3.Hands-on-practice

DataTypes

Int- Use integers for whole numbers

Float- Use float for real numbers with decimal

Complex- Use complex for real and imaginary numbers



Boolean – Boolean list contains elements of true or false

String- A string is a sequence of characters between single or double quote

List- Lists are used to store multiple items in a single variable.



Tuple- Tuples are used to store multiple items in a single variable.

Set- Set used to store several items in a single variable

The main difference between list, set and tuple:

List – Ordered, Mutable and can be modified, allows duplicate values in multiple occurrences

Set – Unordered, mutable and unique

Tuple-Ordered, Immutable, their contents cannot be changed after creation and allow duplicates

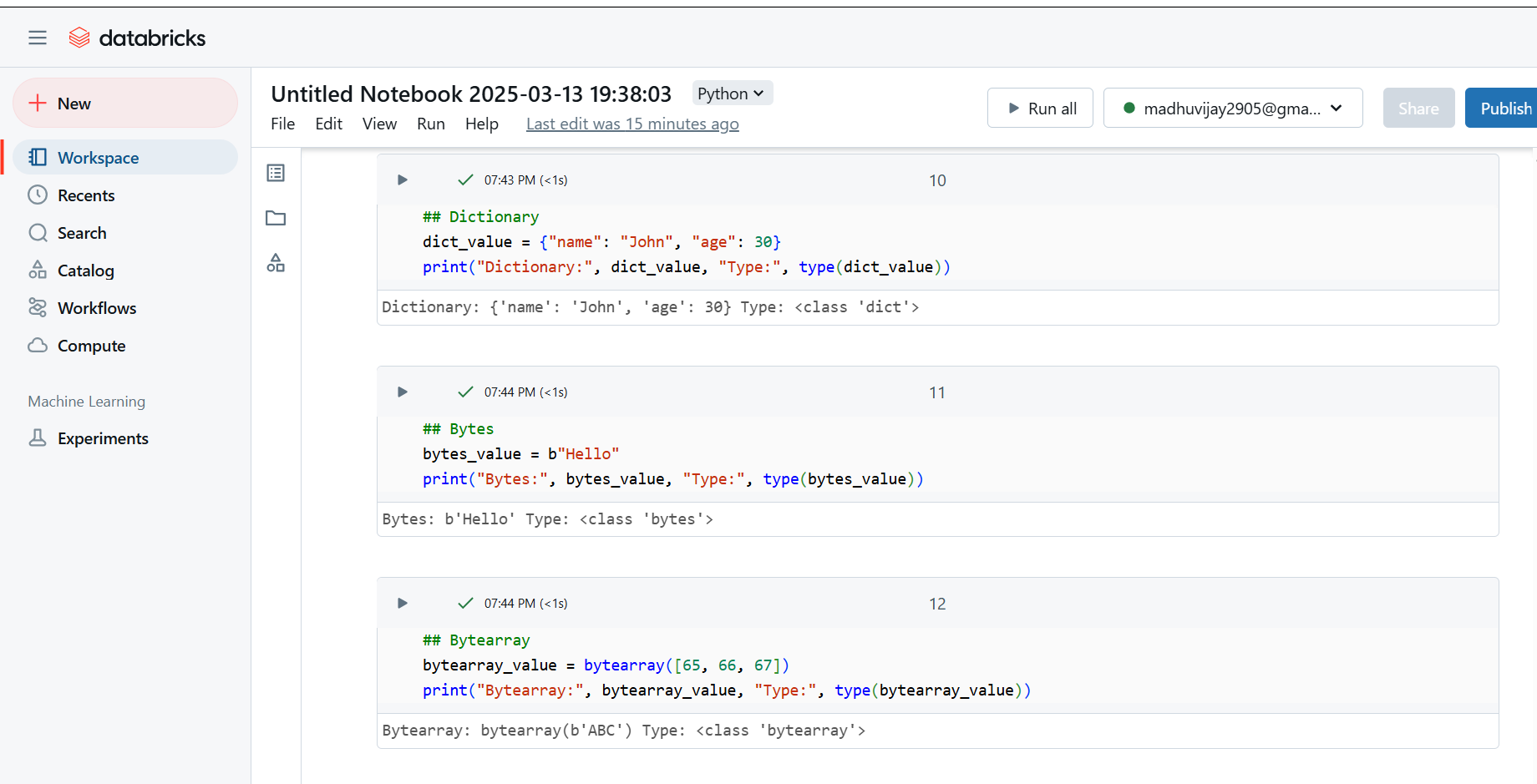
Frozenset- Frozen set don’t change after they’ve created.



Dictionary- Mutable data structures that allow you to store key-value pairs and does not allow duplicate values

Bytes-

Bytearray-



2.Keywords

| **Keyword** | **Explanation** | **Example** |
| --- | --- | --- |
| False | Boolean value representing falsehood. | is\_active = False print(is\_active) # Output: False |
| None | Represents the absence of a value. | result = None print(result is None) # Output: True |
| True | Boolean value representing truth. | is\_valid = True print(is\_valid) # Output: True |
| \_\_peg\_parser\_\_ | Internal keyword (not commonly used). | *No common example* |
| and | Logical AND operator. | if 5 > 2 and 3 < 4: print("Both conditions are True") |
| as | Used for aliasing modules. | import numpy as np arr = np.array([1, 2, 3]) |
| assert | Debugging tool to check conditions. | x = 5 assert x > 0, "x should be positive" |
| async | Declares an asynchronous function. | async def fetch\_data(): return "Data fetched" |
| await | Waits for an asynchronous function to complete. | async def main(): result = await fetch\_data() print(result) |
| break | Exits a loop immediately. | for i in range(5): if i == 3: break print(i) |
| class | Defines a class. | class Car: def \_\_init\_\_(self, brand): self.brand = brand |
| continue | Skips the current iteration and moves to the next one. | for i in range(5): if i == 2: continue print(i) |
| def | Defines a function. | def add(a, b): return a + b |
| del | Deletes an object or variable. | x = [1, 2, 3] del x[1] print(x) # Output: [1, 3] |
| elif | Else if condition in an if-else block. | x = 10 if x > 10: print("Greater") elif x == 10: print("Equal") |
| else | Executes if if condition is false. | if x > 0: print("Positive") else: print("Negative") |
| except | Catches exceptions. | try: x = 1 / 0 except ZeroDivisionError: print("Cannot divide by zero") |
| finally | Executes code regardless of exception occurrence. | try: x = 1 / 0 except: print("Error") finally: print("Execution done") |
| for | Looping construct. | for i in range(3): print(i) |
| from | Imports specific parts of a module. | from math import sqrt print(sqrt(16)) # Output: 4.0 |
| global | Declares a global variable inside a function. | def modify\_global(): global x x = 10 |
| if | Conditional statement. | if x > 5: print("Greater") |
| import | Imports a module. | import os print(os.name) |
| in | Checks membership in a sequence. | if "a" in "apple": print("Found") |
| is | Checks object identity. | if x is None: print("x is None") |
| lambda | Anonymous function. | square = lambda x: x \* x print(square(5)) # Output: 25 |
| nonlocal | Modifies a variable in the enclosing scope. | def outer(): x = 5 def inner(): nonlocal x x = 10 inner() print(x) |
| not | Logical NOT operator. | if not False: print("True") |
| or | Logical OR operator. | if x > 5 or y < 10: print("One condition is True") |
| pass | Placeholder for future code. | def future\_function(): pass |
| raise | Raises an exception. | raise ValueError("Invalid value") |
| return | Returns a value from a function. | def add(a, b): return a + b |
| try | Defines a block to handle exceptions. | try: x = int(input()) except ValueError: print("Invalid input") |
| while | Looping construct. | while x < 10: x += 1 |
| with | Used for resource management. | with open("file.txt") as f: content = f.read() |
| yield | Generates values in a generator function. | def counter(): yield 1 yield 2 gen = counter() print(next(gen)) # Output: 1 |