string concatenation

In [3]:
```
first = 'jayesh'
first
```

Out[3]: 'jayesh'

In [4]:
```
last = 'sreedharan'
last
```

Out[4]: 'sreedharan'

In [5]:
```
full = 'jayesh' + ' ' + 'sreedharan'
full
```

Out[5]: 'jayesh sreedharan'

Variables in Python

In [7]:
```
first_name = 'jayesh'
last_name = 'sreedharan'
state = 'kerala'
city = 'alappuzha'
age = 77777
is_married = False
```

Printing the values stored in the variables

In [9]:
```
print('First name:', first_name)
print('First name length:', len(first_name))
print('Last name: ', last_name)
print('Last name length: ', len(last_name))
print('State: ', state)
print('City: ', city)
print('Age: ', age)
print('Married: ', is_married)
```

```
First name: jayesh
First name length: 6
Last name:  sreedharan
Last name length:  10
State:  kerala
City:  alappuzha
Age:  77777
Married:  False
```

Declaring multiple variables in one line

In [11]:
```
first_name, last_name, state, age, is_married = 'jayesh', 'sreedharan', 'kerala', 250, False

print(first_name, last_name, state, age, is_married)
print('First name:', first_name)
print('Last name: ', last_name)
print('State: ', state)
```

```
print('Age: ', age)
print('Married: ', is_married)
```

```
jayesh sreedharan kerala 250 False
First name: jayesh
Last name:  sreedharan
State:  kerala
Age:  250
Married:  False
```

In [12]:
```
letter ='dfghh'
len(letter)
print(len(letter))
```

```
5
```

Multiline String

In [14]:
```
multiline_string = '''I am a teacher and enjoy teaching.
I didn't find anything as rewarding as empowering people.
That is why I teach.'''
print(multiline_string)
```

```
I am a teacher and enjoy teaching.
I didn't find anything as rewarding as empowering people.
That is why I teach.
```

Another way of doing the same thing

In [16]:
```
multiline_string = """I am a teacher and enjoy teaching.
I didn't find anything as rewarding as empowering people.
That is why I teach."""
print(multiline_string)
```

```
I am a teacher and enjoy teaching.
I didn't find anything as rewarding as empowering people.
That is why I teach.
```

In [17]:
```
#### Unpacking characters
language = 'Python'
a,b,c,d,e,f = language # unpacking sequence characters into variables
```

```
print(a) # P
print(b) # y
print(c) # t
print(d) # h
print(e) # o
print(f) # n
```

```
P
y
t
h
o
n
```

In [18]:
```
#### Unpacking characters
a,b,c,d,e,f = 'Python' # unpacking sequence characters into variables
print(a) # P
print(b) # y
print(c) # t
print(d) # h
print(e) # o
print(f) # n
```

```
P
y
t
h
o
n
```

In [19]:
```
# Accessing characters in strings by index
language = 'Python'
first_letter = language[0]
print(first_letter) # P
second_letter = language[1]
print(second_letter) # y
last_index = len(language) - 1
last_letter = language[last_index]
print(last_letter) # n
```

```
P
y
n
```

In [20]:
```
# If we want to start from right end we can use negative indexing. -1 is the last index
language = 'Python'
last_letter = language[-1]
print(last_letter) # n
second_last = language[-2]
print(second_last) # o
```

```
n
o
```

In [21]:
```
# Slicing
language = 'Python'
first_three = language[0:3] # starts at zero index and up to 3 but not include 3
last_three = language[3:6]
print(last_three) # hon
# Another way
last_three = language[-3:]
print(last_three)   # hon
last_three = language[3:]
print(last_three)   # hon
```

```
hon
hon
hon
```

In [22]:
```
# Skipping character while splitting Python strings
language = 'Python'
pto = language[0:6:2] #
print(pto) # pto
```

```
Pto
```

In [23]:
```
# Escape sequence
print('I hope every one enjoying the python challenge.\nDo you ?') # line break
print('Days\tTopics\tExercises')
print('Day 1\t3\t5')
print('Day 2\t3\t5')
print('Day 3\t3\t5')
print('Day 4\t3\t5')
print('This is a back slash  symbol (\\)') # To write a back slash
print('In every programming language it starts with \"Hello, World!\"')
```

```
I hope every one enjoying the python challenge.
Do you ?
Days    Topics  Exercises
Day 1   3       5
Day 2   3       5
Day 3   3       5
Day 4   3       5
This is a back slash  symbol (\)
In every programming language it starts with "Hello, World!"
```

In [24]:
```python
# capitalize(): Converts the first character the string to Capital Letter

challenge = 'thirty days of python'
print(challenge.capitalize()) # 'Thirty days of python'
```

```
Thirty days of python
```

In [25]:
```python
# count(): returns occurrences of substring in string, count(substring, start=.., end=..)

challenge = 'thirty days of python'
print(challenge.count('y')) # 3
print(challenge.count('y', 7, 14)) # 1
print(challenge.count('th')) # 2`
```

```
3
1
2
```

In [26]:
```python
# endswith(): Checks if a string ends with a specified ending

challenge = 'thirty days of python'
print(challenge.endswith('on'))   # True
print(challenge.endswith('tion')) # False
```

```
True
False
```

In [27]:
```python
# expandtabs(): Replaces tab character with spaces, default tab size is 8. It takes tab size argument

challenge = 'thirty\tdays\tof\tpython'
print(challenge.expandtabs())   # 'thirty  days    of      python'
print(challenge.expandtabs(10)) # 'thirty    days      of        python'
```

```
thirty  days    of      python
thirty    days      of        python
```

In [28]:
```
# find(): Returns the index of first occurrence of substring

challenge = 'thirty days of python'
print(challenge.find('y'))  # 5
print(challenge.find('th')) # 0
```

```
5
0
```

In [29]:
```
# format()      formats string into nicer output
first_name = 'jayesh'
last_name = 's'
job = 'teacher'
country = 'india'
sentence = 'I am {} {}. I am a {}. I live in {}.'.format(first_name, last_name, job, country)
print(sentence) # I am jayesh s. I am a teacher. I live in india.
```

```
I am jayesh s. I am a teacher. I live in india.
```

In [30]:
```
radius = 10
pi = 3.14
area = pi * radius ** 2
result = 'The area of circle with {} is {}'.format(str(radius), str(area))
print(result) # The area of circle with 10 is 314.0
```

```
The area of circle with 10 is 314.0
```

In [31]:
```
# index(): Returns the index of substring
challenge = 'thirty days of python'
print(challenge.find('y'))  # 5
print(challenge.find('th')) # 0
```

```
5
0
```

In [32]:
```
# isalnum(): Checks alphanumeric character

challenge = 'ThirtyDaysPython'
print(challenge.isalnum()) # True

challenge = '30DaysPython'
```

```
print(challenge.isalnum()) # True

challenge = 'thirty days of python'
print(challenge.isalnum()) # False

challenge = 'thirty days of python 2019'
print(challenge.isalnum()) # False
```

```
True
True
False
False
```

In [33]:
```
# isalpha(): Checks if all characters are alphabets

challenge = 'thirty days of python'
print(challenge.isalpha()) # True
num = '123'
print(num.isalpha())      # False
```

```
False
False
```

In [34]:
```
# isdecimal(): Checks Decimal Characters

challenge1 = 'thirty days of python'
print(challenge1.isdecimal()) # False
challenge2 = '67'
print(challenge2.isdecimal()) # True
```

```
False
True
```

In [35]:
```
# isdigit(): Checks Digit Characters

challenge = 'Thirty'
print(challenge.isdigit()) # False
challenge = '30'
print(challenge.isdigit())   # True
```

```
False
True
```

In [36]:
```
# isdecimal():Checks decimal characters

num = '10'
print(num.isdecimal()) # True
num = '10.5'
print(num.isdecimal()) # False
```

```
True
False
```

In [37]:
```
# isidentifier():Checks for valid identifier means it check if a string is a valid variable name

challenge = '30DaysOfPython'
print(challenge.isidentifier()) # False, because it starts with a number
challenge = 'thirty_days_of_python'
print(challenge.isidentifier()) # True
```

```
False
True
```

In [38]:
```
# islower():Checks if all alphabets in a string are lowercase

challenge = 'thirty days of python'
print(challenge.islower()) # True
challenge = 'Thirty days of python'
print(challenge.islower()) # False
```

```
True
False
```

In [39]:
```
# isupper(): returns if all characters are uppercase characters

challenge = 'thirty days of python'
print(challenge.isupper()) #  False
challenge = 'THIRTY DAYS OF PYTHON'
print(challenge.isupper()) # True
```

```
False
True
```

In [40]:
```
# isnumeric():Checks numeric characters

num = '10'
```

```
print(num.isnumeric())      # True
print('ten'.isnumeric())    # False
```

```
True
False
```

In [41]:
```
# join(): Returns a concatenated string

web_tech = ['HTML', 'CSS', 'JavaScript', 'React']
result = '# '.join(web_tech)
print(result) # 'HTML# CSS# JavaScript# React'
```

```
HTML# CSS# JavaScript# React
```

In [42]:
```
# strip(): Removes both leading and trailing characters

challenge = ' thirty days of python '
print(challenge.strip())
```

```
thirty days of python
```

In [43]:
```
# replace(): Replaces substring inside

challenge = 'thirty days of python'
print(challenge.replace('python', 'coding')) # 'thirty days of coding'
```

```
thirty days of coding
```

In [44]:
```
# split():Splits String from Left

challenge = 'thirty days of python'
print(challenge.split()) # ['thirty', 'days', 'of', 'python']
```

```
['thirty', 'days', 'of', 'python']
```

In [45]:
```
# title(): Returns a Title Cased String

challenge = 'thirty days of python'
print(challenge.title()) # Thirty Days Of Python
```

```
Thirty Days Of Python
```

In [46]:
```
# swapcase(): swap upper to lowe and vicevers in a String
```

```
challenge = 'thirty days of python'
print(challenge.swapcase())   # THIRTY DAYS OF PYTHON
challenge = 'Thirty Days Of Python'
print(challenge.swapcase())  # tHIRTY dAYS oF pYTHON
```

```
THIRTY DAYS OF PYTHON
tHIRTY dAYS oF pYTHON
```

In [47]:
```
# startswith(): Checks if String Starts with the Specified String

challenge = 'thirty days of python'
print(challenge.startswith('thirty')) # True
challenge = '30 days of python'
print(challenge.startswith('thirty')) # False
```

```
True
False
```

In [48]:
```
# Arithmetic Operations in Python
# Integers

print('Addition: ', 1 + 2)
print('Subtraction: ', 2 - 1)
print('Multiplication: ', 2 * 3)
print ('Division: ', 4 / 2)                         # Division in python gives floating number
print('Division: ', 6 / 2)
print('Division: ', 7 / 2)
print('Division without the remainder: ', 7 // 2)   # gives without the floating number or without the remaining
print('Modulus: ', 3 % 2)                           # Gives the remainder
print ('Division without the remainder: ', 7 // 3)
print('Exponential: ', 3 ** 2)                     # it means 3 * 3
```

```
Addition:  3
Subtraction:  1
Multiplication:  6
Division:  2.0
Division:  3.0
Division:  3.5
Division without the remainder:  3
Modulus:  1
Division without the remainder:  2
Exponential:  9
```

In [49]:
```
import math
print((7//2))
print((7/2))
print(math.ceil(7/2))
print(math.floor(7/2))

# Arithmetic Operations in Python
# Integers

print('Addition: ', 1 + 2)
print('Subtraction: ', 2 - 1)
print('Multiplication: ', 2 * 3)
print ('Division: ', 4 / 2)                         # Division in python gives floating number
print('Division: ', 6 / 2)
print('Division: ', 7 / 2)
print('Division without the remainder: ', 7 // 2)   # gives without the floating number or without the remaining
print('Modulus: ', 3 % 2)                           # Gives the remainder
print ('Division without the remainder: ', 7 // 3)
print('Exponential: ', 3 ** 2)                     # it means 3 * 3
```

```
3
3.5
4
3
Addition:  3
Subtraction:  1
Multiplication:  6
Division:  2.0
Division:  3.0
Division:  3.5
Division without the remainder:  3
Modulus:  1
Division without the remainder:  2
Exponential:  9
```

In [50]:
```
print(3 > 2)     # True, because 3 is greater than 2
print(3 >= 2)    # True, because 3 is greater than 2
print(3 < 2)     # False,  because 3 is greater than 2
print(2 < 3)     # True, because 2 is less than 3
print(2 <= 3)    # True, because 2 is less than 3
print(3 == 2)    # False, because 3 is not equal to 2
print(3 != 2)    # True, because 3 is not equal to 2
```

```
print(len('mango') == len('avocado'))  # False
print(len('mango') != len('avocado'))  # True
print(len('mango') < len('avocado'))   # True
print(len('milk') != len('meat'))      # False
print(len('milk') == len('meat'))      # True
print(len('tomato') == len('potato'))  # True
print(len('python') > len('dragon'))   # False
```

```
True
True
False
True
True
False
True
False
True
True
False
True
True
False
```

In [51]:
```
# Boolean comparison
print('True == True: ', True == True)
print('True == False: ', True == False)
print('False == False:', False == False)
print('True and True: ', True and True)
print('True or False:', True or False)
```

```
True == True:  True
True == False:  False
False == False: True
True and True:  True
True or False: True
```

In [52]:
```
import sys
sys.version
```

Out[52]:
```
'3.12.7 | packaged by Anaconda, Inc. | (main, Oct  4 2024, 13:17:27) [MSC v.1929 64 bit (AMD64)]'
```

In [53]:
```
import keyword
print(keyword.kwlist)
print(len(keyword.kwlist))
```

```
['False', 'None', 'True', 'and', 'as', 'assert', 'async', 'await', 'break', 'class', 'continue', 'def', 'del', 'eli
f', 'else', 'except', 'finally', 'for', 'from', 'global', 'if', 'import', 'in', 'is', 'lambda', 'nonlocal', 'not', 'o
r', 'pass', 'raise', 'return', 'try', 'while', 'with', 'yield']
35
```

In [54]:
```
import keyword as k
k.kwlist
```

Out[54]:
```
['False',
 'None',
 'True',
 'and',
 'as',
 'assert',
 'async',
 'await',
 'break',
 'class',
 'continue',
 'def',
 'del',
 'elif',
 'else',
 'except',
 'finally',
 'for',
 'from',
 'global',
 'if',
 'import',
 'in',
 'is',
 'lambda',
 'nonlocal',
 'not',
 'or',
 'pass',
 'raise',
 'return',
 'try',
 'while',
 'with',
 'yield']
```

In [55]:
```
![Local Image](C:\Users\jayes\121630167_3547341515327541_7161556486599304120_n.jpg)
```

```
'[Local' is not recognized as an internal or external command,
operable program or batch file.
```

In [56]:
```
a=10
a
```

Out[56]: 10

In [57]:
```
del a
a
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[57], line 2
      1 del a
----> 2 a

NameError: name 'a' is not defined
```

In [64]:
```
a=complex(True)
print(a)
print(a.real)
print(a.imag)
```

```
(1+0j)
1.0
0.0
```

In [62]:
```
sys.getsizeof(int())
```

Out[62]: 28

In [68]:
```
isinstance(a,complex)
```

Out[68]: True

In [70]:
```
isinstance(a,int)
```

Out[70]: False