978-0553573404: {'title': 'A Game of Thrones', 'author': 'George R. R. Martin', 'year': 1996, 'genre': 'Fantasy'}
```

The pop() function in a dictionary is used to remove a key-value pair and return the value associated with the specified key. If the key does not exist, you can provide a default value to return, avoiding an error.

```
my_dict = {"apple": 1, "banana": 2, "cherry": 3}

# Remove and return the value for the key "banana"
value = my_dict.pop("banana")
print(value)  # Output: 2
print(my_dict)  # Output: {"apple": 1, "cherry": 3}

# Attempt to remove a non-existent key, with a
default value
value = my_dict.pop("orange", 0)
print(value)  # Output: 0
print(my_dict)  # Output: {"apple": 1, "cherry": 3}
```

```
# Remove a book from the library
isbn_to_remove = "978-0307277671"
removed_book = library.pop(isbn_to_remove, "Book not found")
print(f"\nRemoved book with ISBN {isbn_to_remove}: {removed_book}")
```

Removed book with ISBN 978-0307277671: {'title': 'The Road', 'author': 'Cormac McCarthy', 'year': 2006, 'genre': 'Post-apocalyptic'}

```
# Update the details of an existing book
isbn_to_update = "978-0439023481"
library[isbn_to_update]["year"] = 2009  # Updating the year of publication
print(f"\nUpdated book details with ISBN {isbn_to_update}: {library[isbn_to_update]}")
```

Updated book details with ISBN 978-0439023481: {'title': 'The Hunger Games', 'author': 'Suzanne Collins', 'year': 2009, 'genre': 'Dystopian'}

List Comprehension

```
# Search for books by a specific author
author_to_search = "Harper Lee"
books_by_author = [book for book in library.values() if book["author"] == author_to_search]
print(f"\nBooks by {author_to_search}: {books_by_author}")
```

Books by Harper Lee: [{'title': 'To Kill a Mockingbird', 'author': 'Harper Lee', 'year': 1960, 'genre': 'Fiction'}]

Dictionary Comprehension

```
# Filter books by genre
genre_to_filter = "Fiction"
books_in_genre = {isbn: book for isbn, book in library.items() if book["genre"] == genre_to_filter}
print(f"\nBooks in the genre '{genre_to_filter}': {books_in_genre}")
```

Books in the genre 'Fiction': {'978-0143127741': {'title': 'To Kill a Mockingbird', 'author': 'Harper Lee', 'year': 1960, 'genre': 'Fiction'}}