COMPLEX DATA TYPE

complex numbers, which consist of a real and an imaginary part

In [ ]:
```
# Creating a Complex Number
#A complex number is created by appending the letter j (or J) to the imaginary p
# format is:
```

In [1]:
```
a = 4 + 5j
```

In [2]:
```
type(a)
```

Out[2]:
```
complex
```

Accessing the Real and Imaginary Parts

In [ ]:
```
# using the .real and .imag attributes
```

In [3]:
```
print(a.real)
```

```
4.0
```

In [4]:
```
print(a.imag)
```

```
5.0
```

Operations with Complex Numbers

In [6]:
```
a
```

Out[6]:
```
(4+5j)
```

In [7]:
```
b = 6 + 7j
b
```

Out[7]:
```
(6+7j)
```

In [8]:
```
a + b
```

Out[8]:
```
(10+12j)
```

In [9]:
```
a - b
```

Out[9]:
```
(-2-2j)
```

In [10]:
```
a * b
```

```
Out[10]: (-11+58j)
```

```
In [15]: a /b
```

```
Out[15]: (0.6941176470588236+0.023529411764705887j)
```

Using Built-in Functions

```
In [19]: # Python provides several functions to work with complex numbers, such as:
          # abs(z): Returns the magnitude (absolute value) of the complex number.
          # conj(z): Returns the complex conjugate of the number
```

```
In [20]: a
```

```
Out[20]: (4+5j)
```

```
In [22]: print(abs(a))   # magnitude
```

```
6.4031242374328485
```

```
In [ ]:
```

print is use for answer

```
In [26]: a=10
         b=20
         a               # without print function
         b
```

```
Out[26]: 20
```

```
In [27]: a=10
         b=20
         print(a)
         print(b)
```

```
10
20
```

```
In [28]: print(10)
         print(10,20)
         print('python')
         print(10,20,'python')
```

```
10
10 20
python
10 20 python
```

```
In [29]: num1=200
         num2=300
         add=num1+num2
         print(add)
```

```
500
```

print result with string

In [33]:
```
num1=20
num2=30
add=num1+num2
print("The addition of",num1, "and" ,num2, "is",add)
```
```
The addition of 20 and 30 is 50
```

In [34]:
```
name='Python'
age=20
city='hyd'
#hello my name is python and i am 10 year old from hydrabad
```

In [37]:
```
print("hello my name is",name,"and i am",age,"old from",city)
```
```
hello my name is Python and i am 20 old from hyd
```

print Format method

In [ ]:
```
# appply .format(val1,val2,....val-n method
```

In [38]:
```
num1=20
num2=30
add=num1+num2
print("the addition of {} and {} is = {}".format(num1, num2, add) )
```
```
the addition of 20 and 30 is = 50
```

In [44]:
```
name='Python'
age=20
city='hyd'
#hello my name is python and i am 10 year old from hyd
print('Hello my name is {}, and i am {} year old from {}'.format(name,age,city))
```
```
Hello my name is Python, and i am 20 year old from hyd
```

In [45]:
```
num1=100
num2=25
num3=333
avg=(num1+num2+num3)/3 # or we can use avg=round(num1+num2+num3)/3,2)
avg1=round((num1+num2+num3)/3,2)
# The avrage of num1,num2,num3 is = avg
print('The average of {}, {},{} is ={} or {}'.format(num1,num2,num3,avg,avg1))
```
```
The average of 100, 25,333 is =152.66666666666666 or 152.67
```

In [46]:
```
round(avg,2) # round of till 2 digite after decimal
```
Out[46]:
```
152.67
```

In [50]:
```
#More short format meythod(f string method)
 #variable should be in curly braces
 #and write everything inside quots ''
 #at starting simpaly add f
```

In [53]:
```
num1=20
num2=30
add = num1 + num2
print(f"THE addition of {num1} and {num2} is = {add}")
```

```
THE addition of 20 and 30 is = 50
```

In [ ]:
```
name='Python'
age=20
city='hyd'
#hello my name is python and i am 10 year old from hydrabad
```

In [56]:
```
print(f"Hello my name is {name} and i am {age} year old from {city}")
```

```
Hello my name is Python and i am 20 year old from hyd
```

End statement

In [57]:
```
# Here we will use end statement that joint line from end of one string to start
```

In [59]:
```
print('hello') # 1st statement
print('good morning') # 2nd statement
# i want print like:- hello good morning
```

```
hello
good morning
```

In [68]:
```
print("hello",end='')
print(" very Good Morning")
```

```
hello very Good Morning
```

In [71]:
```
print("hello sir/mam",end='')
print(" How can we help you?")
```

```
hello sir/mam How can we help you?
```

Seperator

In [ ]:
```
# here one print statement only we use
# inside one print statement we have multipal values
# we want to seperate these multipal values with anything
```

In [74]:
```
print('hello' ,' hiee',' how are you',sep='--->')
```

```
hello---> hiee---> how are you
```

In [76]:
```
print("hiee", " ssup", " How are you",sep='$')
```

```
hiee$ ssup$ How are you
```

In [78]:
```
print('hello','hai','how are you',sep='    ')
```

```
hello    hai    how are you
```

In [79]:
```
print('hello','hai','how are you',sep='@')
```

```
hello@hai@how are you
```

In [82]:
```
print(3,'.') # . is far from 3 so here we will use sep method
```
```
3 .
```

In [85]:
```
print(3,'.',sep='') # see now space setteld(also use to remove space B/W words)
```
```
3.
```

In [86]:
```
print(1,2,end=' ')
print(3,'.',sep='')
# will print 1 2 3.
```
```
1 2 3.
```

In [92]:
```
# Single line comment
letter = 'P'                # A string could be a single character or a bunch of
print(letter)               # P
print(len(letter))          # 1
greeting = 'Hello, World!'  # String could be  a single or double quote,"Hello,
print(greeting)             # Hello, World!
print(len(greeting))        # 13
sentence = "I hope you are enjoying 30 days of python challenge"
print(sentence)
```
```
P
1
Hello, World!
13
I hope you are enjoying 30 days of python challenge
```

In [94]:
```
# Multiline String
multiline_string = '''I am a teacher and enjoy teaching.
I didn't find anything as rewarding as empowering people.
That is why I created 30 days of python.'''
print(multiline_string)
# Another way of doing the same thing
multiline_string = """I am a teacher and enjoy teaching.
I didn't find anything as rewarding as empowering people.
That is why I created 30 days of python."""
print(multiline_string)
```
```
I am a teacher and enjoy teaching.
I didn't find anything as rewarding as empowering people.
That is why I created 30 days of python.
I am a teacher and enjoy teaching.
I didn't find anything as rewarding as empowering people.
That is why I created 30 days of python.
```

String Concatenation

In [97]:
```
# String Concatenation
first_name = 'Sania'
last_name = 'Tabassum'
space = ' '
full_name = first_name  +  space + last_name
```

```
print(full_name) # Sania Tabassum
# Checking length of a string using len() builtin function
print(len(first_name))  # 5
print(len(last_name))   # 8
print(len(first_name) > len(last_name)) # False
print(len(full_name)) # 14
```

```
Sania Tabassum
5
8
False
14
```

Unpacking characters

In [98]:

```
#### 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 [107…

```
word = "Sania"
u,v,w,x,y = word
print(u)
print(v)
print(w)
print(x)
print(y)
```

```
S
a
n
i
a
```

Accessing characters in strings by index

In [108…

```
# 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 [141…

```
lang = "Conrad"
first_let = lang[0]
print(first_let)
sec_let = lang[1]
print(sec_let)
last_let= len(lang) - 1
last_let = lang[last_index]
print(last_let)
lang
```

```
C
o
d
```

Out[141…

```
'Conrad'
```

If we want to start from right end we can use negative indexing. -1 is the last index

In [116…

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

```
n
o
```

In [119…

```
lang="Damon"
last_let = lang[-1]
print(last_let)
third_let=lang[-3]
print(third_let)
lang
```

```
n
m
```

Out[119…

```
'Damon'
```

Slicing

In [121…

```
# 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(first_three)
print(last_three) # hon
# Another way
last_three = language[-3:]
print(last_three)   # hon
```

```
last_three = language[3:]
print(last_three)   # hon
```

```
Pyt
hon
hon
hon
```

In [131…

```
lang = "Stevee"
fir_three = lang[:3]
print(fir_three)
last_3 = lang[3:]
print(last_3)
```

```
Ste
vee
```

Skipping character while splitting Python strings

In [132…

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

```
Pto
```

In [136…

```
lang = "Virat"
vrt= lang[0:5:2]
print(vrt)
```

```
Vrt
```

In [140…

```
lang = "Avengers"
lol = lang[0:8:3]
print(lol)
```

```
Anr
```

Escape sequence

In [143…

```
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!"
```

\t automatically keeps columns more uniform, especially in tabular data.

In [152…

```
print("lets create a table \nExcited huh?? ")
print("Date\tMonths\tYears")
print("1st \tJan \t1979")
print("19th \tApril \t2004")
```

```
lets create a table 
Excited huh?? 
Date    Months  Years
1st     Jan     1979
19th    April   2004
```

String Methods

capitalize(): Converts the first character the string to Capital Letter

In [145…

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

```
Thirty days of python
```

In [153…

```
sent = "how are you ? how may i help you??"
print(sent.capitalize())
```

```
How are you ? how may i help you??
```

# count(): returns occurrences of substring in string, count(substring, start=.., end=..)

In [154…

```
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 [157…
```
letters = "asssaddddffgggghhhjjjjkkklll"
print(letters.count("a"))
```
```
2
```

In [158…
```
print(letters.count("g"))
```
```
4
```

In [159…
```
print(letters.count("s"))
```
```
3
```

endswith(): Checks if a string ends with a specified ending

In [161…
```
challenge = 'thirty days of python'
print(challenge.endswith('on'))   # True
print(challenge.endswith('tion')) # False
```
```
True
False
```

In [162…
```
san = " hello everyone my name is Sania Tabassum"
print(san.endswith("nia"))
```
```
False
```

In [163…
```
print(san.endswith("ssum"))
```
```
True
```

# expandtabs(): Replaces tab character with spaces, default tab size is 8. It takes tab size argument

In [164…
```
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 [177…
```
san = " hello\t everyone\tmy\tname\tis\tSania\tTabassum"
print(san.expandtabs())
print(san.expandtabs(16))
```
```
 hello   everyone       my      name    is      Sania   Tabassum
 hello           everyone       my              name            is              S
ania           Tabassum
```

find(): Returns the index of first occurrence of substring

In [178…

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

```
5
0
```

In [182…

```
soil = "Tree removal weakens soil, increasing landslide risk."
print(soil.find(","))
print(soil.find("risk"))
print(soil[48])
```

```
25
48
r
```

format() formats string into nicer output

In [183…

```
first_name = 'Asabeneh'
last_name = 'Yetayeh'
job = 'teacher'
country = 'Finland'
sentence = 'I am {} {}. I am a {}. I live in {}.'.format(first_name, last_name,
print(sentence) # I am Asabeneh Yetayeh. I am a teacher. I live in Finland.
```

```
I am Asabeneh Yetayeh. I am a teacher. I live in Finland.
```

In [194…

```
yes =  "I went to the park near my house."
wea = "bright and sunny, and the sky looked so clear."
running = "playing on the swings and running around happily."
print("Yesterday,{}The weather was {} I saw many children {}".format(yes,wea,run
```

```
Yesterday,I went to the park near my house.The weather was bright and sunny, and
the sky looked so clear. I saw many children playing on the swings and running ar
ound happily.
```

In [195…

```
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 3.14
```

index(): Returns the index of substring

In [197…

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

```
5
0
```

.isalnum() checks if all characters in the string are alphanumeric.

Alphanumeric means: only letters (A–Z, a–z) and/or digits (0–9).

No spaces, no symbols, no punctuation.

Why this prints True

In [198…
```python
challenge = 'ThirtyDaysPython'
print(challenge.isalnum()) # True
```
```
True
```

In [199…
```python
word = "hello there !!"
print(word.isalnum())
```
```
False
```

In [200…
```python
hallenge = '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
False
False
```

isalpha(): Checks if all characters are alphabets

Returns True if all characters are alphabetic.

Returns False if the string contains:

spaces

numbers

symbols (like !, @, #)

is empty

In [202…
```python
challenge = 'days of python'
print(challenge.isalpha()) # False
num = '123'
print(num.isalpha())      # False
```
```
False
False
```

In [205…
```python
lol = "abcdefgh"
print(lol.isalpha())
mom = "123"
print(mom.isalpha())
```
```
True
False
```

isdecimal(): Checks Decimal Characters

In [208…
```python
num = '10'
print(num.isdecimal()) # True
num = '10.5'
print(num.isdecimal()) # False
```
```
True
False
```

isdigit(): Checks Digit Characters

In [217…
```python
challenge = 'Thirty'
print(challenge.isdigit()) # False
challenge = '30'
print(challenge.isdigit())   # True
num = "10.5"
print(num.isdigit())
```
```
False
True
False
```

isidentifier():Checks for valid identifier means it check if a string is a valid variable name

In [212…
```python
challenge = '30DaysOfPython'
print(challenge.isidentifier()) # False, because it starts with a number
challenge = 'thirty_days_of_python'
print(challenge.isidentifier()) # True
```

```
False
True
```

islower():Checks if all alphabets in a string are lowercase

In [213…

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

```
True
False
```

isupper(): returns if all characters are uppercase characters

In [214…

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

```
False
True
```

isnumeric():Checks numeric characters

In [220…

```
num = '10'
print(num.isnumeric())      # True
print('ten'.isnumeric())    # False
print("1/4".isnumeric())   #False
```

```
True
False
False
```

join(): Returns a concatenated string

In [221…

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

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

In [230…

```
num = "1,2,3,4,5"
res = '-- '.join(num)
print(res)
```

```
1-- ,-- 2-- ,-- 3-- ,-- 4-- ,-- 5
```

strip(): Removes both leading and trailing characters

In [233…
```
challenge = ' thirty days of python '
print(challenge.strip('y'))
print(challenge.strip())
```

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

replace(): Replaces substring inside

In [234…
```
challenge = 'thirty days of python'
print(challenge.replace('python', 'coding')) # 'thirty days of coding'
```

```
thirty days of coding
```

In [236…
```
print(challenge.replace('thirty','50'))
```

```
50 days of python
```

split():Splits String from Left

In [237…
```
challenge = 'thirty days of python'
print(challenge.split()) # ['thirty', 'days', 'of', 'python']
```

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

title(): Returns a Title Cased String

In [240…
```
challenge = 'thirty days of python'
print(challenge.title()) # Thirty Days Of Python
```

```
Thirty Days Of Python
```

In [241…
```
ss = "as if the all an"
print(ss.title())
```

```
As If The All An
```

swapcase(): Checks if String Starts with the Specified String

In [243…
```
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
```

startswith(): Checks if String Starts with the Specified String

In [244…

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

```
True
False
```