Introduction to Data Science Lab
Fall 2025
Lab 03 – Week 03

Python Programming Cont.

Loops

Loops are a way to execute a block of code multiple times. There are two main types of loops: while loops and for loops.

```
i = 0
while i <= len('Program'):
  print(i)
  i += 1 # equivalent to i = i + 1
```

```
0
1
2
3
4
5
6
7
```

```
for p in range(2,10):
  print(p)
```

```
2
3
4
5
6
7
8
9
```

Break and Continue Statement

break statement

- The break statement ends the loop immediately (no further iterations).
- Useful when you've found what you need and don't want to continue looping.

```
for i in range(1, 10):
    if i == 5:
        break   # stop loop completely when i = 5
    print(i)
```

```
1
2
3
4
```

continue statement

- The continue statement skips the current iteration and jumps to the next loop cycle.
- The loop itself does not stop.

```
] for i in range(1, 6):
    if i == 3:
        continue   # skip when i = 3
    print(i)
```

```
1
2
4
5
```

List

List are ordered collection of data. They are mutable (changeable) and allow duplicate elements. They are created using square brackets []. Data can be different types. Lists are versatile and can be used to store and manipulate sequences of items.

Syntax

list_A = ['a','b','c','d','BS_CS', 0343]

print(list_A)

print(list_A[1:4]) or print(list_A[:4]) #Slicing

Different operations can be performed on list

Operation	Description	Syntax
Append	Add an element at the end	list_A.append(1,2,3)

Insert	Inserts an element at a specific index	list_A.insert(1, 'Python')
Extend	Adds more than one element at the end of the list	list_A.extend(['Python', 'Java' , 'Perl'])
Remove	Removes the first item with the specified value	list_A.remove('Perl')
Pop	Removes last element or removes the element at the specified position.	list_A.pop(); list_A.pop(2); #Remove?
Index	Returns the index of the first element with the specified value	list_A.index(0343);
Sorting	Sorts the list. And reverse() reverses the order of the list	list_A.sort(); list_A.reverse();

Built-in Functions

Function	Description
cmp (list1, list2)	Compares elements of both lists.
len (list1)	Gives the total length of the list.
max (list1)	Returns item from the list with max value.
min (list1)	Returns item from the list with min value.
list (seq)	Converts a tuple into list.

Set

Set is unordered collection of data. They are mutable and don't allow duplicate elements (Unique elements). They are created using curly braces {} or the set() constructor. Sets are useful for mathematical operations like union, intersection, and difference.

basket = {'Apple', 'Orange', 'pearl', 'Banana', 1, 2, 57}

Example:

Set1 = {values} Set2 = {values}

Different operations can be performed on set

Operation	Description	Syntax
Union	All the elements of Set1 or Set2	new_set = Set1 | Set2 Set1.union(Set2)
Intersection	Include the common elements between the two sets.	new_set = Set1 **&** Set2 Set1.intersection(Set2)
Difference	Elements of set1 that are not present on set2.	new_set = Set1 - Set2 Set1.difference(Set2) #Output {1,3,4,5,6} Set2.difference(Set1) #Output?
Symmetric difference	all elements of set1 and set2 without the common elements	new_set = Set1 ^ Set2 Set1.symmetric_difference(Set2)
Subset	Returns **True** if another set contains this set	new_set = Set1 <= Set2 Set1.issubset(Set2)

Superset	Returns the index of the first element with the specified value	new_set = Set1 >= Set2 Set1.issuperset(Set2)
Add	Sorts the list. And reverse() reverses the order of the list	new_set.add(4)
Remove	Removes the element from a set. It returns a Key Error if element is not part of the set.	new_set.remove(2)
Discard	Removes the element from the set, and doesn't raise the error	new_set.discard(3)
Clear	Removes all elements from the set	new_set.clear()

Tuples

Tuples are ordered, immutable (unchangeable), and allow duplicate elements. They are created using parentheses (). Tuples are often used for fixed collections of items where immutability is desired. Tuples are sequences, just like lists. The differences between tuples and lists are, the tuples cannot be changed unlike lists and tuples use parentheses, whereas lists use square brackets.

data = (‘Apple’, ‘Orange’, ’pearl’, ‘Banana’, 1, 2, 57)

new_tuple = data [1:4] #create a new tuple by extracting a portion of an existing tuple using slicing.

Dictionary

A dictionary is a collection which is unordered, changeable, and indexed. In Python dictionaries

are written with curly brackets, and they have keys and values.

Syntax:

data = { key : value }

Example:

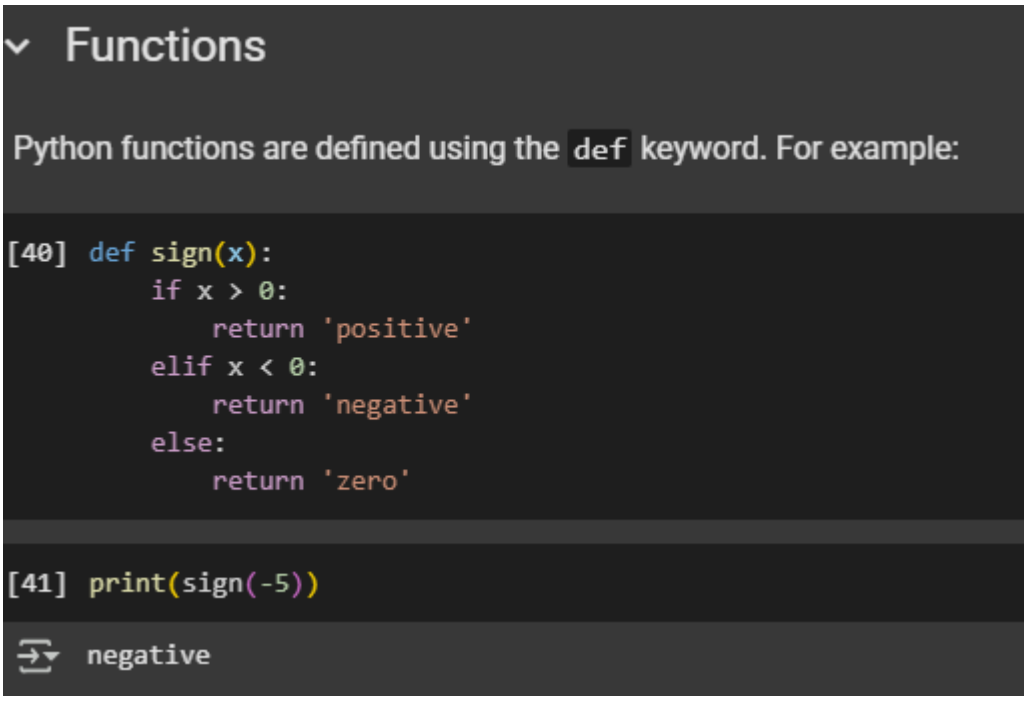
thisdict = { "brand": "Ford", "model": "Mustang", "year": 1964}

print(thisdict)

Accessing Items

x = thisdict["model"] or x = thisdict.get("model")

Functions



Exception handling

1. What is an Exception?

- An exception is an error that happens during program execution.
- Instead of crashing, Python lets you catch and handle the error gracefully.

```
] print(10 / 0)   # ZeroDivisionError
```

```
---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call last)
/tmp/ipython-input-1967088891.py in <cell line: 0>()
----> 1 print(10 / 0)   # ZeroDivisionError

ZeroDivisionError: division by zero
```

Lab Tasks

Task # 01

Online Store Shopping Cart

Write a Python program to simulate the online store's shopping cart system. The program should use a while loop to provide an interactive shopping experience.

Tasks:

- Initialize an empty list called shopping_cart to store items.
- Display a welcome message and available products.
- Loop should be iterating over AI Response Simulator and Investment decision and keep asking for user input.
- Prompt the user to enter the number corresponding to the product they want to add to the cart.
- Check if the user entered 'q'. If so, break the loop.
- Convert the choice to an integer and validate it within the valid range (1 to 3).
- Based on the user's choice, add the selected product to the shopping_cart list.
- Display a confirmation message indicating the added product.
- Continue the loop to allow the user to add more products or proceed to checkout.
- After the loop ends, display the contents of the shopping cart.
- Display a goodbye message.
- End the program.

Task # 02

English-Spanish Dictionary

You are creating a basic English-Spanish dictionary that allows users to look up English words and find their Spanish translations.

Tasks

- Create a dictionary where English words are keys and their Spanish translations are values.
- Ask the user to input an English word.
- Check if the word exists in the dictionary.
- If the word exists, print its Spanish translation.
- If the word does not exist, inform the user that the translation is not available.
- Implement a loop that allows the user to keep looking up words until they choose to exit.

Task # 03

Nested dictionary

Create a dataset using a highly nested dictionary (at least 5 levels deep). The dataset can represent any real-world scenario (e.g., a university system, company departments, or an online store).

Requirements

- Define a dictionary with at least 5 levels of nesting.
- Store meaningful data (not just numbers or empty values).

Write a Python program to

- Access data from the innermost level.
- Update values at different nesting levels.
- Add a new key-value pair inside the deepest level.
- Safely handle cases where a key does not exist.

Example Scenario (University Dataset):

university = {

```
"Faculty of Science": {
    "Department of CS": {
        "BS Program": {
            "Semester 1": {
                "Courses": {
                    "CS101": {"Title": "Intro to Programming", "Credits": 3}
                }
            }
        }
    }
}
```

Students should perform operations such as:

- Print the course title of DS3001.
- Update the credits of DS3001.
- Add a new course inside Semester 1.
- Try to access a non-existing course and handle it safely.

Task # 04

Email Classification

Develope a tool which will classify an email.

Tasks:

- Create a function classify_emails that takes a list of emails as an argument.
- Inside the function, categorize emails into "Important," "Promotions," and "Spam" based on keywords and content analysis.
- Loop through the emails and identify keywords to determine their category.
- If an email contains keywords indicating importance, assign it as "Important."
- If an email contains keywords related to promotions or deals, assign it as "Promotions."
- If an email is suspected to be spam based on certain keywords or patterns, assign it as "Spam."
- Return three lists of emails for each category.
- Print the lists.

Task # 05

Online Bookstore Inventory

You are developing an inventory management system for an online bookstore. The system should allow the bookstore to keep track of books, their quantities, prices, and generate reports.

Tasks:

- Create a list to represent the bookstore's inventory. Each item in the list will be a dictionary containing book details: title, author, quantity, and price.
- Implement a loop that allows the bookstore to add books to the inventory. Ask for the book's title, author, quantity, and price. Append a dictionary with the book details to the inventory list.
- Ask users to input a book title they want to search for. Search the inventory list for the book title and display its details if found.
- Allow the bookstore to update the quantity of a book in the inventory. Ask for the book's title and the new quantity. Update the quantity in the corresponding dictionary.
- Implement a function to calculate the total revenue based on the quantities sold and prices of each book. Calculate the overall revenue for the entire inventory.
- Print a comprehensive report showing all books in the inventory along with their details: title, author, quantity, and price.
- Check the inventory for books with a quantity of zero. Print an alert message for any out-of-stock books.

Task # 06

You are designing a Student Grades Management System. Each student has a name and a list of their scores in different subjects. You need to manage and analyze this data using lists.

Tasks

- Create a list of students, where each student is represented as:
    1. A list containing the student's name and their scores.
    2. Example: ["Alice", [85, 90, 78]]
- Allow the user to input data for at least 3 students.
- Print all students with their scores in a readable format.
- Calculate and display:
    1. The average score of each student.
    2. The highest score among all students.
    3. The student with the lowest average score.
- Update a score:

1. Ask the user to select a student and update one of their subject scores.

- Sort the students:
    1. By their average score in ascending order.
    2. By name in alphabetical order.
- Implement a search function:
    1. Ask the user to enter a student's name and display their scores and average.
    2. If the student does not exist, show an appropriate message.

Task # 07

Write a Python program that:

1. Asks the user for two numbers.
2. Divides the first number by the second.
3. Handles the following exceptions using try-except:
    - ValueError → if the input is not a number.
    - ZeroDivisionError → if the second number is zero.
    - TypeError → if the entered values are not of a compatible type for division.
    - Any other Exception → catch all unexpected errors and print the actual error message sent by Python.
4. If no exception occurs, print "Division successful: result = ..."
5. Always print "Program finished" in a finally block.

Task # 08

Raising a Custom Exception

Write a Python program that deliberately raises an exception based on a certain condition. The raised exception should include a custom error message. This custom exception must be caught in the same way as any built-in exception is caught.

Requirements

- Ask the user to enter a number.
- If the number is negative, raise a ValueError with a custom message like 'Negative numbers are not allowed!'.
- Use try-except to catch this error.
- Print the custom error message when the exception is caught.
- If no exception occurs, print 'Valid input received.'.
- Always print 'Program finished' in a finally block.