image

Agenda

- Strings
- Tuples
- Sets
- Dictionary

```
# Check Consecutive

li = [1, 2, 3, 4, 1, 2, 2, 4]

def check_consecutive(li):
    n = len(li)

    for i in range(n-1):
        # check for consecutive
        if li[i] == li[i+1]:
            return True
    return False

check_consecutive(li)
```

```
True
```

Strings

```
# Defining a string
# Giving command to alexa

# Use of strings in NLP

s = "Alexa! What is the time"

type(s)
```

```
str
```

```
# New line character

email = """
Greetings of the day
```

```
Body

Best Regards
Scaler
"""

print(email)
```

```
Greetings of the day

Body

Best Regards
Scaler
```

```
email
```

```
'\nGreetings of the day\n\nBody\n\nBest Regards\nScaler\n'
```

```
print("Hey my name is \n Rahul")
```

```
Hey my name is
 Rahul
```

```
# Printing quotes in the string

print("What's your name")
```

```
What's your name
```

```
# Printing raw strings

print(r"Hey my name is \n Rahul")
```

```
Hey my name is \n Rahul
```

```
# Reversing a string

name = "Rahul Janghu"

# Print reversed string using a loop

name[::-1]
```

```
'uhgnaJ luhaR'
```

```
# Immutable?
# name[0] = "r"
# Strings are immutable


# String concatenation
greet = "Hello"
greet += " world"
greet
```

```
'Hello world'
```

```
greet * 2
```

```
'Hello worldHello world'
```

```
# greet * greet


# is vs ==
# == gives true or false by comparing values
a = 10
b = 10
print(a == b)
print(a is b)
```

```
True
True
```

```
a = 1000
b = 1000

print(a == b)
print(a is b)
```

```
True
False
```

```
# is: gives true or false by comparing memory address

a = [2, 3, 4]
b = [4, 5, 6]

a is b
```

```
False
```

```
# Quiz

bits = [False, True, False, False, True, False, False, True]
bits = [1 if b==True else 0 for b in bits]
print(bits)
```

```
[0, 1, 0, 0, 1, 0, 0, 1]
```

String Methods

- capitalize
- title
- upper
- lower
- isupper
- islower
- count
- index
- find
- join
- split

```
s
```

```
'Alexa! What is the time'
```

```
# capitalize

"rahul janghu".capitalize()
```

```
'Rahul janghu'
```

```
"rahul JANGHU".capitalize()
```

```
'Rahul janghu'
```

```
# title

"rahul janghu".title()
```

```
'Rahul Janghu'
```

```
"rahul JANGHU".title()
```

```
'Rahul Janghu'
```

```
# upper, lower
```

```
s
```

```
'Alexa! What is the time'
```

```
s.upper()
```

```
'ALEXA! WHAT IS THE TIME'
```

```
s.lower()
```

```
'alexa! what is the time'
```

```
color = input()
```

```
if color.lower() == "red":
    print("Color is red")
```

```
 RED
```

```
Color is red
```

```
# +, *
```

```
# islower and isupper
```

```
"Rahul".islower()
```

```
False
```

```
"rahul".islower()
```

```
True
```

```
"RAHUL".isupper()
```

```
True
```

```
"Rahul".isupper()
```

```
False
```

```
# count, index and find
# count
s
'Alexa! What is the time'
s.count("a")
2
s.count("A")
1

# index
s
'Alexa! What is the time'
s.index("a")
4
s.index("a", 5)
9
# s.index("RA")
# s.index?

# find
s
'Alexa! What is the time'
s.find("a", 5)
9
s.find("Ra")
-1
```

```
# startswith

# Alexa

# web = "https://scaler.com"

s

'Alexa! What is the time'

s.startswith("Alexa")

True
```

String formatting

```
length = 3
breadth = 4
area = length * breadth

print("Area of rectangle is " + str(area))

Area of rectangle is 12

# "Rahul" + 2

print(f"Area of rectangle is {area}")

Area of rectangle is 12
```

Most important: join and split

- string -> list: split
- list -> string: join

```
s

'Alexa! What is the time'

li = s.split()


" ".join(li)

'Alexa! What is the time'
```

```
fruits = ["apples", "bananas", "strawberries"]
fruits = [i.upper() for i in fruits]

fruits
```

```
['APPLES', 'BANANAS', 'STRAWBERRIES']
```

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

Tuples

```
# Creating a tuple

# immutable

# empty tuple

t = ("Python")
type(t)
```

```
str
```

```
t = ("Rahul")
type(t)
```

```
str
```

```
t = ("Rahul",)
type(t)
```

```
tuple
```

```
t = tuple("Rahul")

t
```

```
('R', 'a', 'h', 'u', 'l')
```

```
t = 1, 2, 3

print(t, type(t))
```

```
(1, 2, 3) <class 'tuple'>
```

```
# Adding 2 tuples
```

```
(2, 3, 4) + (5, 6, 7)
```

```
(2, 3, 4, 5, 6, 7)
```

```
# Multiply tuple
(2, 3, 4) * 2
```

```
(2, 3, 4, 2, 3, 4)
```

```
# (2, 3, 4) * (2, 3, 4)
# t[0] = 23
```

```
# Unpacking and packing of a tuple
# Multiple assignments
# packing
t = 1, 2, 3
print(t)
```

```
(1, 2, 3)
```

```
# unpacking
a, b, c = t
print(a, b, c)
```

```
1 2 3
```

```
# return in function
def foo():
    return 1, 2, 3
print(foo())
```

```
(1, 2, 3)
```

```
a, b, c = foo()
print(a, b, c)
```

```
1 2 3
```

```
# unpacking these values to the variables a, b, and c
```

```
# Rahul Janghu
```

```
# Quizzes
```

```
tup1 = 1,2,3
print(tup1)
```

```
(1, 2, 3)
```

```
a = []
for i in range(3):
    b = []
    for j in range(2):
        b.append(i * j)
    a.append(b)

print(a)
```

```
[[0, 0], [0, 1], [0, 2]]
```

Dictionaries

image

```
# Creating a dict
```

```
d = {"name" : "Rahul",
     "Age" : 26,
     "Hobbies" : ["Travelling", "Reading books", "Meditation"]
    }

type(d)
```

```
dict
```

```
# Keys are unique in a dictionary
# We can have same values for different keys
```

```
# Does dictionaries support indexing
```

```
# Dictionary doesnt support indexing
# d[0] will not work
```

```
d
```

```
{'name': 'Rahul',
 'Age': 26,
 'Hobbies': ['Travelling', 'Reading books', 'Meditation']}
```

```
# d[0]
```

```
# d[key]
```

```
d["Age"]
```

```
26
```

```
d["Age"] = 45
```

```
d
```

```
{'name': 'Rahul',
 'Age': 45,
 'Hobbies': ['Travelling', 'Reading books', 'Meditation']}
```

```
# d["age"]
```

```
# d.keys()
```

```
d.keys()
```

```
dict_keys(['name', 'Age', 'Hobbies'])
```

```
# d.values
```

```
d.values()
```

```
dict_values(['Rahul', 45, ['Travelling', 'Reading books',
'Meditation']])
```

```
# d.get()
```

```
d.get("Age", "Not present")
```

```
45
```

```
d.get("age", "Not present")
```

```
'Not present'
```

```
# Iteration on a dictionary

for i in d:
    print(i)
```

```
name
Age
Hobbies
```

```
for i in d:
    print(i, d[i])
```

```
name Rahul
Age 45
Hobbies ['Travelling', 'Reading books', 'Meditation']
```

```
d.items()
```

```
dict_items([('name', 'Rahul'), ('Age', 45), ('Hobbies', ['Travelling',
'Reading books', 'Meditation'])])
```

```
for i, j in d.items():
    print(i, j)
```

```
name Rahul
Age 45
Hobbies ['Travelling', 'Reading books', 'Meditation']
```

```
# remove something from dictionary

# d.pop()

d.pop("name")
```

```
'Rahul'
```

```
d
```

```
{'Age': 45, 'Hobbies': ['Travelling', 'Reading books', 'Meditation']}
```

```
# update

d
```

```
{'Age': 45, 'Hobbies': ['Travelling', 'Reading books', 'Meditation']}
```

```
d["Hobbies"]
```

```
['Travelling', 'Reading books', 'Meditation']
```

```
type(d["Hobbies"])
```

```
list
```

```
d["Hobbies"].append("Biking")
```

```
d
```

```
{'Age': 45, 'Hobbies': ['Travelling', 'Reading books', 'Meditation',
'Biking']}
```

```
# Mutable?

# Dictionaries are mutable in nature

d1 = {"name" : "Rahul",
      "Profession" : "Masterji"
     }

d.update(d1)

d
```

```
{'Age': 45,
 'Hobbies': ['Travelling', 'Reading books', 'Meditation', 'Biking'],
 'name': 'Rahul',
 'Profession': 'Masterji'}
```

```
d["Age"] = 26

d
```

```
{'Age': 26,
 'Hobbies': ['Travelling', 'Reading books', 'Meditation', 'Biking'],
 'name': 'Rahul',
 'Profession': 'Masterji'}
```

```
# Given 2 lists:
# list1 = ["RJ", "TS", "OT"]
# list2 = ["Rahul Janghu", "Tony Stark", "Odin Thor"]
# answer = {"RJ" : "Rahul Janghu", "TS" : "Tony Stark", "OT" : "Odin
Thor"}

list1 = ["RJ", "TS", "OT"]
list2 = ["Rahul Janghu", "Tony Stark", "Odin Thor"]

answer = zip(list1, list2)

type(answer)

zip

print(answer)

<zip object at 0x7f97904e0d40>

print(dict(answer))

{'RJ': 'Rahul Janghu', 'TS': 'Tony Stark', 'OT': 'Odin Thor'}




# Doubts

# hit and trial
# Docstring
# Refer to google

s

'Alexa! What is the time'

s = "Rahul 123"

s.isalnum()

False

s = "Rahul123"

s.isalnum()

True

"s".isalnum()

True

"2".isalnum()
```

```
True
```

```
"".isalnum()
```

```
False
```

```
s.isalnum?
```

```
Signature: s.isalnum()
Docstring:
Return True if the string is an alpha-numeric string, False otherwise.

A string is alpha-numeric if all characters in the string are alpha-
numeric and
there is at least one character in the string.
Type:      builtin_function_or_method
```

```
"2@".isalnum()
```

```
False
```