Practice

May 26, 2022

[5]:
```
#try your first python output
print('Hello World')
```

```
Hello World
```

[6]:
```
#check python version
import sys
print(sys.version)
```

```
3.7.12 | packaged by conda-forge | (default, Oct 26 2021, 06:08:53)
[GCC 9.4.0]
```

[7]:
```
#practice on writing comments
print("Hello, Python!") #this line prints a string
#print("Hi")
```

```
Hello, Python!
```

[8]:
```
#print string as error message
frint("Hello, World!")
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
/tmp/ipykernel_338/2348191561.py in <module>
      1 #print string as error message
----> 2 frint("Hello, World!")

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

[9]:
```
#try to see built in error message
print("Hello, Python)
```

```
  File "/tmp/ipykernel_338/2372593944.py", line 2
    print("Hello, Python)
                         ^
SyntaxError: EOL while scanning string literal
```

[10]:
```
#print string and error to see the running order
print('This will be printed")
frint("This will cause an error")
print("This will NOT be printed")
```

```
  File "/tmp/ipykernel_338/2184925210.py", line 2
    print('This will be printed")
                                 ^
SyntaxError: EOL while scanning string literal
```

[14]:
```
#type of 12
type(12)
```

[14]: int

[12]:
```
#type of 2.4
type(2.4)
```

[12]: float

[13]:
```
#type of "Hello, Python 101!"
type("Hello, Python 101!")
```

[13]: str

[15]:
```
#Convert 2 to a float
float(2)
```

[15]: 2.0

[16]:
```
#Convert 1 to a bool
bool(1)
```

[16]: True

[17]:
```
#COnvert integer 2 to a float and check its type
type(float(2))
```

[17]: float

[18]:
```
#casting 1.1 to an integer will result to loss of information
int(1.1)
```

[18]: 1

[19]:
```
#convert a string to an integer
int('1')
```

[19]: 1

[1]:
```
# Convert a string into an integer with error
int("1 or 2")
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
/tmp/ipykernel_3439/488786847.py in <module>
      1 # Convert a string into an integer with error
----> 2 int("1 or 2")

ValueError: invalid literal for int() with base 10: '1 or 2'
```

[21]:
```
# Convert tht string "1.2" into a float
float("1.2")
```

[21]: 1.2

[22]:
```
# Convert an integer to a string
str(1)
```

[22]: '1'

[25]:
```
# Type of True
type(True)
```

[25]: bool

[28]:
```
#Convert True to integer
int(True)
```

[28]: 1

[29]:
```
# Convert 1 to boolean
bool(1)
```

[29]: True

[30]:
```
# Convert 0 to boolean
bool(0)
```

[30]: False

[31]:
```
# Convert True to float
float(True)
```

[31]: 1.0

[35]:
```
# Store values into variable
x = 43 + 60 + 16 + 41
# Print out the value in variable
x
```

[35]: 160

[36]:
```
# Use another variable to store the result of the operation between variable␣
↪and value
```

[37]:
```
y = x/60
y
```

[37]: 2.6666666666666665

[38]:
```
# Overwrite variable with new value
x = x/60
x
```

[38]: 2.6666666666666665

[40]:
```
# Name the variables meaningfully
total_min = 43 + 42 + 57
total_min
```

[40]: 142

[41]:
```
total_hours = total_min/60
total_hours
```

[41]: 2.3666666666666667

[42]:
```
total_hours = (43 + 42 + 57) / 60
total_hours
```

[42]: 2.3666666666666667

[52]:
```
Name ='Michael Jackson'
Name
```

[52]:
```
'Michael Jackson'
```

[57]:
```
Statement = Name +' is the best'
Statement
```

```
'Michael Jackson is the best'
```

[61]:
```
3*" Michael Jackson"
```

```
' Michael Jackson Michael Jackson Michael Jackson'
```

[63]:
```
print("Michael Jackson\nis the best")
```

```
Michael Jackson
is the best
```

[92]:
```
print("Michael Jackson \t is the best")
```

```
Michael Jackson 	 is the best
```

[65]:
```
print("Michael Jackson\\is the best")
```

```
Michael Jackson\is the best
```

[66]:
```
print(r"Michael Jackson is the best")
```

```
Michael Jackson is the best
```

[73]:
```
A="Thriller is the sixth studio album"
```

[76]:
```
B=A.upper()
B
```

```
'THRILLER IS THE SIXTH STUDIO ALBUM'
```

[78]:
```
A="Michael Jackson is the best"
B=A.replace("Michael","Janet")
B
```

```
'Janet Jackson is the best'
```

[80]:
```
Numbers=("0123456")
Numbers[::2]
```

```
'0246'
```

[81]:
```
"0123456". find("1")
```

[81]:
```
1
```

[83]:
```
len("Michael Jackson")
```

[83]:
```
15
```

[84]:
```
Name[0:4]
```

[84]:
```
'Mich'
```

[86]:
```
Name[8:15]
```

[86]:
```
'Jackson'
```

[87]:
```
Name[::2]
```

[87]:
```
'McalJcsn'
```

[89]:
```
#Concentrate two strings
statement="Michael Jackson" +" is the best"
statement
```

[89]:
```
'Michael Jackson is the best'
```

[94]:
```
3 + 2 * 2
```

[94]:
```
7
```

[99]:
```
A=(0,1,2,3)
A[-1]
```

[99]:
```
3
```

[100]:
```
B=["a","b","c"]
B[1:]
```

[100]:
```
['b', 'c']
```

[105]:
```
tuple1 = ("disco",10,1.2 )
tuple1
```

[105]:
```
('disco', 10, 1.2)
```

[111]:
```
tuple1[0]
```

[111]:
```
'disco'
```

[112]:
```
print(tuple1[0])
print(tuple1[1])
print(tuple1[2])
```

```
disco
10
1.2
```

[114]:
```
print(type(tuple1[0]))
print(type(tuple1[1]))
print(type(tuple1[2]))
```

```
<class 'str'>
<class 'int'>
<class 'float'>
```

[115]:
```
print(type(tuple1[-3]))
print(type(tuple1[-2]))
print(type(tuple1[-1]))
```

```
<class 'str'>
<class 'int'>
<class 'float'>
```

[116]:
```
#Use negative index to get the value of the last element
tuple1[-1]
```

[116]: 1.2

[120]:
```
#Use negative index to get the value of the second last element
tuple1[-2]
```

[120]: 10

[121]:
```
#Combine two tuples
tuple2 = tuple1+("hardrock", 10)
tuple2
```

[121]: ('disco', 10, 1.2, 'hardrock', 10)

[122]:
```
#slice from index 0 to index 2
tuple1[0:3]
```

[122]: ('disco', 10, 1.2)

[123]:
```
len(tuple2)
```

[123]: 5

[124]:
```
# A sample tuple
Ratings = (0, 9, 6, 5, 10, 8, 9, 6, 2)
```

[129]:
```
ratingsSorted = sorted(Ratings)
ratingsSorted
```

[129]:
```
[0, 2, 5, 6, 6, 8, 9, 9, 10]
```

[130]:
```
# Create a Nest Tuple
NestedT =(1, 2, ("pop", "rock") ,(3,4),("disco",(1,2)))
```

[3]:
```
NestedT =(1, 2, ("pop", "rock") ,(3,4),("disco",(1,2)))
print("Element 0 of Tuple: ", NestedT[0])
print("Element 1 of Tuple: ", NestedT[1])
print("Element 2 of Tuple: ", NestedT[2])
print("Element 3 of Tuple: ", NestedT[3])
print("Element 4 of Tuple: ", NestedT[4])
```

```
Element 0 of Tuple:  1
Element 1 of Tuple:  2
Element 2 of Tuple:  ('pop', 'rock')
Element 3 of Tuple:  (3, 4)
Element 4 of Tuple:  ('disco', (1, 2))
```

[4]:
```
print("Element 0 of Tuple: ", NestedT[0])
```

```
Element 0 of Tuple:  1
```

[5]:
```
print("Element 2,1 of Tuple:", NestedT[2][1])
```

```
Element 2,1 of Tuple: rock
```

[7]:
```
print("Element 3,1 of Tuple:", NestedT[3][1])
```

```
Element 3,1 of Tuple: 4
```

[8]:
```
print("Element 2,0 of Tuple:", NestedT[2][0])
```

```
Element 2,0 of Tuple: pop
```

[148]:
```
# Print the first element in the second nested tuples
NestedT[2][1][0]
```

[148]:
```
'r'
```

[146]:
```
NestedT[2]
```

[146]:
```
('pop', 'rock')
```

[150]:
```
# Print the second element in the second nested tuples
NestedT[2][1][1]
```

[150]:
```
'o'
```

[152]:
```
# Print the first element in the second nested tuples
NestedT[4][1][0]
```

[152]:
```
1
```

[154]:
```
genres_tuple = ("pop", "rock", "soul", "hard rock", "soft rock", \
                "R&B", "progressive rock", "disco")
genres_tuple
```

[154]:
```
('pop',
 'rock',
 'soul',
 'hard rock',
 'soft rock',
 'R&B',
 'progressive rock',
 'disco')
```

[155]:
```
#access the element with respect to index 3
genres_tuple[3]
```

[155]:
```
'hard rock'
```

[156]:
```
# Generate a sorted list from the tuple C_tuple = (-5, 1, -3)
C_tuple=(-5,1,-3)
C_list = sorted(C_tuple)
C_list
```

[156]:
```
[-5, -3, 1]
```

[157]:
```
#find the first index of disco
genres_tuple.index("disco")
```

[157]:
```
7
```

[158]:
```
# Find first two elements of the tuple genres_tuple
genres_tuple[0:2]
```

[158]:
```
('pop', 'rock')
```

[13]:
```
# Create a list
```

```
L = ["Michael Jackson", 10.1, 1982]
L
```

[13]:
```
['Michael Jackson', 10.1, 1982]
```

[15]:
```
# Print the elements on each index
print(L[0])
print(L[1])
print(L[2])
```

```
Michael Jackson
10.1
1982
```

[ ]:

[2]:
```
S={'A','B','C'}

U={'A','Z','C'}

U.union(S)
```

[2]:
```
{'A', 'B', 'C', 'Z'}
```

[17]:
```
# Use extend to add elements to list

L = [ "Michael Jackson", 10.2]
L.extend(['pop', 10])
L
```

[17]:
```
['Michael Jackson', 10.2, 'pop', 10]
```

[18]:
```
# Use append to add elements to list

L = [ "Michael Jackson", 10.2]
L.append(['pop', 10])
L
```

[18]:
```
['Michael Jackson', 10.2, ['pop', 10]]
```

[29]:
```
# Change the element based on the index

A = ["disco", 10, 1.2]
print('Before change:', A)
A[0] = 'hardrock'
print('After change:', A)
```

```
Before change: ['disco', 10, 1.2]
After change: ['hardrock', 10, 1.2]
```

[31]:
```
# Delete the element based on the index

A = ["disco", 10, 1.2]
print('Before change:', A)
del(A[0])
print('After change:', A)
```

```
Before change: ['disco', 10, 1.2]
After change: [10, 1.2]
```

[33]:
```
# Split the string, default is by space
'hard rock'.split()
```

[33]:
```
['hard', 'rock']
```

[35]:
```
# Split the string by comma
'A, B, C, D, E'.split()
```

[35]:
```
['A,', 'B,', 'C,', 'D,', 'E']
```

[36]:
```
# Copy (copy by reference) the list A
A = ["hard rock", 10, 1.2]
B=A
print('A:', A)
print('B:', B)
```

```
A: ['hard rock', 10, 1.2]
B: ['hard rock', 10, 1.2]
```

[38]:
```
a_list=[1, 'hello', [1, 2, 3], True]
a_list[1:3]
```

[38]:
```
['hello', [1, 2, 3]]
```

[39]:
```
# Write your code below and press Shift+Enter to execute
A= [1, 'a']
B= [2, 1, 'd']
A+B
```

[39]:
```
[1, 'a', 2, 1, 'd']
```

[1]:
```
# Condition Equal

a = 5
a == 6
```

[1]: False

[2]:
```
# Greater than Sign

i = 6
i > 5
```

[2]: True

[3]:
```
# Greater than Sign

i = 2
i > 5
```

[3]: False

[4]:
```
# Inequality Sign

i = 2
i != 6
```

[4]: True

[5]:
```
# Inequality Sign

i = 6
i != 6
```

[5]: False

[6]:
```
# Use Equality sign to compare the strings

"ACDC" == "Michael Jackson"
```

[6]: False

[7]:
```
# Use Inequality sign to compare the strings

"ACDC" != "Michael Jackson"
```

[7]: True

[9]:
```
# If statement example

age = 19
#age = 18
```

```
#expression that can be true or false
if age > 18:

    #within an indent, we have the expression that is run if the condition is␣
 ↪true
    print("you can enter" )

#The statements after
print("move on")
```

```
you can enter
move on
```

[10]:
```
# Else statement example

age = 18
# age = 19

if age > 18:
    print("you can enter" )
else:
    print("go see Meat Loaf" )

print("move on")
```

```
go see Meat Loaf
move on
```

[11]:
```
# Elif statment example

age = 18

if age > 18:
    print("you can enter" )
elif age == 18:
    print("go see Pink Floyd")
else:
    print("go see Meat Loaf" )

print("move on")
```

```
go see Pink Floyd
move on
```

[12]:
```
# Condition statement example

album_year = 1983
```

```
album_year = 1970

if album_year > 1980:
    print("Album year is greater than 1980")

print('do something..')
```

```
do something..
```

[34]:
```
birth_date = 1997
birth_date = 1992

if birth_date > 1997:
    print("accredited")
print("do something..")
```

```
do something..
```

[31]:
```
# Condition statement example

album_year = 1980

if(album_year > 1979) and (album_year < 1990):
    print ("Album year was in between 1980 and 1989")

print("")
print("Do Stuff..")
```

```
Album year was in between 1980 and 1989

Do Stuff..
```

[39]:
```
date_of_birth = 1997

if(date_of_birth > 1997 ) and (date_of_birth < 2007):
    print("date_of_birth was greater than 10 years interval")
else:
    print("Not Allowed")
```

```
Not Allowed
```

[40]:
```
# Condition statement example

album_year = 1990

if(album_year < 1980) or (album_year > 1989):
    print ("Album was not made in the 1980's")
else:
```

```
    print("The Album was made in the 1980's ")
```

```
The Album was made in the 1980's
```

[42]:
```
# Condition statement example

album_year = 1983

if not (album_year == 1984):
    print ("Album year is not 1984")
```

```
Album year is not 1984
```

[45]:
```
# Write an if statement to determine if an album came out before 1980 or in the␣
 ↪years: 1991 or 1993. If the condition is true print out the year the album␣
 ↪came out.

album_year = 1979

if album_year < 1980 or album_year == 1991 or album_year == 1993:
    print("This album came out in the year", album_year)
```

```
This album came out in the year 1979
```

[47]:
```
#Find the value of <code> x </code> that will print out the sequence <code>␣
 ↪1,2,..,10

x = 11
y = 1

while y != x:
    print(y)
    y = y + 1
```

```
1
2
3
4
5
6
7
8
9
10
```

[48]:
```
# For loop example
```

```
dates = [1982,1980,1973]
N = len(dates)

for i in range(N):
    print(dates[i])
```

```
1982
1980
1973
```

[49]:
```
# Example of for loop

for i in range(0, 8):
    print(i)
```

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

[50]:
```
# Exmaple of for loop, loop through list

for year in dates:
    print(year)
```

```
1982
1980
1973
```

[68]:
```
# Use for loop to change the elements in list

squares = ['red', 'yellow', 'green', 'purple', 'blue']

for i in range(0, 5):
    print("Before square ", i, "is", squares[i])
    squares[i] = 'white'
    print("After square ", i, "is", squares[i])
```

```
Before square  0 is red
After square  0 is white
Before square  1 is yellow
After square  1 is white
Before square  2 is green
After square  2 is white
```

```
Before square  3 is purple
After square  3 is white
Before square  4 is blue
After square  4 is white
```

[63]:
```
# Use for loop to change the elements in list

squares = ['red', 'yellow', 'green', 'purple', 'blue']

for i in range(0, 5):
    print("Before square ", i, 'is',  squares[i])
    squares[i] = 'white'
    print("After square ", i, 'is',  squares[i])
```

```
Before square  0 is red
After square  0 is white
Before square  1 is yellow
After square  1 is white
Before square  2 is green
After square  2 is white
Before square  3 is purple
After square  3 is white
Before square  4 is blue
After square  4 is white
```

[69]:
```
# Loop through the list and iterate on both index and element value

squares=['red', 'yellow', 'green', 'purple', 'blue']

for i, square in enumerate(squares):
    print(i, square)
```

```
0 red
1 yellow
2 green
3 purple
4 blue
```

[70]:
```
squares = ['beans', 'rice', 'yam', 'plantain', 'pasta']

for i, square in enumerate (squares):
    print(i, square)
```

```
0 beans
1 rice
2 yam
3 plantain
4 pasta
```

[77]:
```
squares = ['beans', 'rice', 'yam', 'plantain', 'pasta']

for i in range(0, 5):
    print("First plate", i, "is", squares[i])
    squares[i] = "empty"
    print("Second plate", i, "is", squares[i])
```

```
First plate 0 is beans
Second plate 0 is empty
First plate 1 is rice
Second plate 1 is empty
First plate 2 is yam
Second plate 2 is empty
First plate 3 is plantain
Second plate 3 is empty
First plate 4 is pasta
Second plate 4 is empty
```

[78]:
```
# While Loop Example

dates = [1982, 1980, 1973, 2000]

i = 0
year = dates[0]

while(year != 1973):
    print(year)
    i = i + 1
    year = dates[i]


print("It took ", i ,"repetitions to get out of loop.")
```

```
1982
1980
It took  2 repetitions to get out of loop.
```

[81]:
```
#Print the elements of the following list:
Genres=[ 'rock', 'R\&B', 'Soundtrack', 'R\&B', 'soul', 'pop']
for Genre in Genres:
    print(Genre)
```

```
rock
R\&B
Soundtrack
R\&B
soul
pop
```

[2]:
```
#Write a while loop to display the values of the Rating of an album playlist␣
 ↪stored in the list PlayListRatings. If the score is less than 6, exit the␣
 ↪loop. The list PlayListRatings is given by: PlayListRatings = [10, 9.5, 10,␣
 ↪8, 7.5, 5, 10, 10]

PlayListRatings = [10, 9.5, 10, 8, 7.5, 5, 10, 10]
i = 0
Rating = PlayListRatings[0]
while(i < len(PlayListRatings) and Rating >= 6):
    Rating = PlayListRatings[i]
    print(Rating)
    i = i + 1
```

```
10
9.5
10
8
7.5
5
```

[3]:
```
#Write a while loop to copy the strings 'orange' of the list squares to the␣
 ↪list new_squares. Stop and exit the loop if the value on the list is not␣
 ↪'orange':

squares = ['orange', 'orange', 'purple', 'blue ', 'orange']
new_squares = []
i = 0
while(i < len(squares) and squares[i] == 'orange'):
    new_squares.append(squares[i])
    i = i + 1
print (new_squares)
```

```
['orange', 'orange']
```

[5]:
```
a=1

def add(b):

    return a+b

c=add(10)
```

[8]:
```
a=1

def add(b):
    return a+b
```

```
    c=add(10)
```

[7]:
```
# Define a function for multiple two numbers

def Mult(a, b):
    c = a * b
    return(c)
    print('This is not printed')

result = Mult(12,2)
print(result)
```

```
24
```

[9]:
```
# Get a help on add function

help(add)
```

```
Help on function add in module __main__:

add(b)
```

[10]:
```
# Call the function add()

add(1)
```

[10]: 2

[11]:
```
# First function example: Add 1 to a and store as b

def add(a):
    """
    add 1 to a
    """
    b = a + 1
    print(a, "if you add one", b)
    return(b)
```

[12]:
```
# Call the function add()

add(1)
```

```
1 if you add one 2
```

[12]: 2

[13]:
```
# Call the function add()

add(2)
```

```
2 if you add one 3
```

[13]: 3

[14]:
```
# Use mult() multiply two integers

Mult(2, 3)
```

[14]: 6

[15]:
```
# Use mult() multiply two floats

Mult(10.0, 3.14)
```

[15]: 31.400000000000002

[18]:
```
# Use mult() multiply two different type values together

Mult(2, "Michael Jackson ")
```

[18]: 'Michael Jackson Michael Jackson '

[1]:
```
# Function Definition

def square(a):

    # Local variable b
    b = 1
    c = a * a + b
    print(a, "if you square + 1", c)
    return(c)
```

[20]:
```
# Initializes Global variable

x = 3
# Makes function call and return function a y
y = square(x)
y
```

```
3 if you square + 1 10
```

[20]: 10

[2]:
```
def f(a,b):

    return
```

[3]:
```
a=4
b=2

if a*b==f(a,b):
    print("Correct.")
else:
    print("Incorrect.")
```

```
Incorrect.
```

[4]:
```
def g(c):

    return
```

[5]:
```
c=[1,2,3,4,5]

if sum(c)==g(c):
    print("Correct.")
else:
    print("Incorrect.")
```

```
Incorrect.
```

[1]:
```
# Define a function for multiple two numbers

def Mult(a, b):
    c = a * b
    return(c)
    print('This is not printed')

result = Mult(12,2)
print(result)
```

```
24
```

[2]:
```
# Use mult() multiply two different type values together

Mult(2, "Michael Jackson ")
```

[2]:
```
'Michael Jackson Michael Jackson '
```

[3]:
```
# Use mult() multiply two floats
```

```
Mult(10.0, 3.14)
```

[3]: 31.400000000000002

[4]:
```
# Use mult() multiply two integers

Mult(2, 3)
```

[4]: 6

[5]:
```
# Function Definition

def square(a):

    # Local variable b
    b = 1
    c = a * a + b
    print(a, "if you square + 1", c)
    return(c)
```

[6]:
```
# Initializes Global variable

x = 3
# Makes function call and return function a y
y = square(x)
y
```

```
3 if you square + 1 10
```

[6]: 10

[7]:
```
# Directly enter a number as parameter

square(2)
```

```
2 if you square + 1 5
```

[7]: 5

[9]:
```
# Define functions, one with return value None and other without return value

def MJ():
    print('Michael Jackson')

def MJ1():
    print('Michael Jackson')
    return(None)
```

[10]:
```
# See what functions returns are

print(MJ())
print(MJ1())
```

```
Michael Jackson
None
Michael Jackson
None
```

[11]:
```
# Define the function for combining strings

def con(a, b):
    return(a + b)
```

[12]:
```
# Test on the con() function

con("This ", "is")
```

[12]: 'This is'

[13]:
```
# a and b calculation block1

a1 = 4
b1 = 5
c1 = a1 + b1 + 2 * a1 * b1 - 1
if(c1 < 0):
    c1 = 0
else:
    c1 = 5
c1
```

[13]: 5

[14]:
```
# a and b calculation block2

a2 = 0
b2 = 0
c2 = a2 + b2 + 2 * a2 * b2 - 1
if(c2 < 0):
    c2 = 0
else:
    c2 = 5
c2
```

[14]: 0

[15]:
```
# Make a Function for the calculation above

def Equation(a,b):
    c = a + b + 2 * a * b - 1
    if(c < 0):
        c = 0
    else:
        c = 5
    return(c)
```

[16]:
```
a1 = 4
b1 = 5
c1 = Equation(a1, b1)
c1
```

[16]: 5

[17]:
```
a2 = 0
b2 = 0
c2 = Equation(a2, b2)
c2
```

[17]: 0

[18]:
```
# Build-in function print()

album_ratings = [10.0, 8.5, 9.5, 7.0, 7.0, 9.5, 9.0, 9.5]
print(album_ratings)
```

```
[10.0, 8.5, 9.5, 7.0, 7.0, 9.5, 9.0, 9.5]
```

[19]:
```
# Use sum() to add every element in a list or tuple together

sum(album_ratings)
```

[19]: 70.0

[20]:
```
# Show the length of the list or tuple

len(album_ratings)
```

[20]: 8

[21]:
```
# Function example

def type_of_album(artist, album, year_released):
```

```
    print(artist, album, year_released)
    if year_released > 1980:
        return "Modern"
    else:
        return "Oldie"

x = type_of_album("Michael Jackson", "Thriller", 1980)
print(x)
```

```
Michael Jackson Thriller 1980
Oldie
```

[23]:
```
# Print the list using for loop

def PrintList(the_list):
    for element in the_list:
        print(element)
```

[24]:
```
# Implement the printlist function

PrintList(['1', 1, 'the man', "abc"])
```

```
1
1
the man
abc
```

[25]:
```
# Example for setting param with default value

def isGoodRating(rating=4):
    if(rating < 7):
        print("this album sucks it's rating is",rating)

    else:
        print("this album is good its rating is",rating)
```

[26]:
```
isGoodRating()
isGoodRating(10)
```

```
this album sucks it's rating is 4
this album is good its rating is 10
```

[27]:
```
# Example of global variable

artist = "Michael Jackson"
def printer1(artist):
    internal_var1 = artist
```

```
    print(artist, "is an artist")

printer1(artist)
# try runningthe following code
#printer1(internal_var1)
```

```
Michael Jackson is an artist
```

[28]:
```
artist = "Michael Jackson"

def printer(artist):
    global internal_var
    internal_var= "Whitney Houston"
    print(artist,"is an artist")

printer(artist)
printer(internal_var)
```

```
Michael Jackson is an artist
Whitney Houston is an artist
```

[33]:
```
# Example of global variable

myFavouriteBand = "AC/DC"

def getBandRating(bandname):
    if bandname == myFavouriteBand:
        return 10.0
    else:
        return 0.0

print("AC/DC's rating is:", getBandRating("AC/DC"))
print("Deep Purple's rating is:",getBandRating("Deep Purple"))
print("My favourite band is:", myFavouriteBand)
```

```
AC/DC's rating is: 10.0
Deep Purple's rating is: 0.0
My favourite band is: AC/DC
```

[34]:
```
# Deleting the variable "myFavouriteBand" from the previous example to␣
 ↪demonstrate an example of a local variable

del myFavouriteBand

# Example of local variable

def getBandRating(bandname):
```

```
    myFavouriteBand = "AC/DC"
    if bandname == myFavouriteBand:
        return 10.0
    else:
        return 0.0

print("AC/DC's rating is: ", getBandRating("AC/DC"))
print("Deep Purple's rating is: ", getBandRating("Deep Purple"))
print("My favourite band is", myFavouriteBand)
```

```
AC/DC's rating is:  10.0
Deep Purple's rating is:  0.0
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
/tmp/ipykernel_69/1053098295.py in <module>
     14 print("AC/DC's rating is: ", getBandRating("AC/DC"))
     15 print("Deep Purple's rating is: ", getBandRating("Deep Purple"))
---> 16 print("My favourite band is", myFavouriteBand)

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

[35]:
```
# Example of global variable and local variable with the same name

myFavouriteBand = "AC/DC"

def getBandRating(bandname):
    myFavouriteBand = "Deep Purple"
    if bandname == myFavouriteBand:
        return 10.0
    else:
        return 0.0

print("AC/DC's rating is:",getBandRating("AC/DC"))
print("Deep Purple's rating is: ",getBandRating("Deep Purple"))
print("My favourite band is:",myFavouriteBand)
```

```
AC/DC's rating is: 0.0
Deep Purple's rating is:  10.0
My favourite band is: AC/DC
```

[36]:
```
def printAll(*args): # All the arguments are 'packed' into args which can be␣
 ↪treated like a tuple
    print("No of arguments:", len(args))
    for argument in args:
        print(argument)
```

```
#printAll with 3 arguments
printAll('Horsefeather','Adonis','Bone')
#printAll with 4 arguments
printAll('Sidecar','Long Island','Mudslide','Carriage')
```

```
No of arguments: 3
Horsefeather
Adonis
Bone
No of arguments: 4
Sidecar
Long Island
Mudslide
Carriage
```

[37]:
```
def printDictionary(**args):
    for key in args:
        print(key + " : " + args[key])

printDictionary(Country='Canada',Province='Ontario',City='Toronto')
```

```
Country : Canada
Province : Ontario
City : Toronto
```

[38]:
```
def addItems(list):
    list.append("Three")
    list.append("Four")

myList = ["One","Two"]

addItems(myList)

myList
```

[38]:
```
['One', 'Two', 'Three', 'Four']
```

[40]:
```
#come up with a function that divides first input by second input

def div(a, b):
    return(a/b)
```

[45]:
```
div(20, 5)
```

[45]:
```
4.0
```

[43]:
```
# Use the con function for the following question

def con(a, b):
    return(a + b)
```

[44]:
```
con(2, 2)
```

[44]: 4

[46]:
```
#function connecting list and tuples

con(['a', 1], ['b', 1])
```

[46]: ['a', 1, 'b', 1]

[59]:
```
#Create a  Car  object "my_car" with the given data attributes:
class Car(object):
    def __init__(self,make,model,color):
        self.make=make;
        self.model=model;
        self.color=color;
        self.owner_number=0
    def car_info(self):
        print("make: ",self.make)
        print("model:", self.model)
        print("color:",self.color)
        print("number of owners:",self.owner_number)
    def sell(self):
        self.owner_number=self.owner_number+1

    make="BMW"
    model="M3"
    color="red"
my_car = Car(make,model,color)
```

[60]:
```
my_car.car_info()
```

```
make:  BMW
model: M3
color: red
number of owners: 0
```

[61]:
```
for i in range(5):
    my_car.sell()

my_car.car_info()
```

```
make:  BMW
model: M3
color: red
number of owners: 5
```

[ ]:
```
A =({'a':1,'b':2})
dictionary.keys()
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
/tmp/ipykernel_347/2624116433.py in <module>
      1 A =({'a':1,'b':2})
----> 2 dictionary.keys()

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

[28]:
```
def Delta(x):

    if x==0:

        y=1;

    else:

        y=0;

    return(y)
Delta(0)
```

[28]: 1

[29]:
```
# Create a class Circle

class Circle(object):

    # Constructor
    def __init__(self, radius=3, color='blue'):
        self.radius = radius
        self.color = color

    # Method
    def add_radius(self, r):
        self.radius = self.radius + r
        return(self.radius)

    # Method
    def drawCircle(self):
```

```
        plt.gca().add_patch(plt.Circle((0, 0), radius=self.radius, fc=self.
↪color))
        plt.axis('scaled')
        plt.show()
```

[30]:
```
# Create an object RedCircle

RedCircle = Circle(10, 'red')
```

[31]:
```
# Create a new Rectangle class for creating a rectangle object

class Rectangle(object):

    # Constructor
    def __init__(self, width=2, height=3, color='r'):
        self.height = height
        self.width = width
        self.color = color

    # Method
    def drawRectangle(self):
        plt.gca().add_patch(plt.Rectangle((0, 0), self.width, self.height␣
↪,fc=self.color))
        plt.axis('scaled')
        plt.show()
```

[32]:
```
# Create a new object rectangle

SkinnyBlueRectangle = Rectangle(2, 3, 'blue')
```

[33]:
```
# Print the object attribute height

SkinnyBlueRectangle.height
```

[33]: 3

[34]:
```
# Print the object attribute width

SkinnyBlueRectangle.width
```

[34]: 2

[36]:
```
# Print the object attribute color

SkinnyBlueRectangle.color
```

[36]:
```
'blue'
```

[38]:
```
# Use the drawRectangle method to draw the shape

SkinnyBlueRectangle.drawRectangle()
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
/tmp/ipykernel_347/1239561112.py in <module>
      1 # Use the drawRectangle method to draw the shape
      2
----> 3 SkinnyBlueRectangle.drawRectangle()

/tmp/ipykernel_347/4264899945.py in drawRectangle(self)
     11     # Method
     12     def drawRectangle(self):
---> 13         plt.gca().add_patch(plt.Rectangle((0, 0), self.width, self.
 ↪height ,fc=self.color))
     14         plt.axis('scaled')
     15         plt.show()

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

[39]:
```
# Create a new object rectangle

FatYellowRectangle = Rectangle(20, 5, 'yellow')
```

[40]:
```
# Print the object attribute height

FatYellowRectangle.height
```

[40]:
```
5
```

[41]:
```
# Print the object attribute width

FatYellowRectangle.width
```

[41]:
```
20
```

[42]:
```
# Print the object attribute color

FatYellowRectangle.color
```

[42]:
```
'yellow'
```

[43]:
```
# Use the drawRectangle method to draw the shape

FatYellowRectangle.drawRectangle()
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
/tmp/ipykernel_347/3426178370.py in <module>
      1 # Use the drawRectangle method to draw the shape
      2
----> 3 FatYellowRectangle.drawRectangle()

/tmp/ipykernel_347/4264899945.py in drawRectangle(self)
     11     # Method
     12     def drawRectangle(self):
---> 13         plt.gca().add_patch(plt.Rectangle((0, 0), self.width, self.
 ↪height ,fc=self.color))
     14         plt.axis('scaled')
     15         plt.show()

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

[12]:
```
# Read the Example1.txt

file1 = open("/resources/data/Example1.txt","r")

example1 = "Example1.txt"
file1 = open(example1, "r")
```

```
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
/tmp/ipykernel_373/3763924869.py in <module>
      1 # Read the Example1.txt
      2
----> 3 file1 = open("/resources/data/Example1.txt","a+")
      4
      5 example1 = "Example1.txt"

FileNotFoundError: [Errno 2] No such file or directory: '/resources/data/
 ↪Example1.txt'
```

[ ]:
```
File1 = open("/resources/data/Example1.txt","w")
File1.write("This is line A")
```

```
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
```

```
/tmp/ipykernel_373/91382632.py in <module>
----> 1 File1 = open("/This PC/Desktop/movie night/Example1.txt","w")
      2 File1.write("This is line A")

FileNotFoundError: [Errno 2] No such file or directory: '/This PC/Desktop/movie
 ↪night/Example1.txt'
```

[13]:
```
with open('Example2.txt', 'a+') as testwritefile:
    print("Initial Location: {}".format(testwritefile.tell()))

    data = testwritefile.read()
    if (not data):  #empty strings return false in python
            print('Read nothing')
    else:
            print(testwritefile.read())

    testwritefile.seek(0,0) # move 0 bytes from beginning.

    print("\nNew Location : {}".format(testwritefile.tell()))
    data = testwritefile.read()
    if (not data):
            print('Read nothing')
    else:
            print(data)

    print("Location after read: {}".format(testwritefile.tell()) )
```

```
Initial Location: 0
Read nothing

New Location : 0
Read nothing
Location after read: 0
```

[15]:
```
with open('Example2.txt', 'r+') as testwritefile:
    data = testwritefile.readlines()
    testwritefile.seek(0,0) #write at beginning of file

    testwritefile.write("Line 1" + "\n")
    testwritefile.write("Line 2" + "\n")
    testwritefile.write("Line 3" + "\n")
    testwritefile.write("finished\n")
    #Uncomment the line below
    #testwritefile.truncate()
    testwritefile.seek(0,0)
    print(testwritefile.read())
```

```
Line 1
Line 2
Line 3
finished
```

[16]:
```
with open('Example2.txt', 'r+') as testwritefile:
    data = testwritefile.readlines()
    testwritefile.seek(0,0) #write at beginning of file

    testwritefile.write("Line 1" + "\n")
    testwritefile.write("Line 2" + "\n")
    testwritefile.write("Line 3" + "\n")
    testwritefile.write("finished\n")
    testwritefile.truncate()
    testwritefile.seek(0,0)
    print(testwritefile.read())
```

```
Line 1
Line 2
Line 3
finished
```

[18]:
```
# Copy file to another

with open('Example2.txt','r') as readfile:
    with open('Example3.txt','w') as writefile:
          for line in readfile:
                writefile.write(line)
```

[19]:
```
# Verify if the copy is successfully executed

with open('Example3.txt','r') as testwritefile:
    print(testwritefile.read())
```

```
Line 1
Line 2
Line 3
finished
```

[20]:
```
#Run this prior to starting the exercise
from random import randint as rnd

memReg = 'members.txt'
exReg = 'inactive.txt'
fee =('yes','no')
```

```
def genFiles(current,old):
    with open(current,'w+') as writefile:
        writefile.write('Membership No  Date Joined  Active  \n')
        data = "{:^13}  {:<11}  {:<6}\n"

        for rowno in range(20):
            date = str(rnd(2015,2020))+ '-' + str(rnd(1,12))+'-'+str(rnd(1,25))
            writefile.write(data.format(rnd(10000,99999),date,fee[rnd(0,1)]))


    with open(old,'w+') as writefile:
        writefile.write('Membership No  Date Joined  Active  \n')
        data = "{:^13}  {:<11}  {:<6}\n"
        for rowno in range(3):
            date = str(rnd(2015,2020))+ '-' + str(rnd(1,12))+'-'+str(rnd(1,25))
            writefile.write(data.format(rnd(10000,99999),date,fee[1]))


genFiles(memReg,exReg)
```

[21]:
```
def cleanFiles(currentMem,exMem):
    with open(currentMem,'r+') as writeFile:
        with open(exMem,'a+') as appendFile:
            #get the data
            writeFile.seek(0)
            members = writeFile.readlines()
            #remove header
            header = members[0]
            members.pop(0)

            inactive = [member for member in members if ('no' in member)]
            '''
            The above is the same as

            for member in members:
            if 'no' in member:
                inactive.append(member)
            '''
            #go to the beginning of the write file
            writeFile.seek(0)
            writeFile.write(header)
            for member in members:
                if (member in inactive):
                    appendFile.write(member)
                else:
                    writeFile.write(member)
```

```
            writeFile.truncate()

memReg = 'members.txt'
exReg = 'inactive.txt'
cleanFiles(memReg,exReg)

# code to help you see the files

headers = "Membership No  Date Joined  Active  \n"

with open(memReg,'r') as readFile:
    print("Active Members: \n\n")
    print(readFile.read())

with open(exReg,'r') as readFile:
    print("Inactive Members: \n\n")
    print(readFile.read())
```

```
Active Members:


Membership No  Date Joined  Active
    44401      2017-7-3     yes
    32042      2017-2-19    yes
    71709      2019-9-25    yes
    95765      2016-4-25    yes
    78673      2018-2-13    yes
    73180      2020-1-5     yes
    19736      2018-11-17   yes
    55872      2016-8-22    yes

Inactive Members:


Membership No  Date Joined  Active
    48125      2015-2-20    no
    90862      2019-10-13   no
    64032      2015-3-10    no
    18652      2019-8-25    no
    12446      2018-3-6     no
    49679      2015-7-25    no
    42948      2018-4-1     no
    28071      2020-2-24    no
    10665      2020-7-19    no
    75514      2020-3-4     no
    92714      2015-5-10    no
    28999      2015-10-18   no
```

```
17329       2018-6-12    no
64821       2019-9-24    no
13047       2017-10-16   no
```

[27]:
```
import pandas as pd
df=pd.DataFrame({'a':[1,2,1],'b':[1,1,1]})
```

[31]:
```
df.head()
```

[31]:
```
   a  b
0  1  1
1  2  1
2  1  1
```

[32]:
```
#obtain column a
df['a']
```

[32]:
```
0    1
1    2
2    1
Name: a, dtype: int64
```

[33]:
```
#find the unique values in column a
df['a'].unique()
```

[33]:
```
array([1, 2])
```

[34]:
```
df[df['a']<2]
```

[34]:
```
   a  b
0  1  1
2  1  1
```

[39]:
```
# Dependency needed to install file

!pip install xlrd

# Import required library

import pandas as pd
```

```
Requirement already satisfied: xlrd in
/home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (1.2.0)
```

[40]:
```
# Read data from CSV file
```

```
csv_path = 'https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/
 ↪IBMDeveloperSkillsNetwork-PY0101EN-SkillsNetwork/labs/Module%204/data/
 ↪TopSellingAlbums.csv'
df = pd.read_csv(csv_path)
```

[41]:
```
# Print first five rows of the dataframe

df.head()
```

[41]:
```
             Artist                     Album  Released   Length  \
0  Michael Jackson                   Thriller      1982  0:42:19
1            AC/DC               Back in Black      1980  0:42:11
2       Pink Floyd  The Dark Side of the Moon      1973  0:42:49
3  Whitney Houston               The Bodyguard      1992  0:57:44
4        Meat Loaf             Bat Out of Hell      1977  0:46:33

                         Genre  Music Recording Sales (millions)  \
0               pop, rock, R&B                              46.0
1                    hard rock                              26.1
2             progressive rock                              24.2
3               R&B, soul, pop                              27.4
4  hard rock, progressive rock                              20.6

   Claimed Sales (millions) Released.1 Soundtrack  Rating
0                        65  30-Nov-82        NaN    10.0
1                        50  25-Jul-80        NaN     9.5
2                        45  01-Mar-73        NaN     9.0
3                        44  17-Nov-92          Y     8.5
4                        43  21-Oct-77        NaN     8.0
```

[42]:
```
# Read data from Excel File and print the first five rows

xlsx_path = 'https://s3-api.us-geo.objectstorage.softlayer.net/cf-courses-data/
 ↪CognitiveClass/PY0101EN/Chapter%204/Datasets/TopSellingAlbums.xlsx'

df = pd.read_excel(xlsx_path)
df.head()
```

```
---------------------------------------------------------------------------
ImportError                               Traceback (most recent call last)
/tmp/ipykernel_373/2455837228.py in <module>
      3 xlsx_path = 'https://s3-api.us-geo.objectstorage.softlayer.net/
 ↪cf-courses-data/CognitiveClass/PY0101EN/Chapter%204/Datasets/TopSellingAlbums
 ↪xlsx'
      4
----> 5 df = pd.read_excel(xlsx_path)
      6 df.head()
```

```
~/conda/envs/python/lib/python3.7/site-packages/pandas/util/_decorators.py in␣
↪wrapper(*args, **kwargs)
    309                     stacklevel=stacklevel,
    310                 )
--> 311             return func(*args, **kwargs)
    312
    313         return wrapper

~/conda/envs/python/lib/python3.7/site-packages/pandas/io/excel/_base.py in␣
↪read_excel(io, sheet_name, header, names, index_col, usecols, squeeze, dtype,␣
↪engine, converters, true_values, false_values, skiprows, nrows, na_values,␣
↪keep_default_na, na_filter, verbose, parse_dates, date_parser, thousands,␣
↪comment, skipfooter, convert_float, mangle_dupe_cols, storage_options)
    362     if not isinstance(io, ExcelFile):
    363         should_close = True
--> 364         io = ExcelFile(io, storage_options=storage_options,␣
↪engine=engine)
    365     elif engine and engine != io.engine:
    366         raise ValueError(

~/conda/envs/python/lib/python3.7/site-packages/pandas/io/excel/_base.py in␣
↪__init__(self, path_or_buffer, engine, storage_options)
   1231         self.storage_options = storage_options
   1232
-> 1233         self._reader = self._engines[engine](self._io,␣
↪storage_options=storage_options)
   1234
   1235     def __fspath__(self):

~/conda/envs/python/lib/python3.7/site-packages/pandas/io/excel/_openpyxl.py in␣
↪__init__(self, filepath_or_buffer, storage_options)
    519             passed to fsspec for appropriate URLs (see␣
↪``_get_filepath_or_buffer``)
    520         """
--> 521         import_optional_dependency("openpyxl")
    522         super().__init__(filepath_or_buffer,␣
↪storage_options=storage_options)
    523

~/conda/envs/python/lib/python3.7/site-packages/pandas/compat/_optional.py in␣
↪import_optional_dependency(name, extra, errors, min_version)
    116     except ImportError:
    117         if errors == "raise":
--> 118             raise ImportError(msg) from None
    119         else:
    120             return None
```

```
ImportError: Missing optional dependency 'openpyxl'.  Use pip or conda to␣
↪install openpyxl.
```

[43]:
```
# Access to the column Length

x = df[['Length']]
x
```

[43]:
```
    Length
0  0:42:19
1  0:42:11
2  0:42:49
3  0:57:44
4  0:46:33
5  0:43:08
6  1:15:54
7  0:40:01
```

[44]:
```
# Get the column as a series

x = df['Length']
x
```

[44]:
```
0    0:42:19
1    0:42:11
2    0:42:49
3    0:57:44
4    0:46:33
5    0:43:08
6    1:15:54
7    0:40:01
Name: Length, dtype: object
```

[45]:
```
# Get the column as a dataframe

x = df[['Artist']]
type(x)
```

[45]:
```
pandas.core.frame.DataFrame
```

[46]:
```
# Access to multiple columns

y = df[['Artist','Length','Genre']]
y
```

[46]:
```
             Artist   Length                        Genre
0  Michael Jackson  0:42:19              pop, rock, R&B
```

```
1            AC/DC  0:42:11                   hard rock
2       Pink Floyd  0:42:49             progressive rock
3  Whitney Houston  0:57:44               R&B, soul, pop
4        Meat Loaf  0:46:33  hard rock, progressive rock
5           Eagles  0:43:08   rock, soft rock, folk rock
6         Bee Gees  1:15:54                       disco
7    Fleetwood Mac  0:40:01                   soft rock
```

[48]:
```
# Access the value on the first row and the first column

df.iloc[0, 0]
```

[48]: 'Michael Jackson'

[49]:
```
# Access the value on the second row and the first column

df.iloc[1,0]
```

[49]: 'AC/DC'

[51]:
```
# Access the value on the first row and the third column

df.iloc[0,2]
```

[51]: 1982

[52]:
```
# Access the value on the second row and the third column
df.iloc[1,2]
```

[52]: 1980

[53]:
```
# Access the column using the name

df.loc[1, 'Artist']
```

[53]: 'AC/DC'

[54]:
```
# Access the column using the name

df.loc[0, 'Released']
```

[54]: 1982

[55]:
```
# Access the column using the name

df.loc[1, 'Released']
```

[55]:
```
1980
```

[56]:
```
# Slicing the dataframe

df.iloc[0:2, 0:3]
```

[56]:
```
             Artist          Album  Released
0  Michael Jackson       Thriller      1982
1            AC/DC  Back in Black      1980
```

[57]:
```
# Slicing the dataframe using name

df.loc[0:2, 'Artist':'Released']
```

[57]:
```
             Artist                     Album  Released
0  Michael Jackson                   Thriller      1982
1            AC/DC               Back in Black      1980
2       Pink Floyd  The Dark Side of the Moon      1973
```

[59]:
```
# Use a variable q to store the column Rating as a dataframe
q = df[['Rating']]
q
```

[59]:
```
   Rating
0    10.0
1     9.5
2     9.0
3     8.5
4     8.0
5     7.5
6     7.0
7     6.5
```

[61]:
```
#Assign the variable q to the dataframe that is made up of the column Released␣
 ↪and Artist:

q = df[['Released','Artist']]
q
```

[61]:
```
   Released           Artist
0      1982  Michael Jackson
1      1980            AC/DC
2      1973       Pink Floyd
3      1992  Whitney Houston
4      1977        Meat Loaf
5      1976           Eagles
6      1977         Bee Gees
```

```
7      1977    Fleetwood Mac
```

[64]:
```
#Access the 2nd row and the 3rd column of df:

df.iloc[1,2]
```

[64]:
```
1980
```

[66]:
```
#Use the following list to convert the dataframe index df to characters and␣
 ↪assign it to df_new; find the element corresponding to the row index a and␣
 ↪column 'Artist'. Then select the rows a through d for the column 'Artist'

new_index=['a','b','c','d','e','f','g','h']

df_new=df
df_new.index=new_index
df_new.loc['a', 'Artist']
df_new.loc['a':'d', 'Artist']
```

[66]:
```
a    Michael Jackson
b               AC/DC
c         Pink Floyd
d    Whitney Houston
Name: Artist, dtype: object
```

[68]:
```
#Cast the following list to a numpy array

import numpy as np
a=[1,2,3,4,5]
x=np.array(a)
```

[69]:
```
#Find the type of x using the function type().
type(x)
```

[69]:
```
numpy.ndarray
```

[74]:
```
#Find the shape of the array

x.shape
```

[74]:
```
(5,)
```

[72]:
```
#Find the type of data in the array

x.dtype
```

[72]:
```
dtype('int64')
```

[75]:
```
#Find the mean of the array

x.mean()
```

[75]:
```
3.0
```

[9]:
```
import numpy as np
A = np.array([[1,2],[3,4],[5,6],[7,8]])

B = np.array([[1,2,3],[4,5,6],[7,8,9]])

C = np.dot(A,B)
C
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
/tmp/ipykernel_736/742411534.py in <module>
      4 B = np.array([[1,2,3],[4,5,6],[7,8,9]])
      5
----> 6 C = np.dot(A,B)
      7 C

<__array_function__ internals> in dot(*args, **kwargs)

ValueError: shapes (4,2) and (3,3) not aligned: 2 (dim 1) != 3 (dim 0)
```

[3]:
```
# Import the libraries

import time
import sys
import numpy as np

import matplotlib.pyplot as plt
%matplotlib inline
```

[4]:
```
# Plotting functions

def Plotvec1(u, z, v):

    ax = plt.axes()
    ax.arrow(0, 0, *u, head_width=0.05, color='r', head_length=0.1)
    plt.text(*(u + 0.1), 'u')

    ax.arrow(0, 0, *v, head_width=0.05, color='b', head_length=0.1)
    plt.text(*(v + 0.1), 'v')
    ax.arrow(0, 0, *z, head_width=0.05, head_length=0.1)
```

```
    plt.text(*(z + 0.1), 'z')
    plt.ylim(-2, 2)
    plt.xlim(-2, 2)

def Plotvec2(a,b):
    ax = plt.axes()
    ax.arrow(0, 0, *a, head_width=0.05, color ='r', head_length=0.1)
    plt.text(*(a + 0.1), 'a')
    ax.arrow(0, 0, *b, head_width=0.05, color ='b', head_length=0.1)
    plt.text(*(b + 0.1), 'b')
    plt.ylim(-2, 2)
    plt.xlim(-2, 2)
```

[5]:
```
# Create a python list

a = ["0", 1, "two", "3", 4]
```

[6]:
```
# Print each element

print("a[0]:", a[0])
print("a[1]:", a[1])
print("a[2]:", a[2])
print("a[3]:", a[3])
print("a[4]:", a[4])
```

```
a[0]: 0
a[1]: 1
a[2]: two
a[3]: 3
a[4]: 4
```

[7]:
```
# import numpy library

import numpy as np
```

[8]:
```
# Create a numpy array

a = np.array([0, 1, 2, 3, 4])
a
```

[8]:
```
array([0, 1, 2, 3, 4])
```

[10]:
```
# Print each element

print("a[0]:", a[0])
print("a[1]:", a[1])
print("a[2]:", a[2])
```

```
print("a[3]:", a[3])
print("a[4]:", a[4])
```

```
a[0]: 0
a[1]: 1
a[2]: 2
a[3]: 3
a[4]: 4
```

[11]:
```
# Check the type of the array

type(a)
```

[11]:
```
numpy.ndarray
```

[12]:
```
# Check the type of the values stored in numpy array

a.dtype
```

[12]:
```
dtype('int64')
```

[13]:
```
# Create a numpy array

b = np.array([3.1, 11.02, 6.2, 213.2, 5.2])
```

[14]:
```
# Check the type of array

type(b)
```

[14]:
```
numpy.ndarray
```

[15]:
```
# Check the value type

b.dtype
```

[15]:
```
dtype('float64')
```

[16]:
```
# Create numpy array

c = np.array([20, 1, 2, 3, 4])
c
```

[16]:
```
array([20,  1,  2,  3,  4])
```

[17]:
```
# Assign the first element to 100

c[0] = 100
```

```
c
```

[17]:
```
array([100,   1,   2,   3,   4])
```

[18]:
```
# Assign the 5th element to 0

c[4] = 0
c
```

[18]:
```
array([100,   1,   2,   3,   0])
```

[19]:
```
# Slicing the numpy array

d = c[1:4]
d
```

[19]:
```
array([1, 2, 3])
```

[20]:
```
# Set the fourth element and fifth element to 300 and 400

c[3:5] = 300, 400
c
```

[20]:
```
array([100,   1,   2, 300, 400])
```

[21]:
```
# Create the index list

select = [0, 2, 3]
```

[22]:
```
# Use List to select elements

d = c[select]
d
```

[22]:
```
array([100,   2, 300])
```

[23]:
```
# Assign the specified elements to new value

c[select] = 100000
c
```

[23]:
```
array([100000,      1, 100000, 100000,    400])
```

[24]:
```
# Create a numpy array

a = np.array([0, 1, 2, 3, 4])
a
```

[24]:
```
array([0, 1, 2, 3, 4])
```

[25]:
```
# Get the size of numpy array

a.size
```

[25]:
```
5
```

[26]:
```
# Get the number of dimensions of numpy array

a.ndim
```

[26]:
```
1
```

[28]:
```
# Get the number of dimensions of numpy array

a.shape
```

[28]:
```
(5,)
```

[29]:
```
# Create a numpy array

a = np.array([1, -1, 1, -1])
```

[30]:
```
# Get the mean of numpy array

mean = a.mean()
mean
```

[30]:
```
0.0
```

[31]:
```
# Get the standard deviation of numpy array

standard_deviation=a.std()
standard_deviation
```

[31]:
```
1.0
```

[32]:
```
# Create a numpy array

b = np.array([-1, 2, 3, 4, 5])
b
```

[32]:
```
array([-1,  2,  3,  4,  5])
```

[33]:
```
# Get the biggest value in the numpy array
```

```
max_b = b.max()
max_b
```

[33]: 5

[34]:
```
# Get the smallest value in the numpy array

min_b = b.min()
min_b
```

[34]: -1

[35]:
```
u = np.array([1, 0])
u
```

[35]: array([1, 0])

[36]:
```
v = np.array([0, 1])
v
```

[36]: array([0, 1])

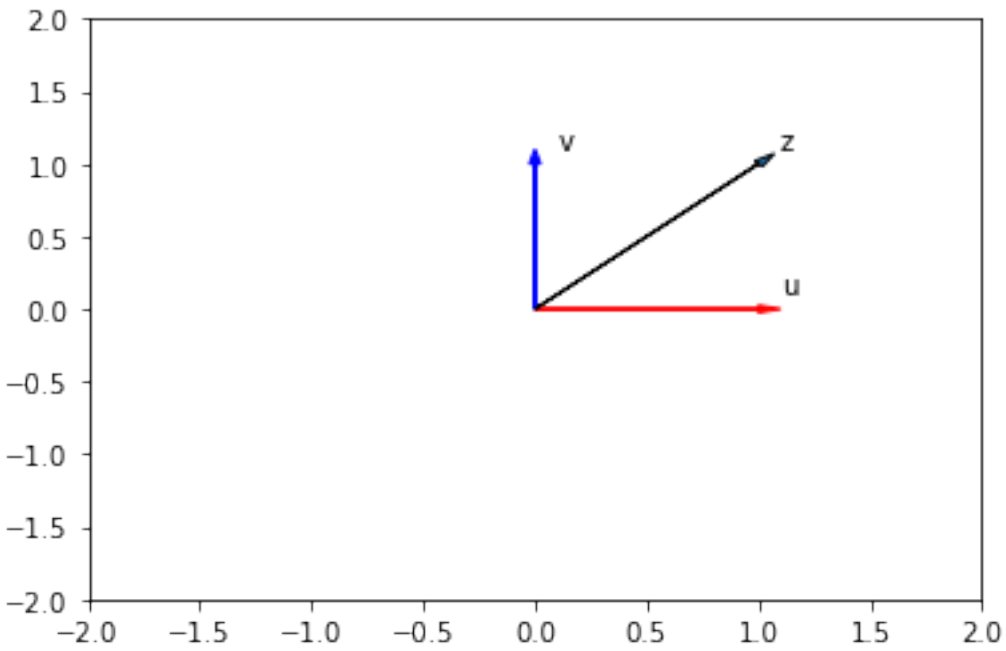
[37]:
```
# Numpy Array Addition

z = u + v
z
```

[37]: array([1, 1])

[38]:
```
# Plot numpy arrays

Plotvec1(u, z, v)
```



[39]:
```
# Create a numpy array

y = np.array([1, 2])
y
```

[39]: array([1, 2])

[40]:
```
# Numpy Array Multiplication

z = 2 * y
z
```

[40]: array([2, 4])

[41]:
```
# Create a numpy array

u = np.array([1, 2])
u
```

[41]: array([1, 2])

[42]:
```
# Create a numpy array

v = np.array([3, 2])
v
```

[42]:
```
array([3, 2])
```

[43]:
```
# Calculate the production of two numpy arrays

z = u * v
z
```

[43]:
```
array([3, 4])
```

[44]:
```
# Calculate the dot product

np.dot(u, v)
```

[44]:
```
7
```

[45]:
```
# Create a constant to numpy array

u = np.array([1, 2, 3, -1])
u
```

[45]:
```
array([ 1,  2,  3, -1])
```

[46]:
```
# Add the constant to array

u + 1
```

[46]:
```
array([2, 3, 4, 0])
```

[47]:
```
# The value of pi

np.pi
```

[47]:
```
3.141592653589793
```

[48]:
```
# Create the numpy array in radians

x = np.array([0, np.pi/2 , np.pi])
```

[49]:
```
# Calculate the sin of each elements

y = np.sin(x)
y
```

[49]:
```
array([0.0000000e+00, 1.0000000e+00, 1.2246468e-16])
```

[60]:
```
# Makeup a numpy array within [-2, 2] and 5 elements
```

```
np.linspace(-2, 2, num=5)
```

[60]:
```
array([-2., -1.,  0.,  1.,  2.])
```

[61]:
```
# Make a numpy array within [-2, 2] and 9 elements

np.linspace(-2, 2, num=9)
```

[61]:
```
array([-2. , -1.5, -1. , -0.5,  0. ,  0.5,  1. ,  1.5,  2. ])
```

[62]:
```
# Make a numpy array within [0, 2] and 100 elements

x = np.linspace(0, 2*np.pi, num=100)
```

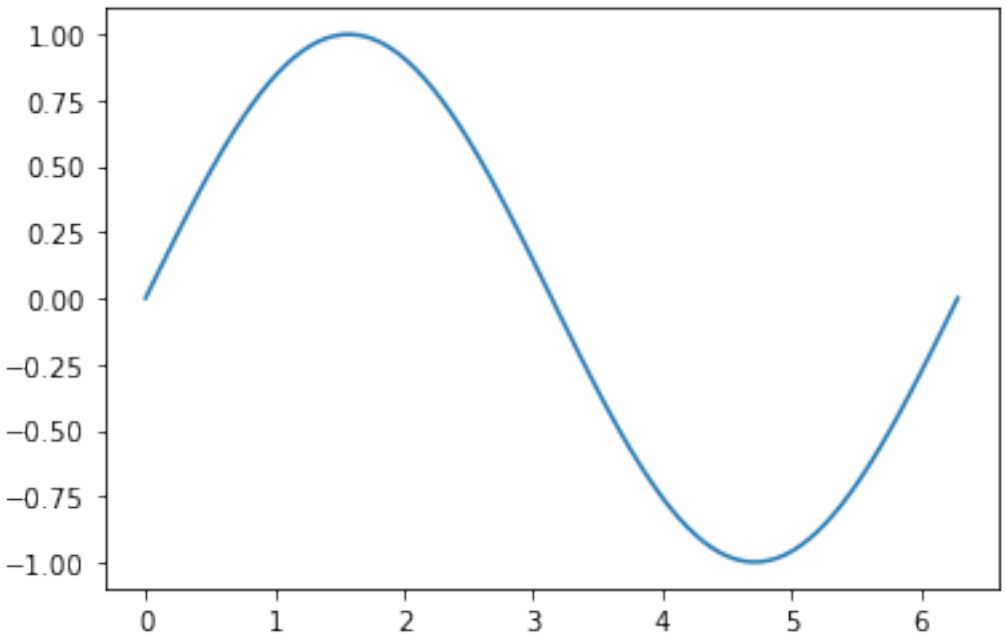
[63]:
```
# Calculate the sine of x list

y = np.sin(x)
```

[64]:
```
# Plot the result

plt.plot(x, y)
```

[64]:
```
[<matplotlib.lines.Line2D at 0x7f61fd476150>]
```



[66]:
```
#Implement the following vector subtraction
u = np.array([1, 0])
v = np.array([0, 1])
z= u - v
z
```

[66]:
```
array([ 1, -1])
```

[68]:
```
#Myltiply the nympy array z with -2
z = np.array([2, 4])
A = z * -2
A
```

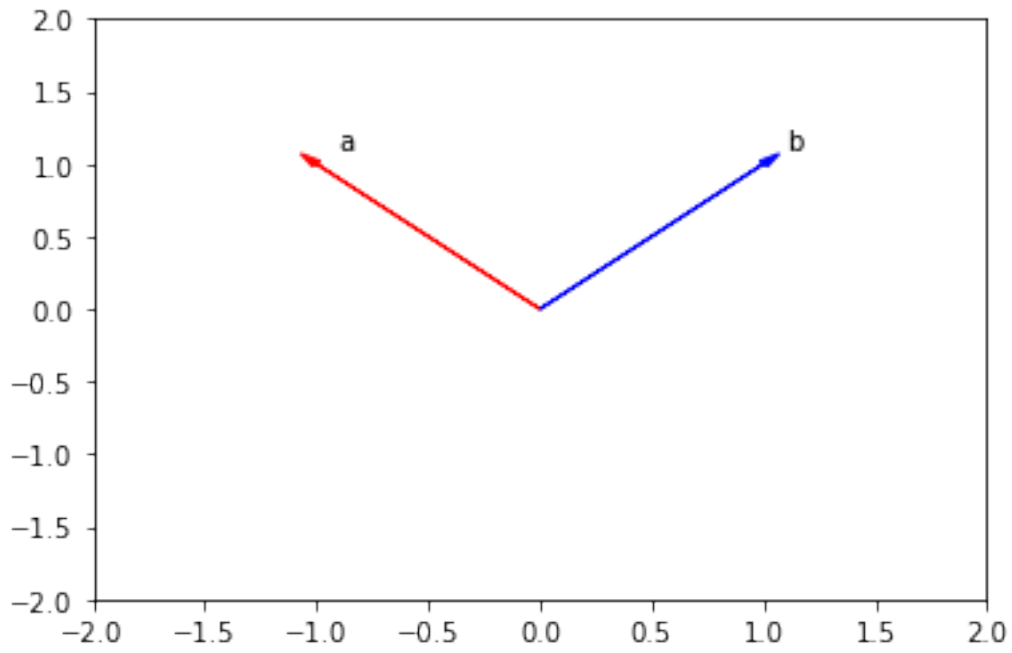
[68]:
```
array([-4, -8])
```

[69]:
```
# Multiply both lists together
x = np.array([1 ,2, 3, 4, 5])
y = np.array([1, 0, 1, 0, 1])
z = x * y
z
```

[69]:
```
array([1, 0, 3, 0, 5])
```

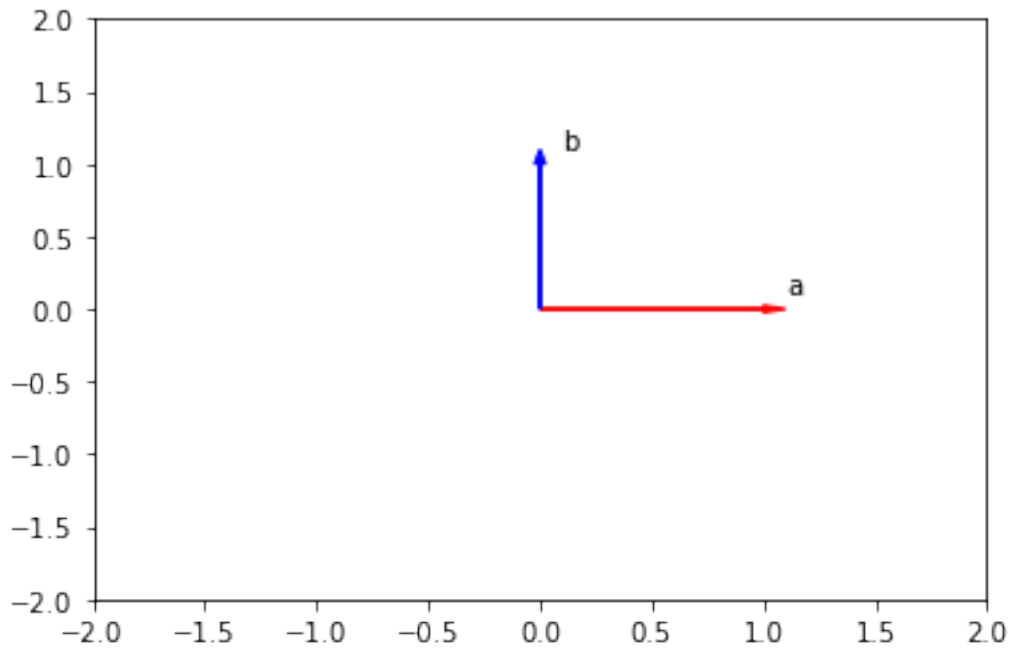
[70]:
```
# Plot the arrays as vectors using the fuction Plotvec2 and find their dot␣
 ↪product:
a = np.array([-1, 1])
b = np.array([1, 1])
Plotvec2(a,b)
print("The dot product is", np.dot(a,b))
```

```
The dot product is 0
```



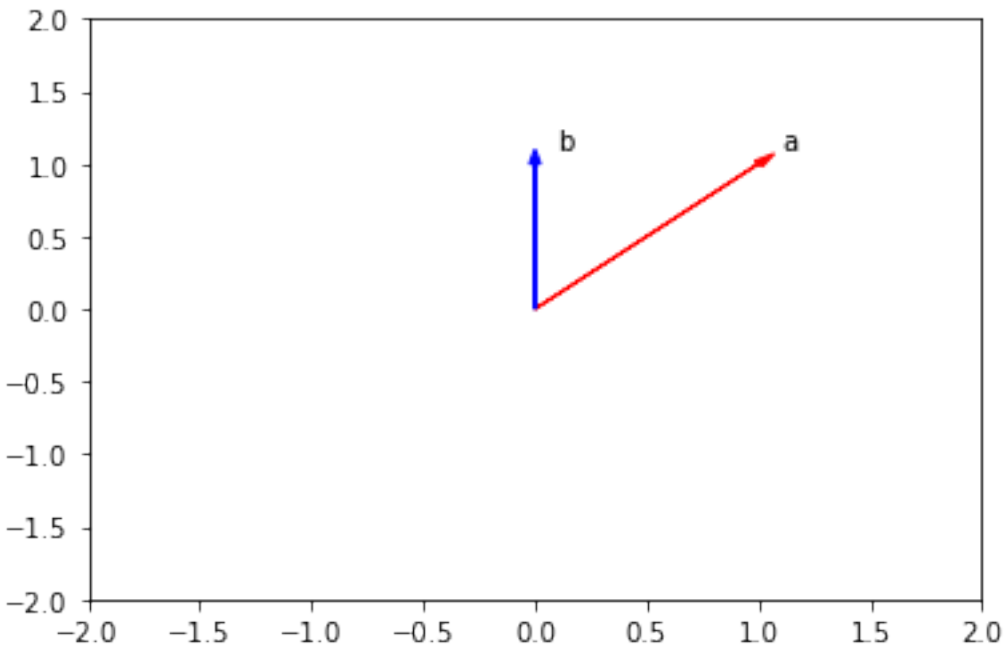
[71]:
```
# Plot the arrays as vectors using the fuction Plotvec2 and find their dot␣
↪product:
a = np.array([1, 0])
b = np.array([0, 1])
Plotvec2(a, b)
print("The dot product is", np.dot(a, b))
```

```
The dot product is 0
```



[72]:
```
# Plot the arrays as vectors using the fuction Plotvec2 and find their dot␣
↪product:
a = np.array([1, 1])
b = np.array([0, 1])
Plotvec2(a, b)
print("The dot product is", np.dot(a, b))
```

```
The dot product is 1
```



[73]:
```
# Import the libraries

import numpy as np
import matplotlib.pyplot as plt
```

[74]:
```
# Create a list

a = [[11, 12, 13], [21, 22, 23], [31, 32, 33]]
a
```

[74]:
```
[[11, 12, 13], [21, 22, 23], [31, 32, 33]]
```

[75]:
```
# Convert list to Numpy Array
# Every element is the same type

A = np.array(a)
A
```

[75]:
```
array([[11, 12, 13],
       [21, 22, 23],
       [31, 32, 33]])
```

[76]:
```
# Show the numpy array dimensions

A.ndim
```

[76]: 2

[77]:
```
# Show the numpy array shape

A.shape
```

[77]: (3, 3)

[78]:
```
# Show the numpy array size

A.size
```

[78]: 9

[79]:
```
# Access the element on the second row and third column

A[1, 2]
```

[79]: 23

[80]:
```
# Access the element on the second row and third column

A[1][2]
```

[80]: 23

[81]:
```
# Access the element on the first row and first column

A[0][0]
```

[81]: 11

[82]:
```
# Access the element on the first row and first and second columns

A[0][0:2]
```

[82]: array([11, 12])

[83]:
```
# Access the element on the first and second rows and third column

A[0:2, 2]
```

[83]: array([13, 23])

[84]:
```
# Create a numpy array X

X = np.array([[1, 0], [0, 1]])
```

```
X
```

[84]:
```
array([[1, 0],
       [0, 1]])
```

[85]:
```
# Create a numpy array Y

Y = np.array([[2, 1], [1, 2]])
Y
```

[85]:
```
array([[2, 1],
       [1, 2]])
```

[86]:
```
# Add X and Y

Z = X + Y
Z
```

[86]:
```
array([[3, 1],
       [1, 3]])
```

[87]:
```
# Create a numpy array Y

Y = np.array([[2, 1], [1, 2]])
Y
```

[87]:
```
array([[2, 1],
       [1, 2]])
```

[88]:
```
# Multiply Y with 2

Z = 2 * Y
Z
```

[88]:
```
array([[4, 2],
       [2, 4]])
```

[89]:
```
# Create a numpy array Y

Y = np.array([[2, 1], [1, 2]])
Y
```

[89]:
```
array([[2, 1],
       [1, 2]])
```

[90]:
```
# Create a numpy array X
```

```
X = np.array([[1, 0], [0, 1]])
X
```

[90]:
```
array([[1, 0],
       [0, 1]])
```

[91]:
```
# Multiply X with Y

Z = X * Y
Z
```

[91]:
```
array([[2, 0],
       [0, 2]])
```

[92]:
```
# Create a matrix A

A = np.array([[0, 1, 1], [1, 0, 1]])
A
```

[92]:
```
array([[0, 1, 1],
       [1, 0, 1]])
```

[93]:
```
# Create a matrix B

B = np.array([[1, 1], [1, 1], [-1, 1]])
B
```

[93]:
```
array([[ 1,  1],
       [ 1,  1],
       [-1,  1]])
```

[94]:
```
# Calculate the dot product

Z = np.dot(A,B)
Z
```

[94]:
```
array([[0, 2],
       [0, 2]])
```

[95]:
```
# Calculate the sine of Z

np.sin(Z)
```

[95]:
```
array([[0.        , 0.90929743],
       [0.        , 0.90929743]])
```

[96]:
```
# Calculate the sine of Z

np.sin(Z)
```

[96]:
```
array([[0.        , 0.90929743],
       [0.        , 0.90929743]])
```

[98]:
```
# Create a matrix C

C = np.array([[1,1],[2,2],[3,3]])
C
```

[98]:
```
array([[1, 1],
       [2, 2],
       [3, 3]])
```

[99]:
```
#Get the transposed of C

C.T
```

[99]:
```
array([[1, 2, 3],
       [1, 2, 3]])
```

[102]:
```
#Convert to a numpy array


a = [[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12]]
x = np.array(a)
x
```

[102]:
```
array([[ 1,  2,  3,  4],
       [ 5,  6,  7,  8],
       [ 9, 10, 11, 12]])
```

[103]:
```
#Calculate the numpy array size

x.size
```

[103]:
```
12
```

[106]:
```
#Access the element on the first row and first and second columns.

x[0][0:2]
```

[106]:
```
array([1, 2])
```

[111]:
```
#Perform matrix multiplication with the numpy arrays A and B.

B = np.array([[0, 1], [1, 0], [1, 1], [-1, 0]])
A = np.dot(x, B)
A
```

[111]:
```
array([[ 1,  4],
       [ 5, 12],
       [ 9, 20]])
```

[114]:
```
import requests
```

[115]:
```
import os
from PIL import Image
from IPython.display inport IFrame
```

```
  File "/tmp/ipykernel_736/3209618625.py", line 3
    from IPython.display inport IFrame
                         ^
SyntaxError: invalid syntax
```

[116]:
```
url='https://www.ibm.com/'
r=requests.get(url)
```

[123]:
```
#View the status code using the attribute status.code

r.status_code
```

[123]:
```
200
```

[122]:
```
#You can view the request headers

print(r.request.headers)
```

```
{'User-Agent': 'python-requests/2.26.0', 'Accept-Encoding': 'gzip, deflate, br',
'Accept': '*/*', 'Connection': 'keep-alive', 'Cookie': '_abck=7DB360D24EC796B4C9
055DE48079AE9A~-1~YAAQJ+8uF5t2C2Z/AQAAn/U5agd9x/B35ssgOiSACioDw519ZRMdfkAg4NTQ2C
UYTUajY6EreYia3cYKkR4b5bvMFagzZAL2syYEFS2d/5/77zUbc6HK0kGvTyuLBAjakp3flSQpS17pDT
d+eMIl8rEWWBktgWJDiVWvUIg7tpEZLG8OMVj6uNaewelyFqolS5itMsaAGu4QdI+kPCmrqm6Gfx6Lnc
G9Px+noEiBx+U/OvXPLg1Z1YTvmNkR8JXfjKv/dEHcACAAk8Jw1CE9+oojvZzn/vC/sN+uxVZ+QhLHj3
fVRUifZC+yxjq37Id+4CjSGVyNrE1pwNgoMVhwGDXd3MpcrO4FX1QdUIup3Me9LtQNxeI=~-1~-1~-1;
bm_sz=4D42C4927D00F7E9404989907045E355~YAAQJ+8uF5x2C2Z/AQAAn/U5ag8Bgs6qLPJsZxr3D
vfdYAbSgdqJn04Up0bIIopuo57zafZP6O8SX0mWHesKyLy6in9mQs+CqaZ+tpfTglUuWYYcTXEhxpAq8
Mv8rn+a4idlCzFBRs05t5k6DhAU2LBM1LgUxBmkpceqAIHjs1Mk18hwWmAAc+eGUhX3hPobewiGILTOl
mE6OxKji9wljKThSCnpmbMhBfvuG+vzVKqAnz6p+Zg4AeQWjaodEYxNqIH32dRcT70kO/eq7GFyIF0Wz
w9MKam27Zysat6Pzkg=~3290417~3289923'}
```

[121]:
```
#You can view the request body, in the following line, as there is no body for␣
 ↪a get request we get a None:
print("request body:", r.request.body)
```

```
request body: None
```

[124]:
```
#You can view the HTTP response header using the attribute headers. This␣
 ↪returns a python dictionary of HTTP response headers.

header = r.headers
print(r.headers)
```

```
{'Cache-Control': 'max-age=301', 'Expires': 'Mon, 07 Mar 2022 22:56:02 GMT',
'Last-Modified': 'Mon, 07 Mar 2022 22:19:18 GMT', 'ETag':
'"198fb-5d9a8416c293f"', 'Accept-Ranges': 'bytes', 'Content-Encoding': 'gzip',
'Content-Type': 'text/html', 'X-Akamai-Transformed': '9 21156 0 pmb=mTOE,1',
'Date': 'Tue, 08 Mar 2022 15:50:57 GMT', 'Content-Length': '21224',
'Connection': 'keep-alive', 'Vary': 'Accept-Encoding', 'x-content-type-options':
'nosniff', 'X-XSS-Protection': '1; mode=block', 'Content-Security-Policy':
'upgrade-insecure-requests', 'Strict-Transport-Security': 'max-age=31536000'}
```

[125]:
```
#We can obtain the date the request was sent using the key Date

header['date']
```

[125]:
```
'Tue, 08 Mar 2022 15:50:57 GMT'
```

[126]:
```
#Content-Type indicates the type of data:

header['Content-Type']
```

[126]:
```
'text/html'
```

[127]:
```
#You can also check the encoding:

r.encoding
```

[127]:
```
'ISO-8859-1'
```

[130]:
```
#As the Content-Type is text/html we can use the attribute text to display the␣
 ↪HTML in the body. We can review the first 100 characters:

r.text[0:100]
```

[130]:
```
'<!DOCTYPE html><html lang="en-US"><head><meta name="viewport"
content="width=device-width"/><meta ch'
```

[131]:
```
#You can load other types of data for non-text requests, like images. Consider␣
 ↪the URL of the following image:

r.requests.get(url)
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
/tmp/ipykernel_736/2824813893.py in <module>
      1 #You can load other types of data for non-text requests, like images.␣
 ↪Consider the URL of the following image:
      2
----> 3 r.requests.get(url)

AttributeError: 'Response' object has no attribute 'requests'
```

[132]:
```
#We can look at the response header
print(r.headers)
```

```
{'Cache-Control': 'max-age=301', 'Expires': 'Mon, 07 Mar 2022 22:56:02 GMT',
'Last-Modified': 'Mon, 07 Mar 2022 22:19:18 GMT', 'ETag':
'"198fb-5d9a8416c293f"', 'Accept-Ranges': 'bytes', 'Content-Encoding': 'gzip',
'Content-Type': 'text/html', 'X-Akamai-Transformed': '9 21156 0 pmb=mTOE,1',
'Date': 'Tue, 08 Mar 2022 15:50:57 GMT', 'Content-Length': '21224',
'Connection': 'keep-alive', 'Vary': 'Accept-Encoding', 'x-content-type-options':
'nosniff', 'X-XSS-Protection': '1; mode=block', 'Content-Security-Policy':
'upgrade-insecure-requests', 'Strict-Transport-Security': 'max-age=31536000'}
```

[133]:
```
#We can see the Content-Type
r.headers['Content-Type']
```

[133]:
```
'text/html'
```

[135]:
```
An image is a response object that contains the image as a bytes-like object.␣
 ↪As a result, we must save it using a file object. First, we specify the file␣
 ↪path and name

path=os.path.join(os.getcwd(),'image.png')
path
```

```
  File "/tmp/ipykernel_736/3194079113.py", line 1
    An image is a response object that contains the image as a bytes-like objec
 ↪ As a result, we must save it using a file object. First, we specify the file
 ↪path and name
            ^

SyntaxError: invalid syntax
```

[ ]:
```
#We save the file, in order to access the body of the response we use the␣
 ↪attribute content then save it using the open function and write method:

with open(path,'wb') as f:
    f.write(r.content)
```

[9]:
```
Name = "Michael Jackson"
Name.find('el')
```

[9]:
```
5
```

[10]:
```
A = '1'
B = '2'
A + B
```

[10]:
```
'12'
```

[13]:
```
A=((11,12),[21,22])
A[0][1]
```

[13]:
```
12
```

[21]:
```
'1,2,3,4'.split(',')
```

[21]:
```
['1', '2', '3', '4']
```

[22]:
```
class Points(object):

    def __init__(self,x,y):

        self.x=x

        self.y=y

    def print_point(self):

        print('x=',self.x,' y=',self.y)

        p2=Points(1,2)

        p2.x=2

        p2.print_point()
```

[25]:
```
V={'A','B','C' }
V.add('C')
```

[27]:
```
for n in range(3):

    print(n)
```

```
0
1
2
```

[29]:
```
A=['1','2','3']

for a in A:

    print(2*a)
```

```
11
22
33
```

[34]:
```
Add('1''1')
def Add(x,y):

    z=y+x

    return(y)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
/tmp/ipykernel_1392/2698768755.py in <module>
----> 1 Add('1''1')
      2 def Add(x,y):
      3
      4     z=y+x
      5

TypeError: Add() missing 1 required positional argument: 'y'
```

[35]:
```
class Points(object):

    def __init__(self,x,y):

        self.x=x

        self.y=y

    def print_point(self):

        print('x=',self.x,'y=',self.y)
```

[38]:
```
class Points(object):

    def __init__(self,x,y):

        self.x=x

        self.y=y

    def print_point(self):

        print('x=',self.x,' y=',self.y)

        p1=Points(1,2)

        p1.print_point()
```

[ ]:

[ ]: