PYTHON

Lecture – 12

Recap

- List
- Tuple
- Set
- Paper Work
    - Write down the differences among List, Tuple, Set, Dictionary and Arrays based on their main features.
    - Write a python code that take few numbers in a list from user input. Remove duplicates from it and find sum of all values.

Contents

- List
- Tuple
- Set
- Dictionary
- JSON
- Array

Dictionay

- Dictionaries are used to store data values in *key:value* pairs.
- A dictionary is a collection which is ordered*, changeable and do not allow duplicates.
- Dictionaries are written with *curly brackets,* and have keys and values.
- The values in dictionary items can be of *any data type*.

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

```
thisdict = {
  "brand": "Ford",
  "electric": False,
  "year": 1964,
  "colors": ["red", "white", "blue"]
}
```

Accessing Dictionay

- You can access the items of a dictionary by referring to its **key** name, inside square brackets.

```
thisdict =  { "brand": "Ford",
        "model": "Mustang",
        "year": 1964
}
x = thisdict["model"]
print(x) #Output: Mustang
y = thisdict.get("brand")
print(y) #Output: Ford
print(thisdict) #Output: {'brand': 'Ford', 'model':
'Mustang', 'year': 1964}
```

Accessing Through Loop

```
thisdict =  { "brand": "Ford", "model": "Mustang", "year": 1964 }
#accessing keys
for x in thisdict:
  print(x)
#accessing values
for y in thisdict:
    print(thisdict[y])
#accessing values using values() method
for val in thisdict.values():
  print(val)
#accessing keys using keys() method
for key in thisdict.keys():
  print(key)
#accessing keys and values both
for k in thisdict:
    print(f"Key: {k} and Value: {thisdict[k]}")
#accesing keys and values both using items() method
for a,b in thisdict.items():
    print(f"Key: {a} and Value: {b}")
```

Add Items

- Adding an item to the dictionary is done by using a *new index key and assigning a value* to it.

```
student = {
          ”name": ”Alice",
         ”age": 20,
         ”grade": ”A”
         }
student[”major"] = ”CSE"
print(student)
```

- The *update()* method will update the dictionary with the items from a given argument. If the item does not exist, the item will be added.

```
student.update({”roll": 123})
```

Change Items

- You can change the value of a specific item by referring to its *key name*.

```
student = {
          ”name": ”Alice",
         ”age": 20,
         ”grade": ”A”
         }
student[”grade"] = ”A+"
print(student)
```

- The *update()* method will update the dictionary with the items from the given argument.

```
student.update({”age": 22})
```

Remove Items

- The *pop()* method removes the item with the specified key name:

```
student = { ”name": ”Alice", ”age": 20, ”grade": ”A”,”roll”:123 }
student.pop(”age”)
print(student)
```

- The *popitem()* method removes the last inserted item

```
student.popoitem()
```

- The *del* keyword removes the item with the specified key name

```
del student[“grade”]
```

- **del** keyword can delete a dictionary completely
- clear() function can be used to remove all items from a dictionary.

Nested Dictionaries

- Dictionaries can contain other dictionaries, making them useful for representing more complex data structures.

```
students = {
    "student1": {"name": "Alice", "age": 20, "grade": "A"},
    "student2": {"name": "Bob", "age": 22, "grade": "B"},
    "student3": {"name": "Charlie", "age": 23, "grade": "A"}
      }
print(students["student1"]["name"])   # Output: Alice
print(students["student2"]["age"])    # Output: 22
```

Nested Dictionaries through Loop

- Dictionaries can contain other dictionaries, making them useful for representing more complex data structures.

```
students = {
    "student1": {"name": "Alice", "age": 20, "grade": "A"},
    "student2": {"name": "Bob", "age": 22, "grade": "B"},
    "student3": {"name": "Charlie", "age": 23, "grade": "A"}
    }
for student_key, student_info in students.items():
    print(student_key)
    for key, value in student_info.items():
       print(f"{key}: {value}")
```

Exercises on Dictionay

12.1 Find the grade of a students from a dictionary. Student name will be taken from user input.

12.2 Create a dictionary that represents a person(name, age, gender, weight, phone etc) and access each key and value pairs.

12.3 Update the age of the person (from 12.2)

12.4 Check whether a key gender exist or not. If exists, then delete this pair.

12.5 Take a sentence as input and create a dictionary that counts the occurrences of each word.

Exercises on Dictionay- Solutions

12.1 Find the grade of a students from a dictionary. Student name will be taken from user input.

```
students = {
    "Alice": {"age": 20, "grade": "A"},
    "Bob": {"age": 22, "grade": "B"},
    "Charlie": {"age": 21, "grade": "A"}
}
student_name = input("Enter the student's name: ")
if student_name in students:
    print(f"{student_name}'s grade is:{students[student_name]['grade']}")
else:
    print(f"Student {student_name} not found.")
```

Python JSON

JSON: JavaScript Object Notation

Python Array

- There is no build in array in python.
- It suggest to use List as array. Practically List has all features that can be performed by array.

Functions

- A **function** is a *block of reusable code* that performs a specific task.
- It allows to structure code, making it modular, easier to maintain, and reusable.
- In Python, a function is defined using the **def** keyword, followed by the function name and parentheses (). [`def addition():`]
- You can pass data, known as parameters, into a function (*receive/ catch data*).
- A function can return data as a result (*return or back data*).
- A function only runs when it is called.
-

Key Elements of Function

- **def**: This keyword is used to define a function.
- **function_name**: This is the name of the function. It should be meaningful and indicate the purpose of the function.
- **parameters**: These are the input values passed to the function.
- **return**: This keyword is used to return a value from the function.
- **Docstring**: A short description of what the function does (optional, but recommended for clarity).

Example

- A function without parameter:

```
def hello():
    print("Hello python")
hello()
```

- A function with parameter:

```
def hello(name):
        print(f"Hello {name}")
hello("Doha")
```

Example

- Write a program that take two numbers as input and print their sum as output.

Example

- Write a program that take two numbers as input and print their sum as output.

```
num1=int(input("Enter first number: "))
num2=int(input("Enter 2nd number: "))
sum=num1+num2
print(f"Sum of {num1} and {num2} is: {sum}")
```

Now, write this code using function.

Example

- Addition using function:

```
def addition(n1, n2): #function declaration
    sum=n1+n2
    return sum

num1=int(input("Enter first number: "))
num2=int(input("Enter 2nd number: "))
summation=addition(num1,num2) #function calling
print(f"Sum of {num1} and {num2} is: {summation}")
```

Exercises on Function

12.6 write a function that find the maximum number from a given number list.

12.7 Write a function is_even(number) that returns True if the given number is even and False if it's odd.

12.8 Write a function factorial(n) that calculates and returns the factorial of a given number n

12.9 Write a function sum_list(numbers) that takes a list of numbers and returns the sum of all elements in the list.

12.10 Write a function is_prime(n) that checks if the given number is prime. Return True if it is prime, and False otherwise.

Exercises on Function

12.6 write a function that find the maximum number from a given number list.

```
def find_max(numbers):
    max_num = -1000
    for num in numbers:
        if num > max_num:
            max_num = num  # Update max_num number
    return max_num

# Numbers List
number_list = [34, 12, 56, 78, 23, 45]
max_number = find_max(number_list)
print(f"The maximum number is: {max_number}")
```

Thank You