Data Input/ Output

NEE2106 Computer Programming for Electrical Engineers

Prepared by: Dr. Rui Li (rui.li@vu.edu.au)

© 2024 Victoria University. CRICOS No. 00124K (Melbourne), 02475D (Sydney and Brisbane), RTO 3113, TEQSA No. PRV12152, Provider Category: Australian University

< Contents >

- Access strings and lists
- User defined functions
- Read/ Write to a file

VICTORIA UNIVERSITY
MELBOURNE AUSTRALIA

© 2024 Victoria University. CRICOS No. 00124K (Melbourne), 02475D (Sydney and Brisbane), RTO 3113, TEQSA No. PRV12152, Provider Category: Australian University

< List >

A list is written as a sequence of data values separated by commas:

```
[1951, 1969, 1984]                    # A list of integers

['apples', 'oranges', 'cherries']     # A list of strings
```

Build lists of integers using the range and list functions:

```
>>> first = [1, 2, 3, 4]
>>> second = list(range(1, 5))
>>> first
[1, 2, 3, 4]
>>> second
[1, 2, 3, 4]
>>>
```

Equality (==) :

```
>>> first == second
True
>>>
```

Concatenation (list + list):

```
>>> first + [5, 6]
[1, 2, 3, 4, 5, 6]
```

in operator to detect the presence or absence of element:

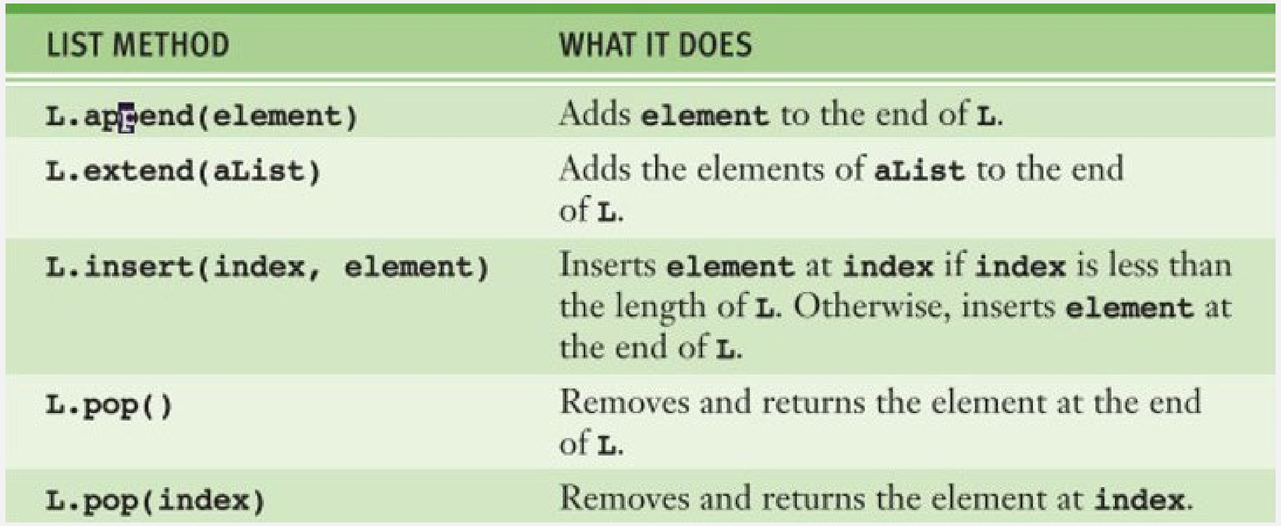
```
>>> 3 in [1, 2, 3]
True
>>> 0 in [1, 2, 3]
False
```

© 2024 Victoria University. CRICOS No. 00124K (Melbourne), 02475D (Sydney and Brisbane), RTO 3113, TEQSA No. PRV12152, Provider Category: Australian University

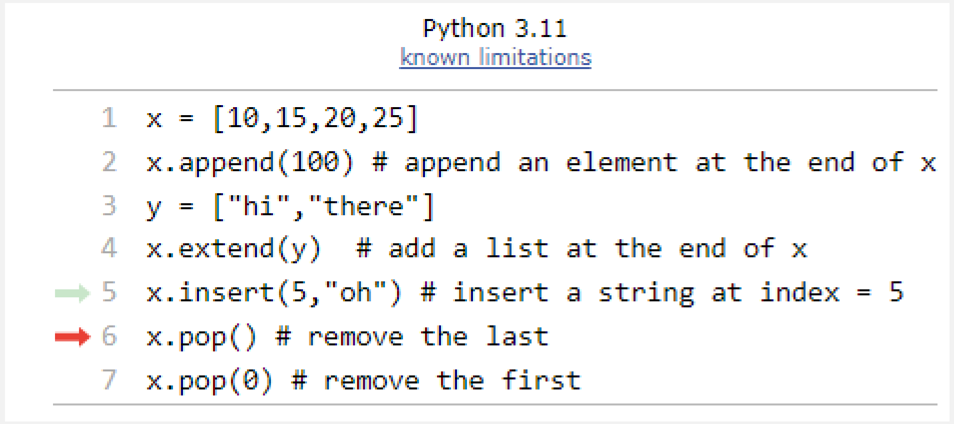
< List >

Replace an element at given position:

```
>>> example = [1, 2, 3, 4]
>>> example
[1, 2, 3, 4]
>>> example[3] = 0
>>> example
[1, 2, 3, 0]
```

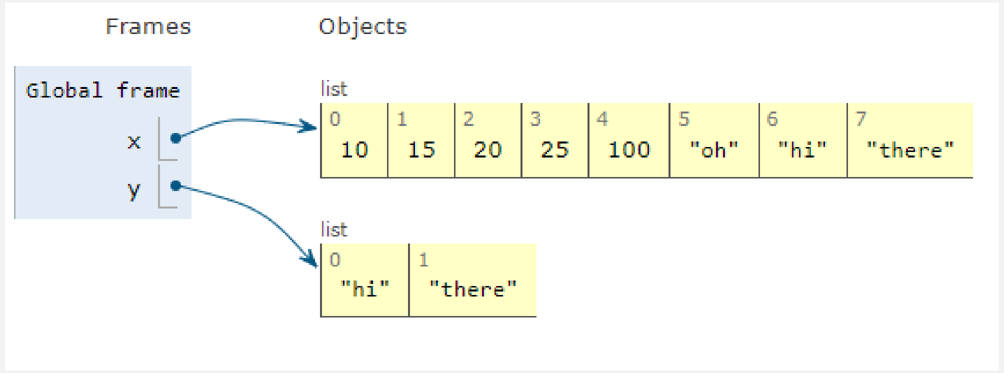


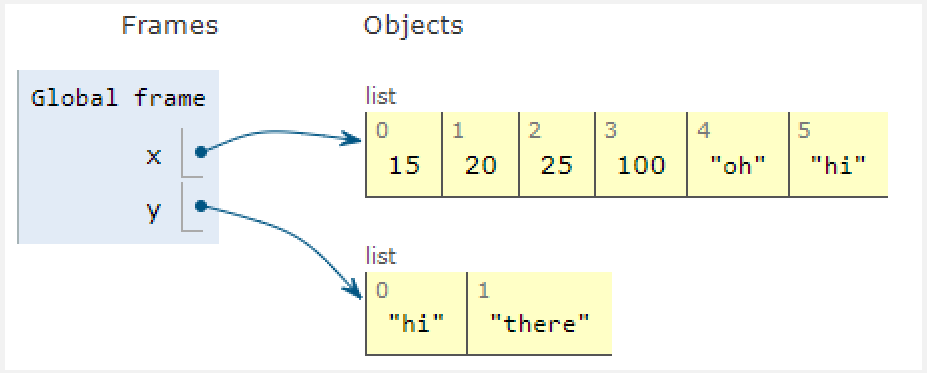
LIST METHOD	WHAT IT DOES
L.append(element)	Adds **element** to the end of **L**.
L.extend(aList)	Adds the elements of **aList** to the end of **L**.
L.insert(index, element)	Inserts **element** at **index** if **index** is less than the length of **L**. Otherwise, inserts **element** at the end of **L**.
L.pop()	Removes and returns the element at the end of **L**.
L.pop(index)	Removes and returns the element at **index**.



Python 3.11
known limitations

```
x = [10,15,20,25]
x.append(100) # append an element at the end of x
y = ["hi","there"]
x.extend(y)  # add a list at the end of x
x.insert(5,"oh") # insert a string at index = 5
x.pop() # remove the last
x.pop(0) # remove the first
```



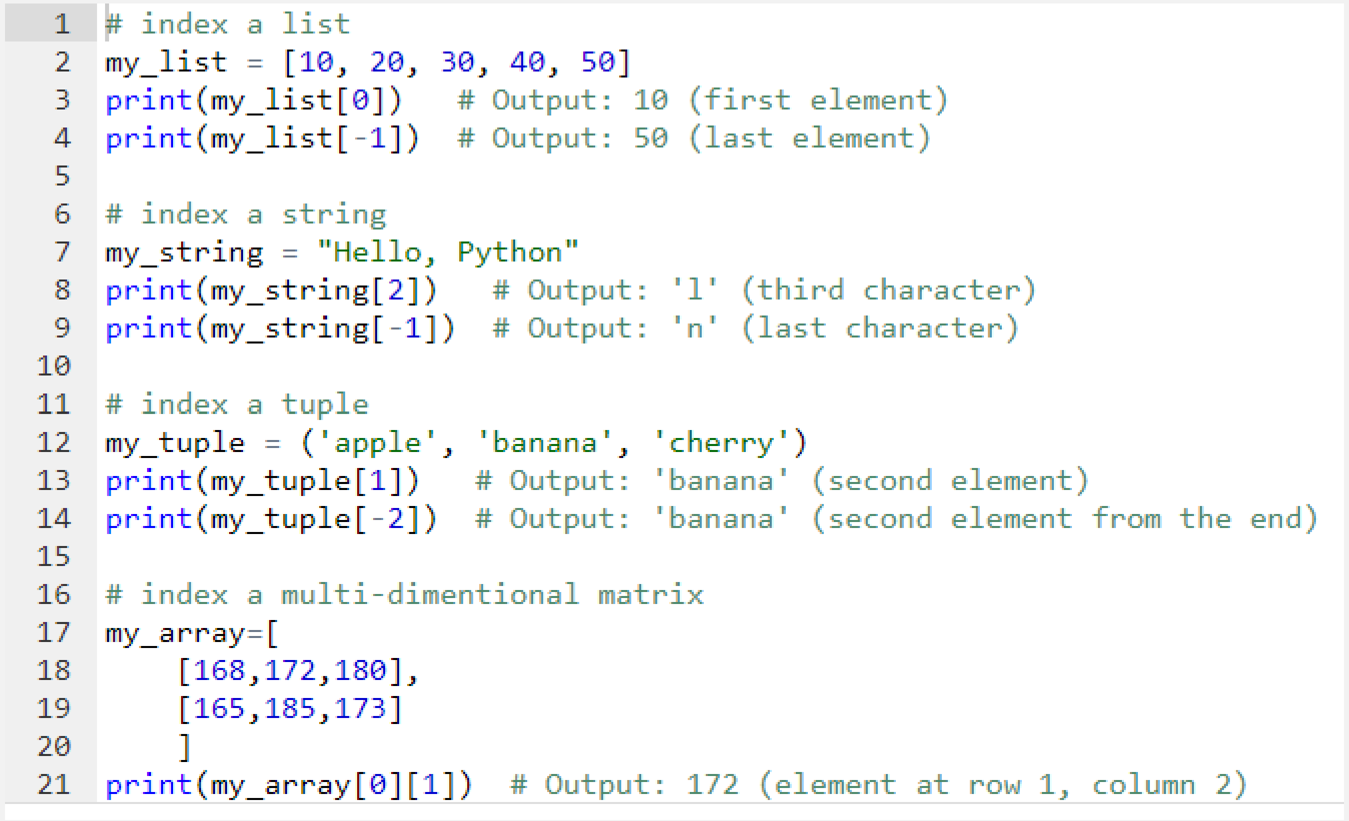


© 2024 Victoria University. CRICOS No. 00124K (Melbourne), 02475D (Sydney and Brisbane), RTO 3113, TEQSA No. PRV12152, Provider Category: Australian University

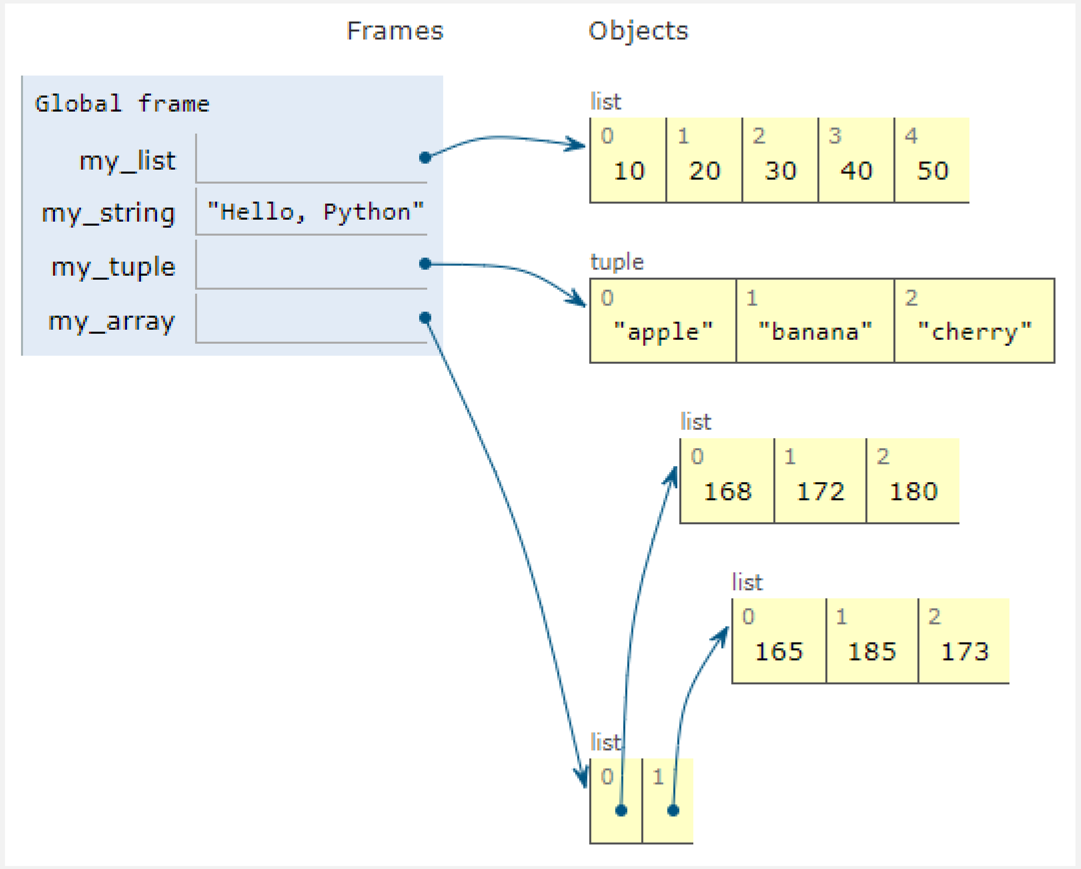
< Index >

The indexing notation [ ] is commonly used to specify position of element(s).

- The 1st element is indexed at 0, i.e. **x[0]**
- Positive indices count from the beginning (starting with 0)
- Negative indices count from the end (starting with -1), i.e. **x[-1]** refers to the last element
- Multi-dimensional array is referred as **x[row][column]**



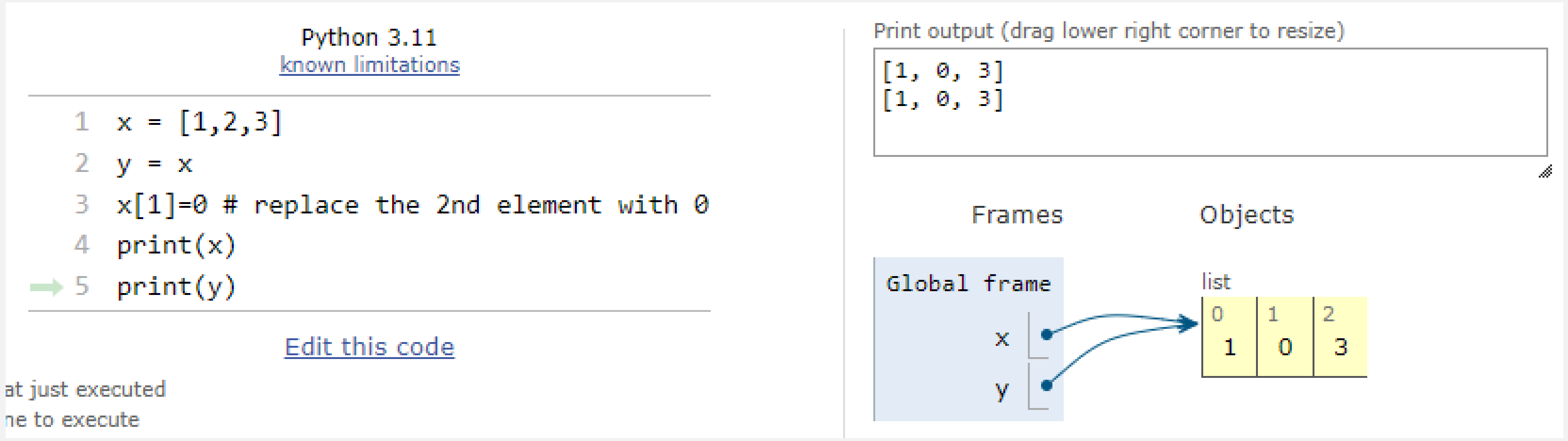
```python
# index a list
my_list = [10, 20, 30, 40, 50]
print(my_list[0])   # Output: 10 (first element)
print(my_list[-1])  # Output: 50 (last element)

# index a string
my_string = "Hello, Python"
print(my_string[2])   # Output: 'l' (third character)
print(my_string[-1])  # Output: 'n' (last character)

# index a tuple
my_tuple = ('apple', 'banana', 'cherry')
print(my_tuple[1])   # Output: 'banana' (second element)
print(my_tuple[-2])  # Output: 'banana' (second element from the end)

# index a multi-dimentional matrix
my_array=[
    [168,172,180],
    [165,185,173]
    ]
print(my_array[0][1])  # Output: 172 (element at row 1, column 2)
```



© 2024 Victoria University. CRICOS No. 00124K (Melbourne), 02475D (Sydney and Brisbane), RTO 3113, TEQSA No. PRV12152, Provider Category: Australian University

< Side effect/ Aliasing >

Do you notice anything interesting/ strange here?



```
x = [1,2,3]
y = x
x[1]=0 # replace the 2nd element with 0
print(x)
print(y)
```

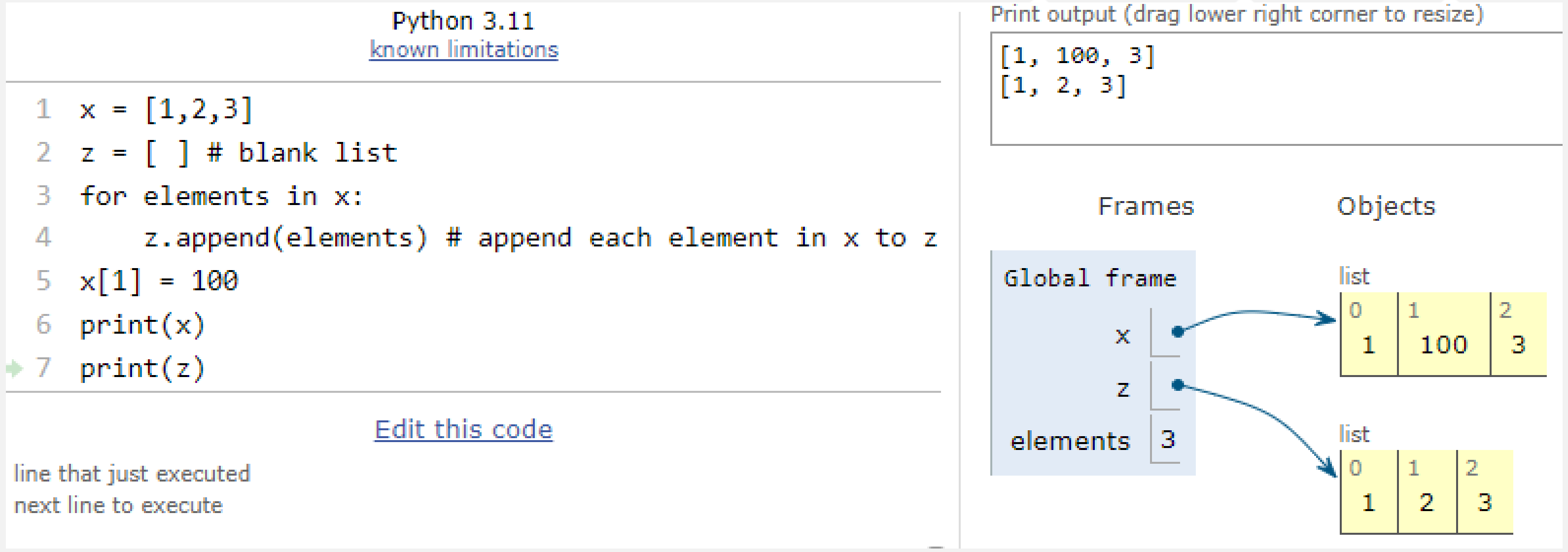
```
[1, 0, 3]
[1, 0, 3]
```

Side effect: after replacing an element in x, list y is also changed!

Variables **x** and **y** refer to the exact same list object

To prevent: create a new object and copy the contents of the original to it.

Variables **x** and **y** refer to different list objects.

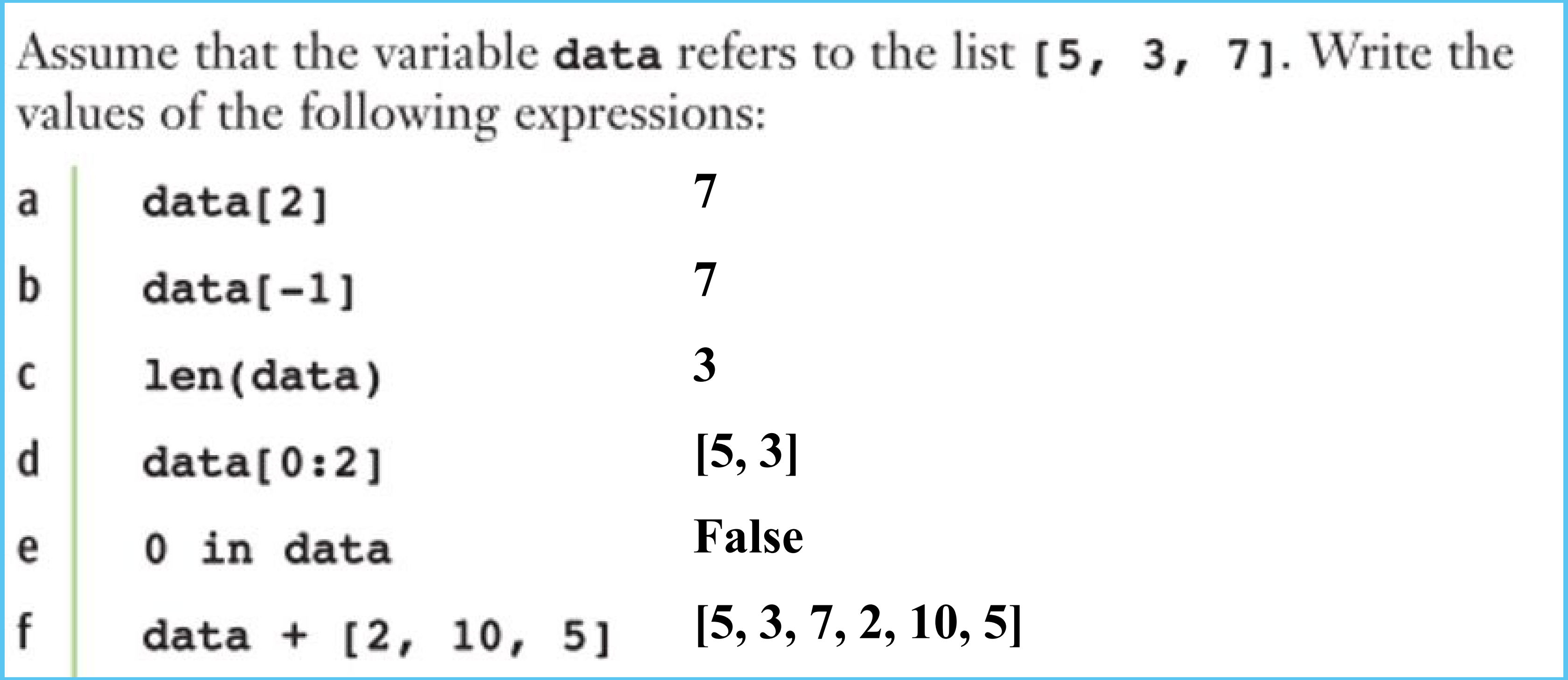


```
x = [1,2,3]
z = [ ] # blank list
for elements in x:
    z.append(elements) # append each element in x to z
x[1] = 100
print(x)
print(z)
```

```
[1, 100, 3]
[1, 2, 3]
```

© 2024 Victoria University. CRICOS No. 00124K (Melbourne), 02475D (Sydney and Brisbane), RTO 3113, TEQSA No. PRV12152, Provider Category: Australian University

< Exercise>



Assume that the variable **data** refers to the list [5, 3, 7]. Write the values of the following expressions:

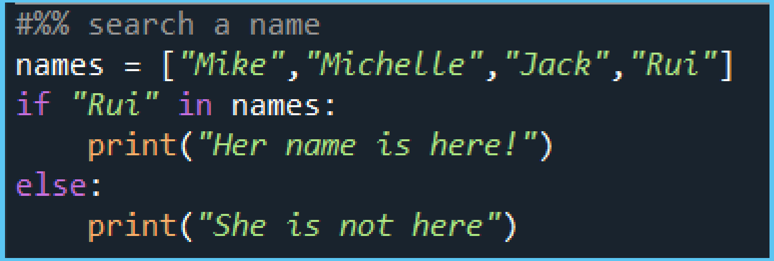
a	`data[2]`	7
b	`data[-1]`	7
c	`len(data)`	3
d	`data[0:2]`	[5, 3]
e	`0 in data`	False
f	`data + [2, 10, 5]`	[5, 3, 7, 2, 10, 5]

VICTORIA UNIVERSITY MELBOURNE AUSTRALIA

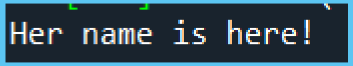
© 2024 Victoria University. CRICOS No. 00124K (Melbourne), 02475D (Sydney and Brisbane), RTO 3113, TEQSA No. PRV12152, Provider Category: Australian University

< Exercise>

Q1: Write a program to search if Rui’s name is in the list of “Mike”, “Michelle”, “Jack”, “Rui”.

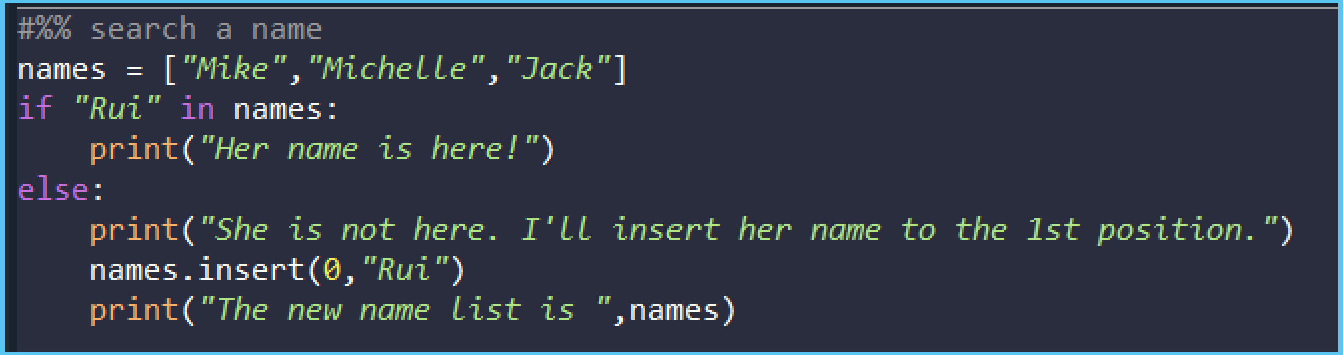


```python
#%% search a name
names = ["Mike","Michelle","Jack","Rui"]
if "Rui" in names:
    print("Her name is here!")
else:
    print("She is not here")
```

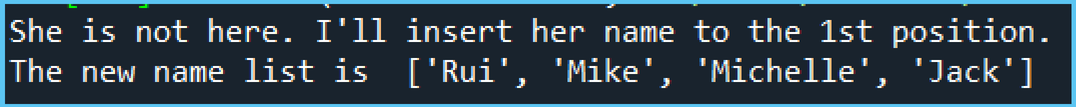


```
Her name is here!
```

Q2: If Rui’s name is not there, insert her name to the first position.



```python
#%% search a name
names = ["Mike","Michelle","Jack"]
if "Rui" in names:
    print("Her name is here!")
else:
    print("She is not here. I'll insert her name to the 1st position.")
    names.insert(0,"Rui")
    print("The new name list is ",names)
```



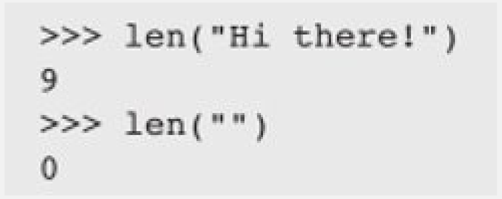
```
She is not here. I'll insert her name to the 1st position.
The new name list is  ['Rui', 'Mike', 'Michelle', 'Jack']
```

© 2024 Victoria University. CRICOS No. 00124K (Melbourne), 02475D (Sydney and Brisbane), RTO 3113, TEQSA No. PRV12152, Provider Category: Australian University

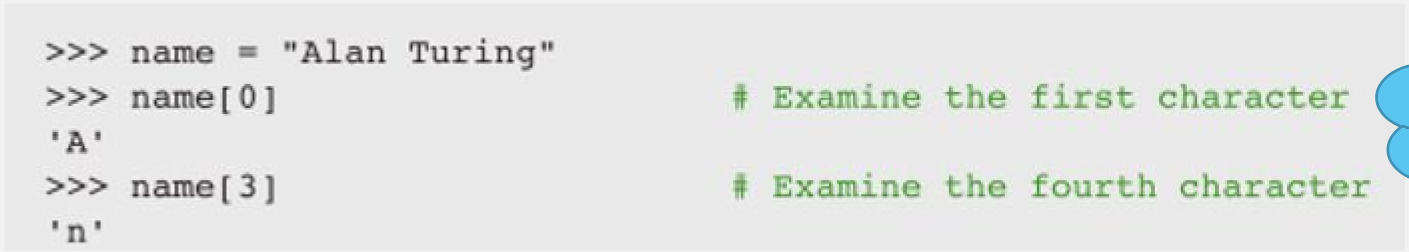
< String >

A string is a data structure that consists of several smaller pieces of data.

len( ) function returns the length of a string:
max() returns the largest value:
min() returns the smallest value:

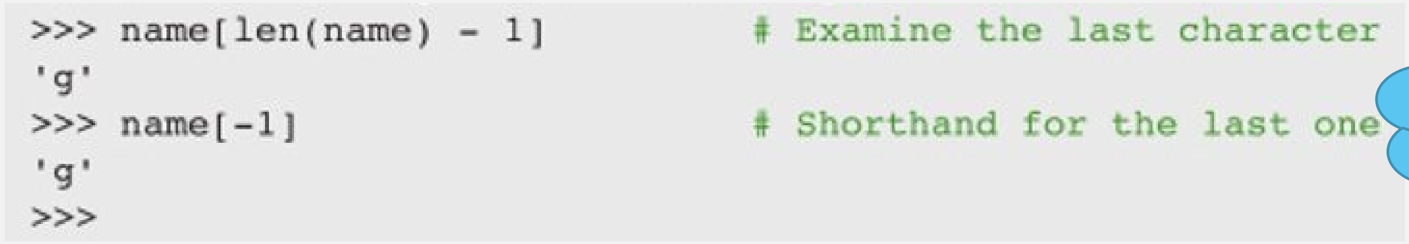

```
>>> len("Hi there!")
9
>>> len("")
0
```

Access string elements:


```
>>> name = "Alan Turing"
>>> name[0]                    # Examine the first character
'A'
>>> name[3]                    # Examine the fourth character
'n'
```

The 1st element has an index of 0

Access the last element:


```
>>> name[len(name) - 1]        # Examine the last character
'g'
>>> name[-1]                   # Shorthand for the last one
'g'
>>>
```

var(-1) refers to the last element

Q: how to refer to the second last element of a variable “name”?

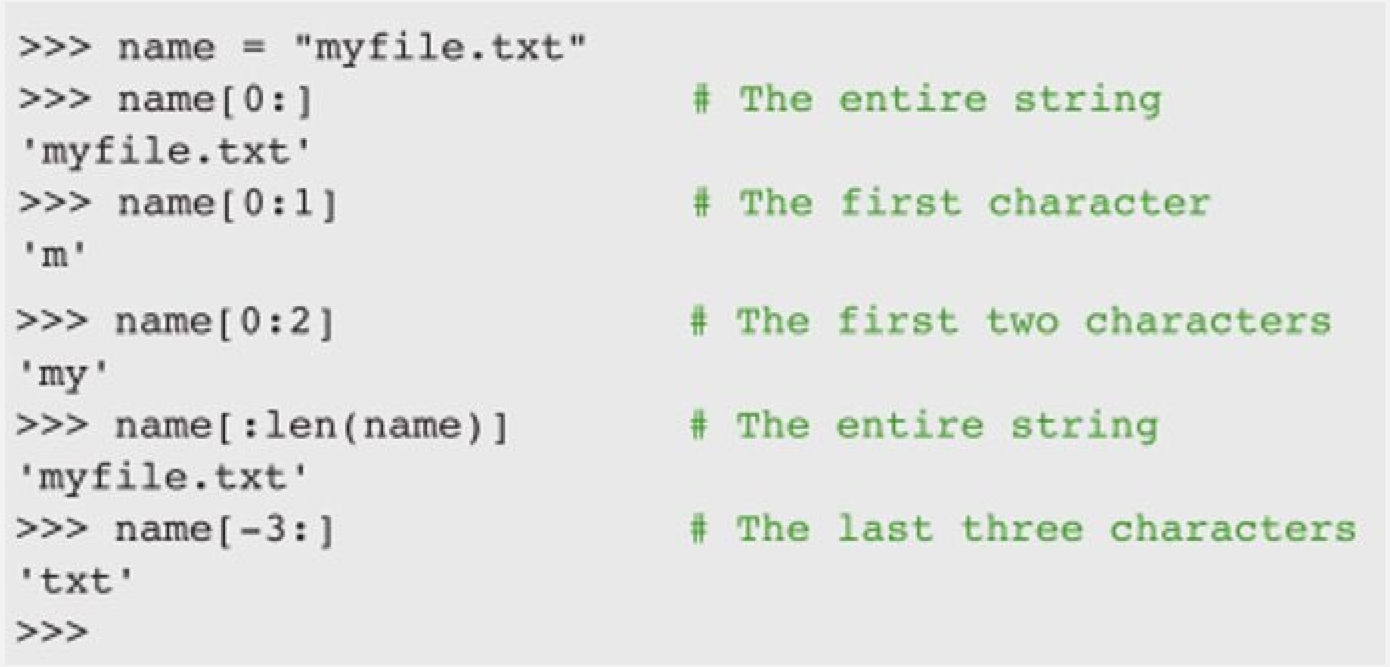
A: name(-2)

© 2024 Victoria University. CRICOS No. 00124K (Melbourne), 02475D (Sydney and Brisbane), RTO 3113, TEQSA No. PRV12152, Provider Category: Australian University

< Slicing >

Python uses var[start:end:step] to extract a slice/ portion of elements from a sequence

- **var**: the variable you from which you want to extract a slice
- **start**: the starting index. If not given, the slice starts from *index=0*.
- **end**: the ending index (up to but NOT include). If not given, the slice extends to the *end* of the sequence.
- **step**: the interval at which to slice. If not given, default is 1

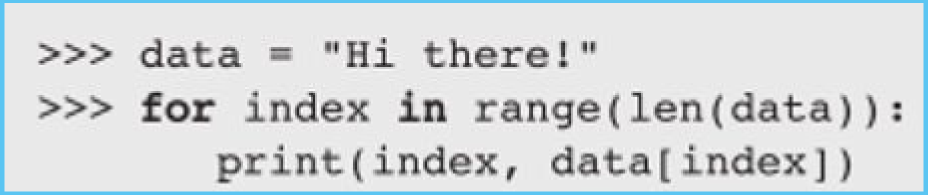


```
>>> name = "myfile.txt"
>>> name[0:]                  # The entire string
'myfile.txt'
>>> name[0:1]                 # The first character
'm'
>>> name[0:2]                 # The first two characters
'my'
>>> name[:len(name)]          # The entire string
'myfile.txt'
>>> name[-3:]                 # The last three characters
'txt'
>>>
```

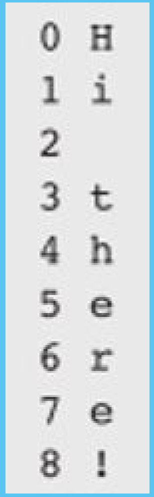
© 2024 Victoria University. CRICOS No. 00124K (Melbourne), 02475D (Sydney and Brisbane), RTO 3113, TEQSA No. PRV12152, Provider Category: Australian University

< Example>

Q1: What is the output of the following program?

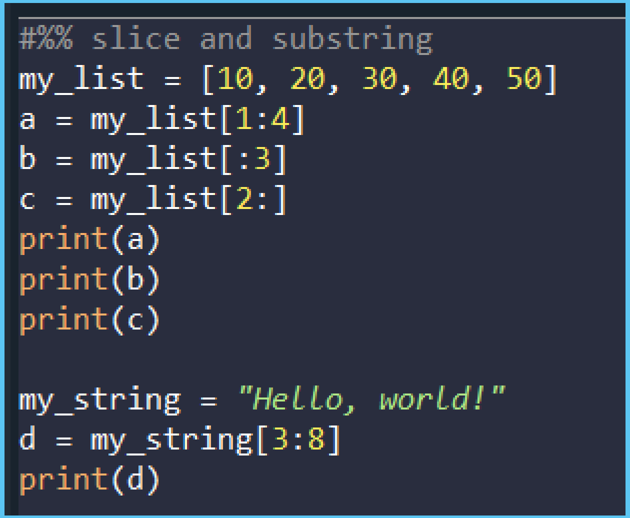


```
>>> data = "Hi there!"
>>> for index in range(len(data)):
        print(index, data[index])
```

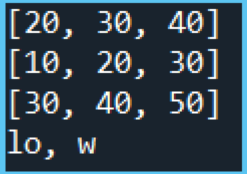


```
0 H
1 i
2
3 t
4 h
5 e
6 r
7 e
8 !
```

Q2: What is the output of the following program?



```
#%% slice and substring
my_list = [10, 20, 30, 40, 50]
a = my_list[1:4]
b = my_list[:3]
c = my_list[2:]
print(a)
print(b)
print(c)

my_string = "Hello, world!"
d = my_string[3:8]
print(d)
```

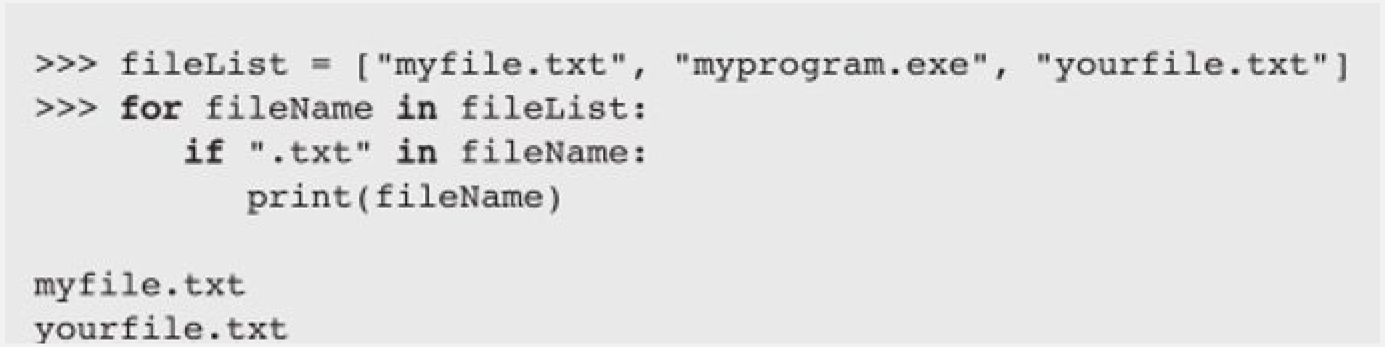


```
[20, 30, 40]
[10, 20, 30]
[30, 40, 50]
lo, w
```

© 2024 Victoria University. CRICOS No. 00124K (Melbourne), 02475D (Sydney and Brisbane), RTO 3113, TEQSA No. PRV12152, Provider Category: Australian University

< String methods>

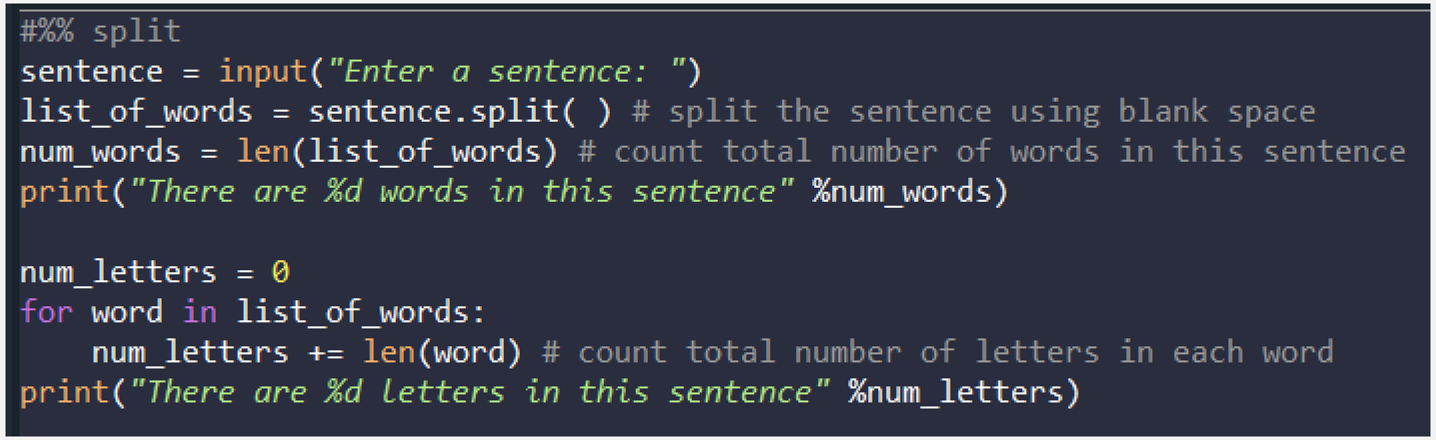
Testing for a Substring with the **in** Operator:



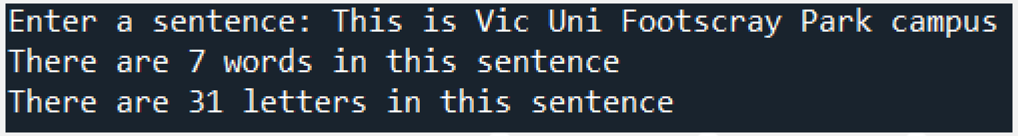
```
>>> fileList = ["myfile.txt", "myprogram.exe", "yourfile.txt"]
>>> for fileName in fileList:
        if ".txt" in fileName:
            print(fileName)

myfile.txt
yourfile.txt
```

String method **split** to obtain a list of the words contained in an input string.

By default, split on any whitespace (spaces, tabs, newlines). It can also be defined using split('separator'), i.e. split(',')



```
#%% split
sentence = input("Enter a sentence: ")
list_of_words = sentence.split( ) # split the sentence using blank space
num_words = len(list_of_words) # count total number of words in this sentence
print("There are %d words in this sentence" %num_words)

num_letters = 0
for word in list_of_words:
    num_letters += len(word) # count total number of letters in each word
print("There are %d letters in this sentence" %num_letters)
```

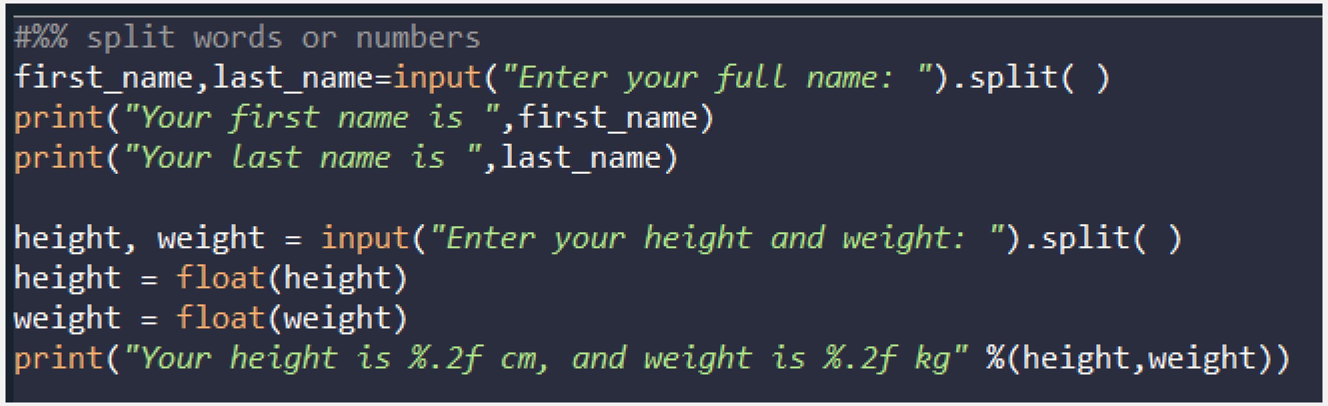


```
Enter a sentence: This is Vic Uni Footscray Park campus
There are 7 words in this sentence
There are 31 letters in this sentence
```

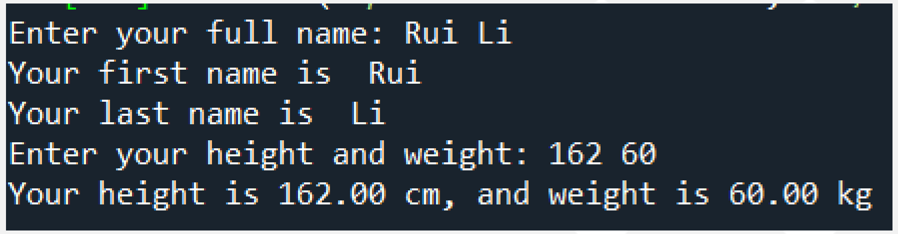
© 2024 Victoria University. CRICOS No. 00124K (Melbourne), 02475D (Sydney and Brisbane), RTO 3113, TEQSA No. PRV12152, Provider Category: Australian University

< Split( ) >

The **split( )** function can be used to obtain multiple inputs in one single line:



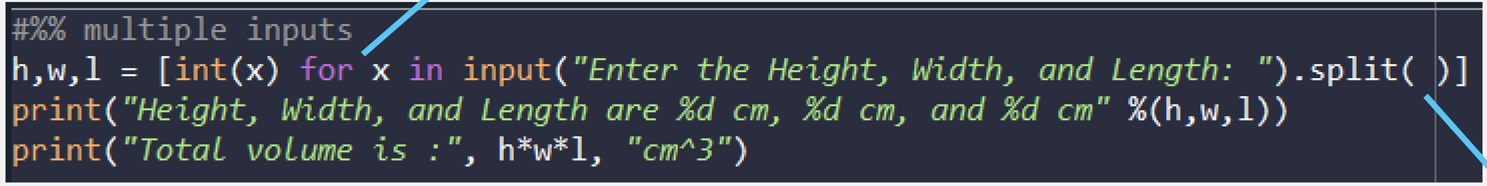
```
#%% split words or numbers
first_name,last_name=input("Enter your full name: ").split( )
print("Your first name is ",first_name)
print("Your last name is ",last_name)

height, weight = input("Enter your height and weight: ").split( )
height = float(height)
weight = float(weight)
print("Your height is %.2f cm, and weight is %.2f kg" %(height,weight))
```



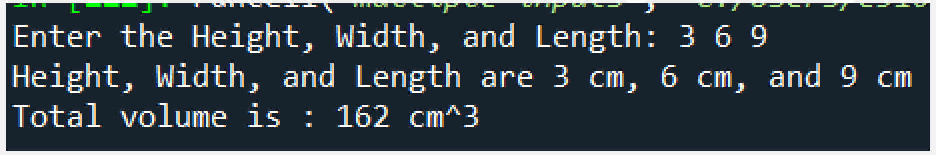
```
Enter your full name: Rui Li
Your first name is  Rui
Your last name is  Li
Enter your height and weight: 162 60
Your height is 162.00 cm, and weight is 60.00 kg
```

Consider even more input values – this may need a loop-like statement:

for loop runs through each value in x and convert it to integer type.
This is a common practice to convert a string-type list to numerical type.



```
#%% multiple inputs
h,w,l = [int(x) for x in input("Enter the Height, Width, and Length: ").split( )]
print("Height, Width, and Length are %d cm, %d cm, and %d cm" %(h,w,l))
print("Total volume is :", h*w*l, "cm^3")
```



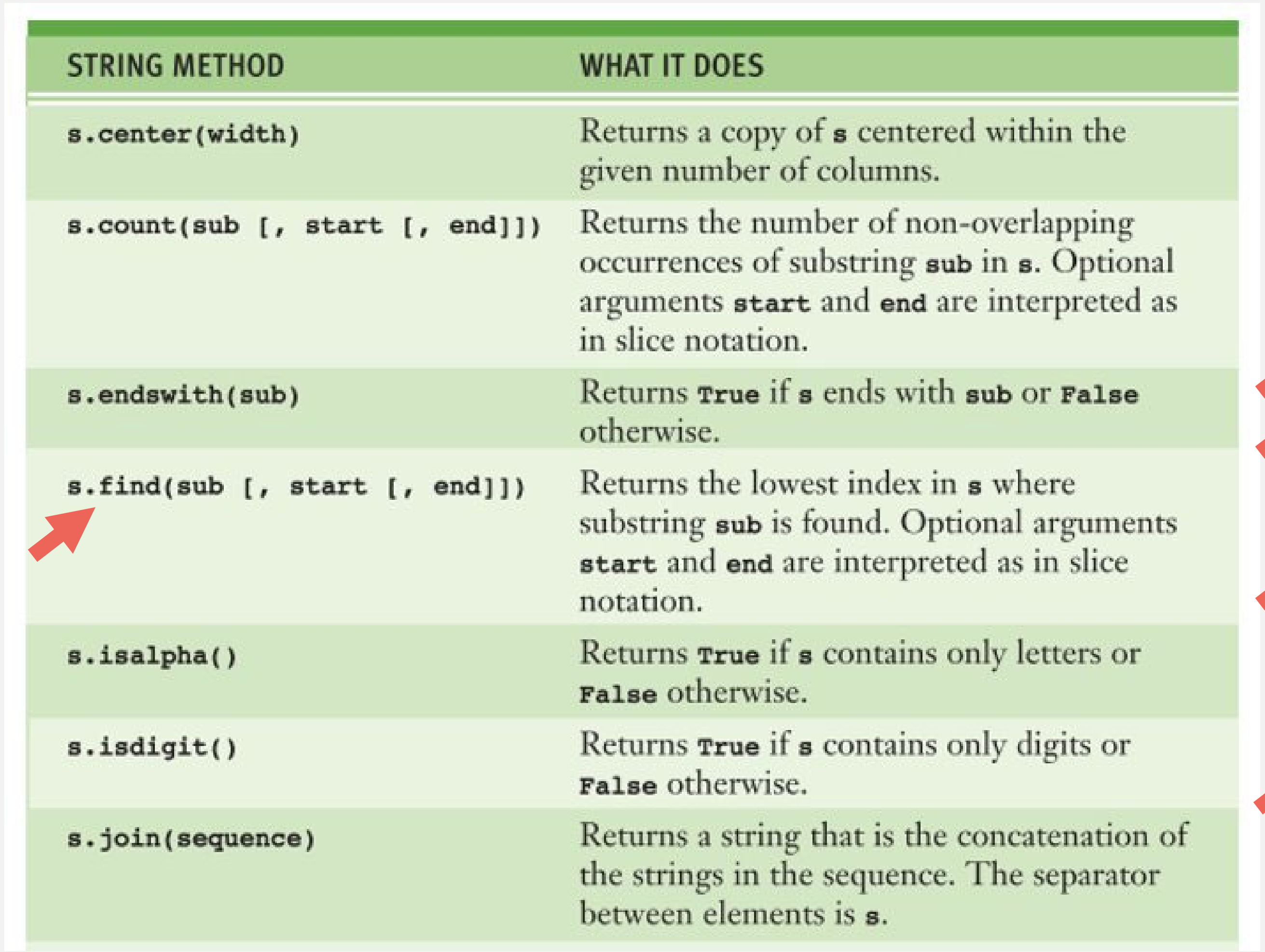
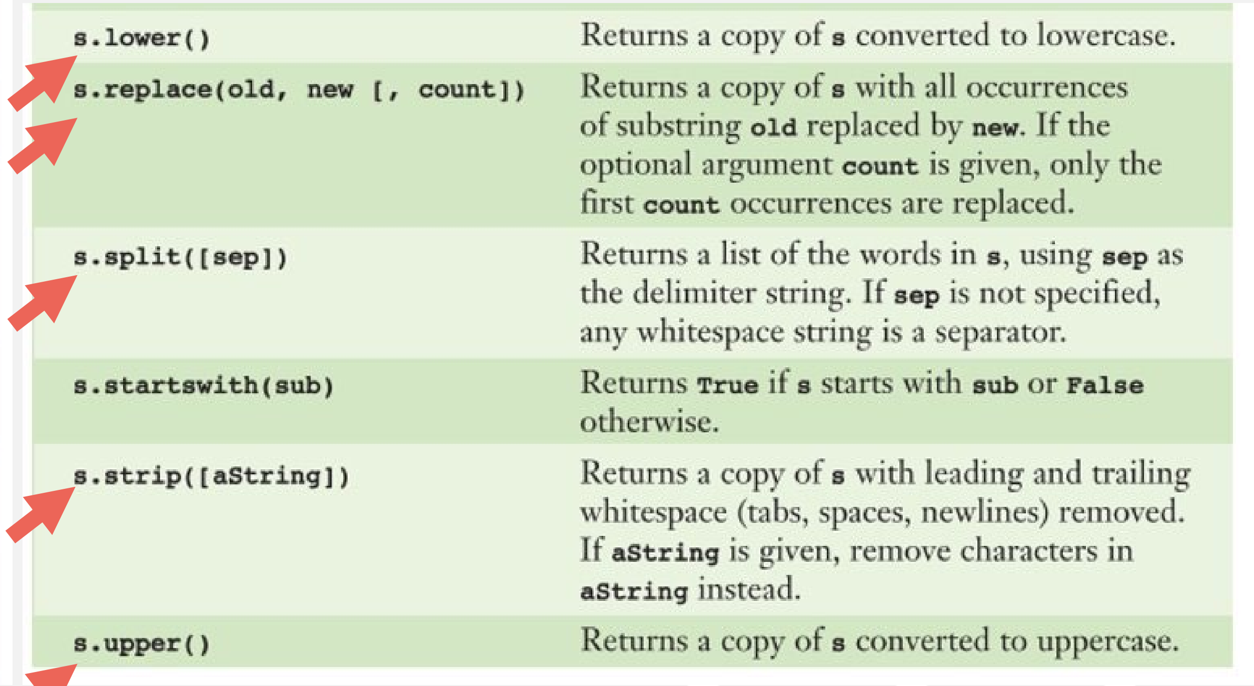
```
Enter the Height, Width, and Length: 3 6 9
Height, Width, and Length are 3 cm, 6 cm, and 9 cm
Total volume is : 162 cm^3
```

Separate values using the default whitespace, or any other arguments such as “,”, “:”, “$”, “#”…

VICTORIA UNIVERSITY
MELBOURNE AUSTRALIA

© 2024 Victoria University. CRICOS No. 00124K (Melbourne), 02475D (Sydney and Brisbane), RTO 3113, TEQSA No. PRV12152, Provider Category: Australian University

< String methods>

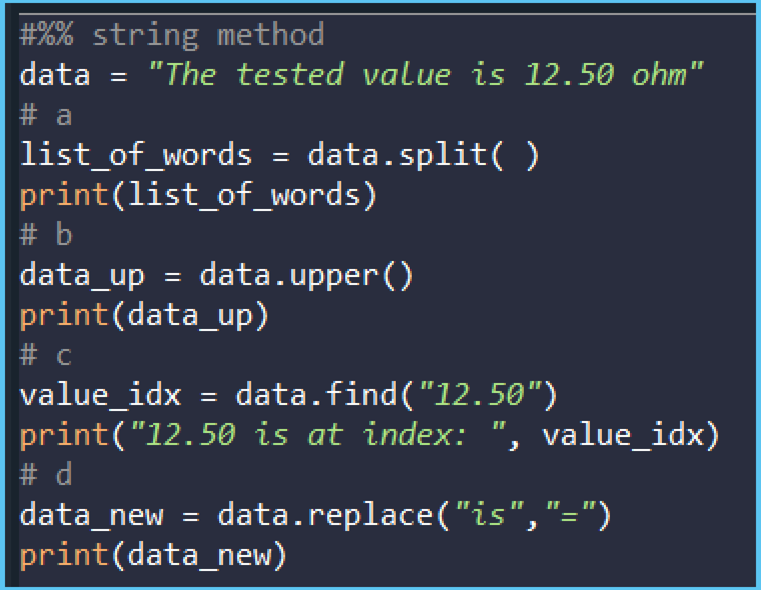
STRING METHOD	WHAT IT DOES
`s.center(width)`	Returns a copy of **s** centered within the given number of columns.
`s.count(sub [, start [, end]])`	Returns the number of non-overlapping occurrences of substring **sub** in **s**. Optional arguments **start** and **end** are interpreted as in slice notation.
`s.endswith(sub)`	Returns **True** if **s** ends with **sub** or **False** otherwise.
`s.find(sub [, start [, end]])`	Returns the lowest index in **s** where substring **sub** is found. Optional arguments **start** and **end** are interpreted as in slice notation.
`s.isalpha()`	Returns **True** if **s** contains only letters or **False** otherwise.
`s.isdigit()`	Returns **True** if **s** contains only digits or **False** otherwise.
`s.join(sequence)`	Returns a string that is the concatenation of the strings in the sequence. The separator between elements is **s**.
`s.lower()`	Returns a copy of **s** converted to lowercase.
`s.replace(old, new [, count])`	Returns a copy of **s** with all occurrences of substring **old** replaced by **new**. If the optional argument **count** is given, only the first **count** occurrences are replaced.
`s.split([sep])`	Returns a list of the words in **s**, using **sep** as the delimiter string. If **sep** is not specified, any whitespace string is a separator.
`s.startswith(sub)`	Returns **True** if **s** starts with **sub** or **False** otherwise.
`s.strip([aString])`	Returns a copy of **s** with leading and trailing whitespace (tabs, spaces, newlines) removed. If **aString** is given, remove characters in **aString** instead.
`s.upper()`	Returns a copy of **s** converted to uppercase.

© 2024 Victoria University. CRICOS No. 00124K (Melbourne), 02475D (Sydney and Brisbane), RTO 3113, TEQSA No. PRV12152, Provider Category: Australian University

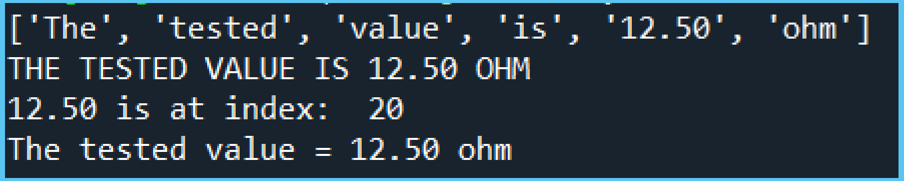
< Example>

Assume that variable **data** refers to the string “The tested value is 12.50 ohm”.
Use a string to perform the following tasks:

a. Obtain a list of the words in the string
b. Convert the string to uppercase
c. Locate the position of the value “12.50”
d. Replace the word “is” with an equal sign “=”



```
#%% string method
data = "The tested value is 12.50 ohm"
# a
list_of_words = data.split( )
print(list_of_words)
# b
data_up = data.upper()
print(data_up)
# c
value_idx = data.find("12.50")
print("12.50 is at index: ", value_idx)
# d
data_new = data.replace("is","=")
print(data_new)
```



```
['The', 'tested', 'value', 'is', '12.50', 'ohm']
THE TESTED VALUE IS 12.50 OHM
12.50 is at index:  20
The tested value = 12.50 ohm
```

© 2024 Victoria University. CRICOS No. 00124K (Melbourne), 02475D (Sydney and Brisbane), RTO 3113, TEQSA No. PRV12152, Provider Category: Australian University

< Contents >

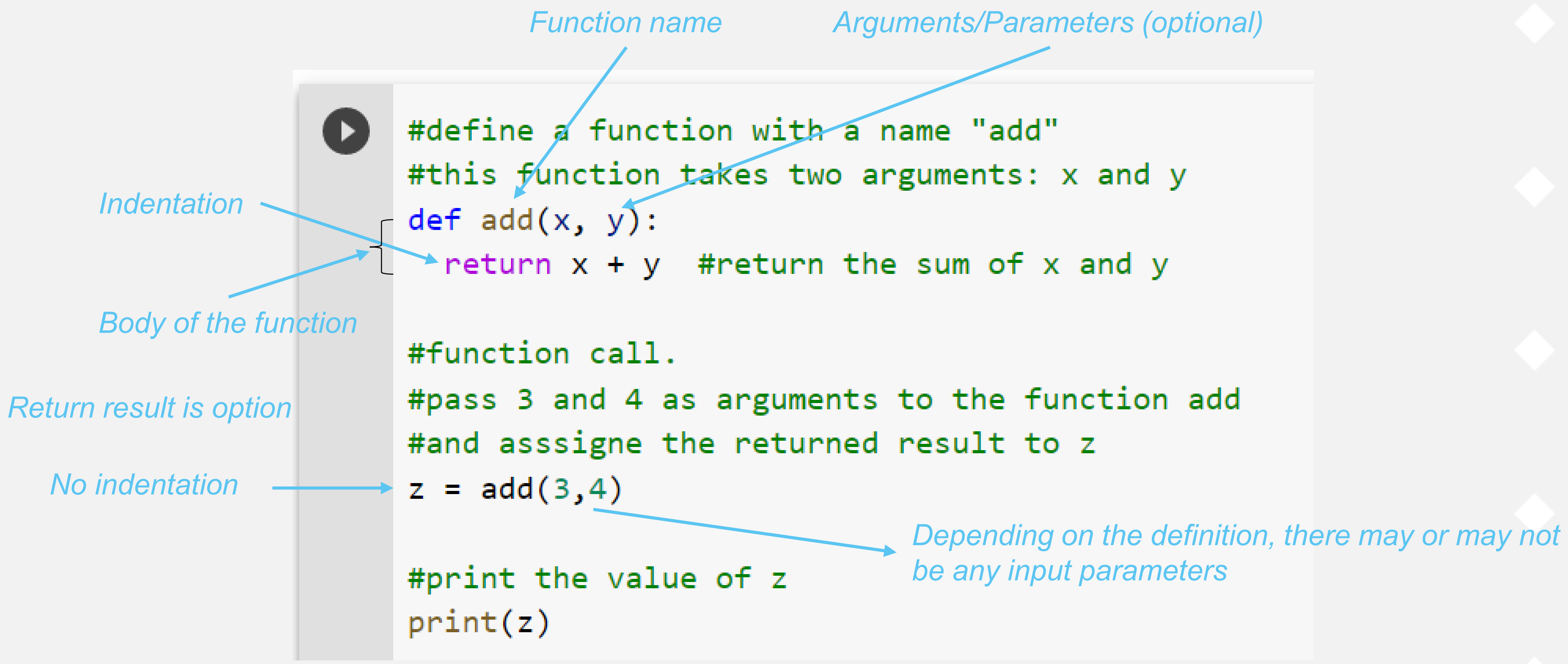
- Access strings and lists
- User defined functions
- Read/ Write to a file

VICTORIA UNIVERSITY MELBOURNE AUSTRALIA

© 2024 Victoria University. CRICOS No. 00124K (Melbourne), 02475D (Sydney and Brisbane), RTO 3113, TEQSA No. PRV12152, Provider Category: Australian University

< Function >

In Python, we can declare a new function by using the keyword “**def**”:



Function name
Arguments/Parameters (optional)
Indentation
Body of the function
Return result is option
No indentation
Depending on the definition, there may or may not be any input parameters

```python
#define a function with a name "add"
#this function takes two arguments: x and y
def add(x, y):
  return x + y  #return the sum of x and y

#function call.
#pass 3 and 4 as arguments to the function add
#and asssigne the returned result to z
z = add(3,4)

#print the value of z
print(z)
```

© 2024 Victoria University. CRICOS No. 00124K (Melbourne), 02475D (Sydney and Brisbane), RTO 3113, TEQSA No. PRV12152, Provider Category: Australian University

< Example>

Draw the following 3 figures by defining the common parts as “top()”,”bottom()”, and “line()”.

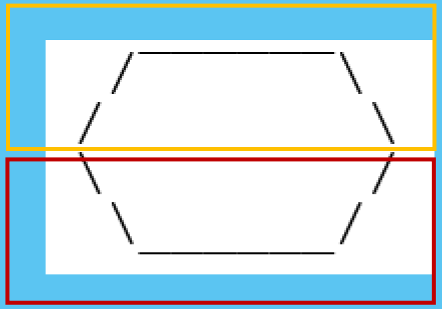


Figure 1

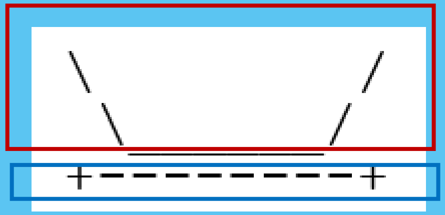


Figure 2

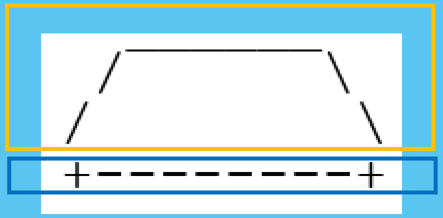
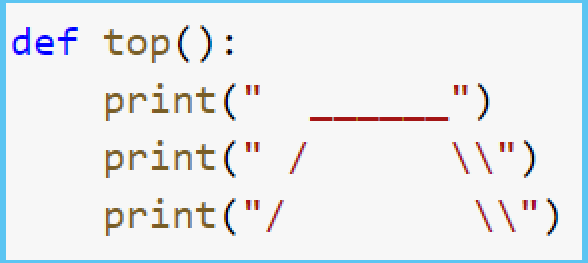
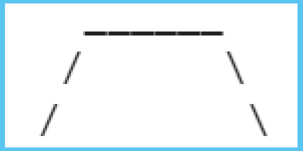


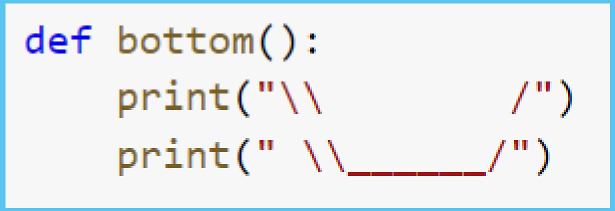
Figure 3



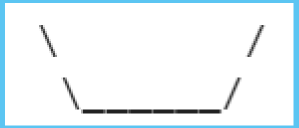
```python
def top():
    print("    _______")
    print("  /         \\")
    print("/           \\")
```



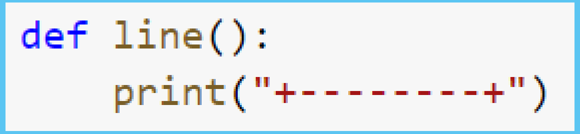
```
  _______
 /       \
/         \
```



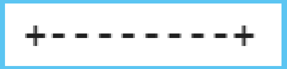
```python
def bottom():
    print("\\          /")
    print(" \\_______/")
```



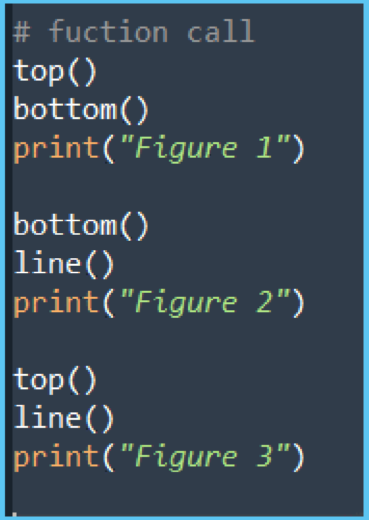
```
\         /
 \_______/
```



```python
def line():
    print("+--------+")
```



```
+--------+
```



```python
# fuction call
top()
bottom()
print("Figure 1")

bottom()
line()
print("Figure 2")

top()
line()
print("Figure 3")
```

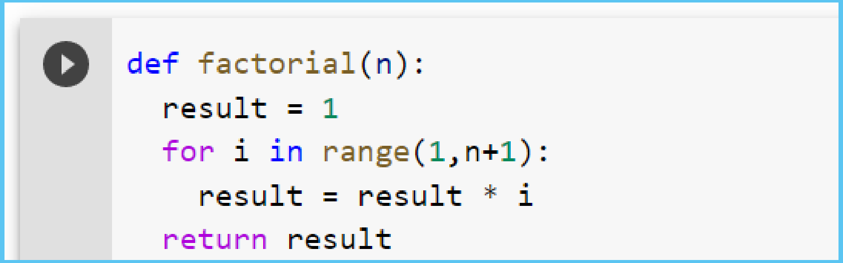
VICTORIA UNIVERSITY MELBOURNE AUSTRALIA
© 2024 Victoria University. CRICOS No. 00124K (Melbourne), 02475D (Sydney and Brisbane), RTO 3113, TEQSA No. PRV12152, Provider Category: Australian University

< Exercise >

Factorial of a positive integer *n* is denoted as *n!*.
This is the product of all positive integers less than or equal to *n*:

$$n! = n \times (n-1) \times (n-2) \ldots \ldots \times 3 \times 2 \times 1$$

Define a function to calculate **factorial(n)**.

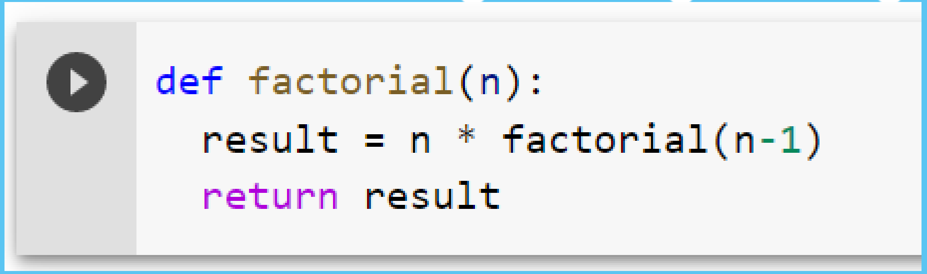


```python
def factorial(n):
  result = 1
  for i in range(1,n+1):
    result = result * i
  return result
```

$$n! = n \times (n-1) \times (n-2) \ldots \ldots \times 3 \times 2 \times 1$$

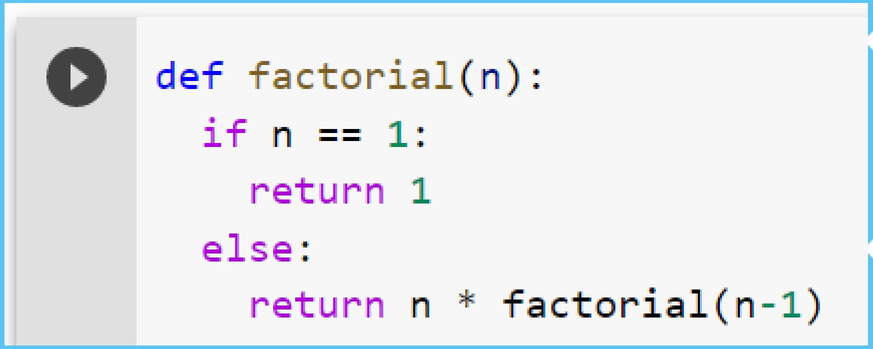
$$\Rightarrow \quad n! = n \times (n-1)!$$

In this case, assume that we have defined a function *factorial(n)*, then we can produce the result of *factorial(n)* = n x *factorial(n-1)*:



```python
def factorial(n):
  result = n * factorial(n-1)
  return result
```

What if n=1?
Note: can not proceed with factorial(0)



```python
def factorial(n):
  if n == 1:
    return 1
  else:
    return n * factorial(n-1)
```

© 2024 Victoria University. CRICOS No. 00124K (Melbourne), 02475D (Sydney and Brisbane), RTO 3113, TEQSA No. PRV12152, Provider Category: Australian University

< Contents >

- Access strings and lists
- User defined functions
- Read/ Write to a file

VICTORIA UNIVERSITY
MELBOURNE AUSTRALIA

© 2024 Victoria University. CRICOS No. 00124K (Melbourne), 02475D (Sydney and Brisbane), RTO 3113, TEQSA No. PRV12152, Provider Category: Australian University

< Open a file >

open(file, mode)

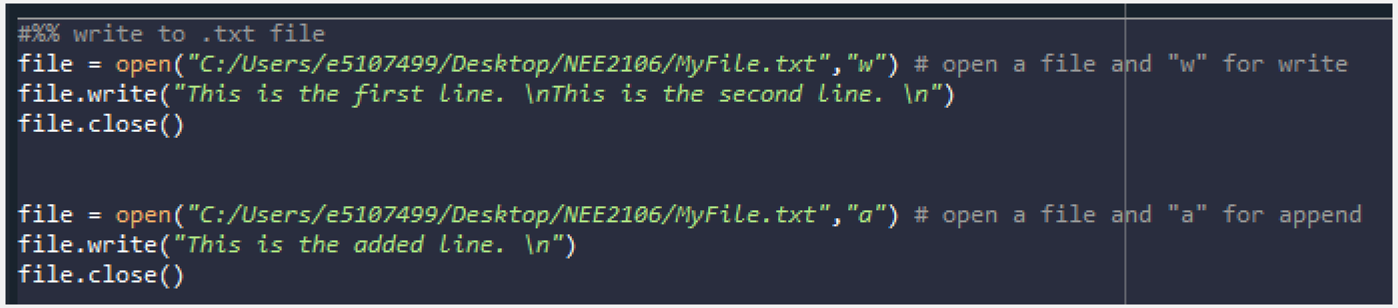
- **file**: the full path of the file you want to open
- **mode**:
  - use “**r**” for reading mode;
  - “**w**” for writing mode (write as new);
  - “**a**” for appended mode (add after existing)
  - “**b**” for binary mode
  - “**t**” for text mode (default), handle text files only

Modes	Description
r	Opens a file for reading only. The file pointer is placed at the beginning of the file. This is the default mode.
rb	Opens a file for reading only in binary format. The file pointer is placed at the beginning of the file. This is the default mode.
r+	Opens a file for both reading and writing. The file pointer will be at the beginning of the file.
rb+	Opens a file for both reading and writing in binary format. The file pointer will be at the beginning of the file.
w	Opens a file for writing only. Overwrites the file if the file exists. If the file does not exist, creates a new file for writing.
wb	Opens a file for writing only in binary format. Overwrites the file if the file exists. If the file does not exist, creates a new file for writing.
w+	Opens a file for both writing and reading. Overwrites the existing file if the file exists. If the file does not exist, creates a new file for reading and writing.
wb+	Opens a file for both writing and reading in binary format. Overwrites the existing file if the file exists. If the file does not exist, creates a new file for reading and writing.
a	Opens a file for appending. The file pointer is at the end of the file if the file exists. That is, the file is in the append mode. If the file does not exist, it creates a new file for writing.
ab	Opens a file for appending in binary format. The file pointer is at the end of the file if the file exists. That is, the file is in the append mode. If the file does not exist, it creates a new file for writing.
a+	Opens a file for both appending and reading. The file pointer is at the end of the file if the file exists. The file opens in the append mode. If the file does not exist, it creates a new file for reading and writing.
ab+	Opens a file for both appending and reading in binary format. The file pointer is at the end of the file if the file exists. The file opens in the append mode. If the file does not exist, it creates a new file for reading and writing.

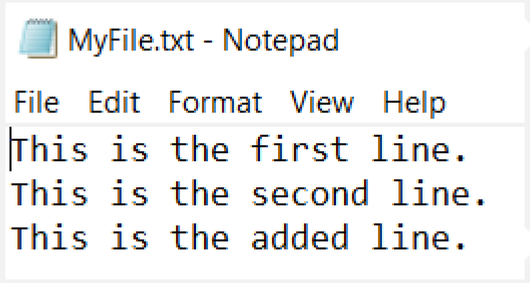
© 2024 Victoria University. CRICOS No. 00124K (Melbourne), 02475D (Sydney and Brisbane), RTO 3113, TEQSA No. PRV12152, Provider Category: Australian University

< Write to a file >

Data can be output to a text file using a **file** object:



```
#%% write to .txt file
file = open("C:/Users/e5107499/Desktop/NEE2106/MyFile.txt","w") # open a file and "w" for write
file.write("This is the first line. \nThis is the second line. \n")
file.close()


file = open("C:/Users/e5107499/Desktop/NEE2106/MyFile.txt","a") # open a file and "a" for append
file.write("This is the added line. \n")
file.close()
```

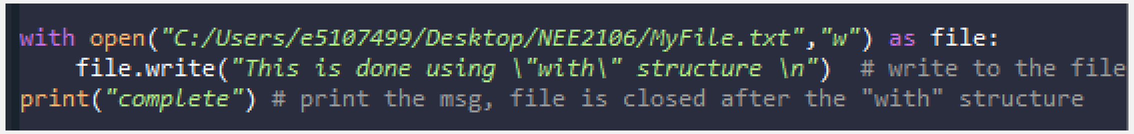


MyFile.txt - Notepad

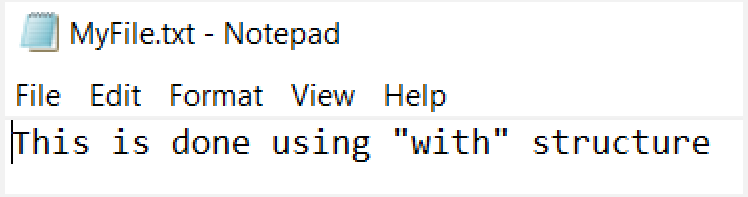
File Edit Format View Help

```
This is the first line.
This is the second line.
This is the added line.
```

Alternatively, use “with open () as file” structure exempt the need of close()



```
with open("C:/Users/e5107499/Desktop/NEE2106/MyFile.txt","w") as file:
    file.write("This is done using \"with\" structure \n")  # write to the file
print("complete") # print the msg, file is closed after the "with" structure
```



MyFile.txt - Notepad

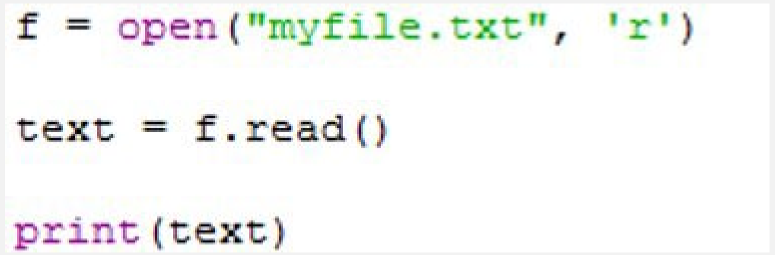
File Edit Format View Help

```
This is done using "with" structure
```

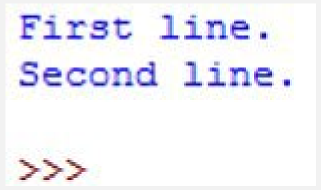
© 2024 Victoria University. CRICOS No. 00124K (Melbourne), 02475D (Sydney and Brisbane), RTO 3113, TEQSA No. PRV12152, Provider Category: Australian University

< Read a file>

Assume that your current working directory contain the file **myfile.txt**

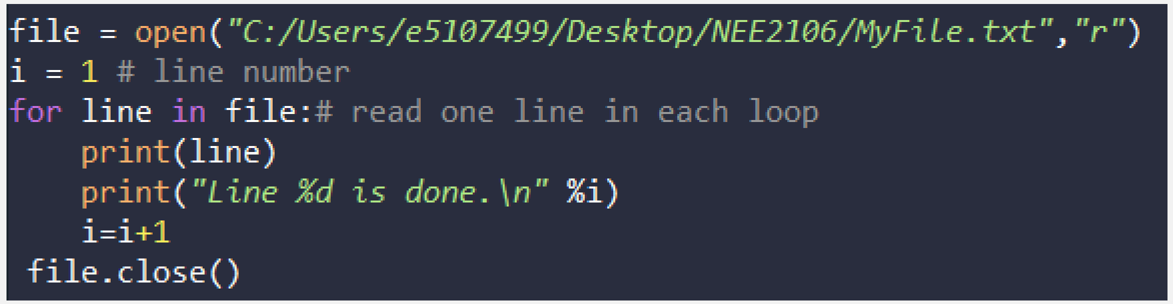


```
f = open("myfile.txt", 'r')

text = f.read()

print(text)
```

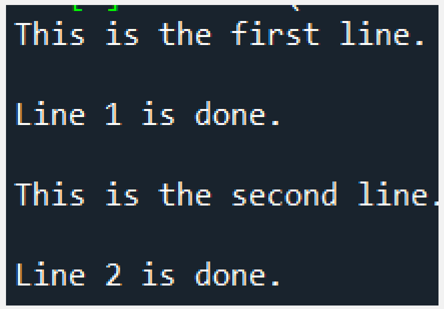


```
First line.
Second line.

>>>
```

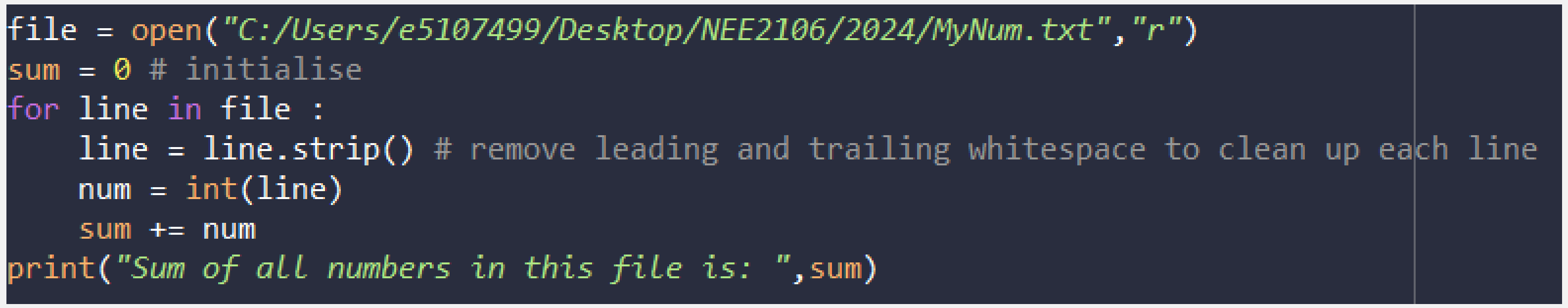
Alternatively, read and process one line at a time.



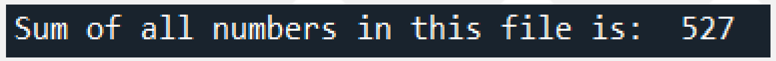
```
file = open("C:/Users/e5107499/Desktop/NEE2106/MyFile.txt","r")
i = 1 # line number
for line in file:# read one line in each loop
    print(line)
    print("Line %d is done.\n" %i)
    i=i+1
 file.close()
```



```
This is the first line.

Line 1 is done.

This is the second line.

Line 2 is done.
```



```
file = open("C:/Users/e5107499/Desktop/NEE2106/2024/MyNum.txt","r")
sum = 0 # initialise
for line in file :
    line = line.strip() # remove leading and trailing whitespace to clean up each line
    num = int(line)
    sum += num
print("Sum of all numbers in this file is: ",sum)
```



```
Sum of all numbers in this file is:  527
```

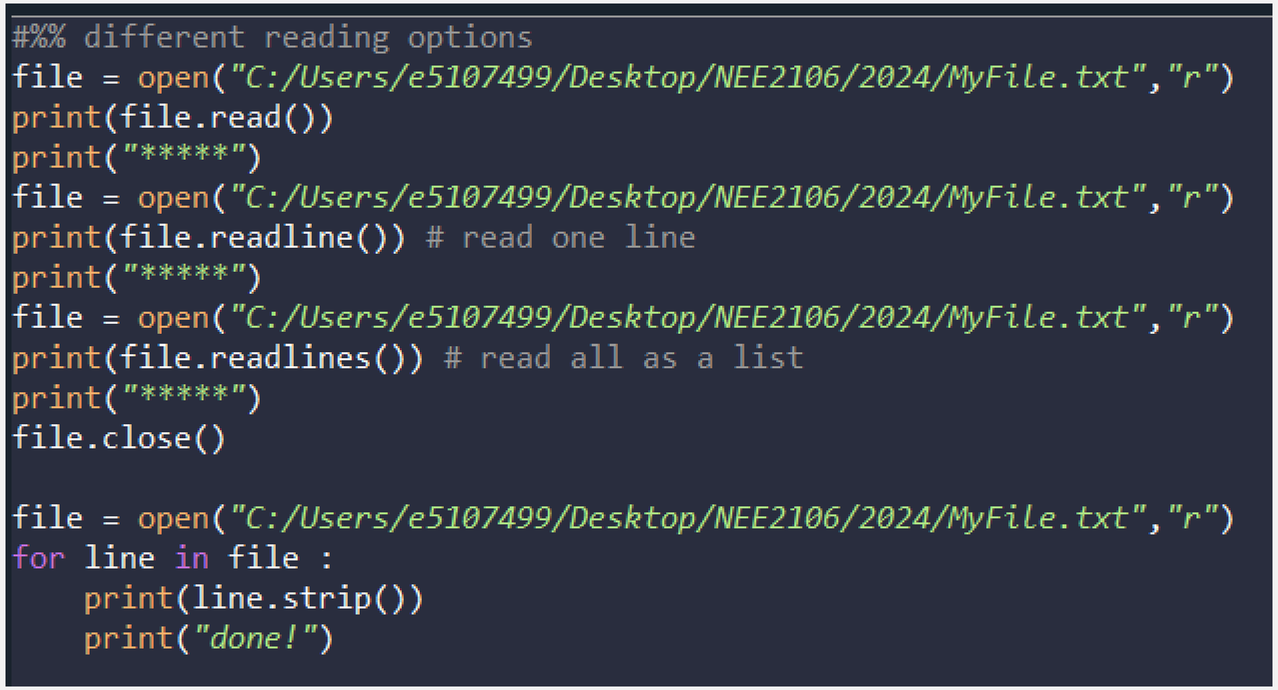
VICTORIA UNIVERSITY MELBOURNE AUSTRALIA

© 2024 Victoria University. CRICOS No. 00124K (Melbourne), 02475D (Sydney and Brisbane), RTO 3113, TEQSA No. PRV12152, Provider Category: Australian University

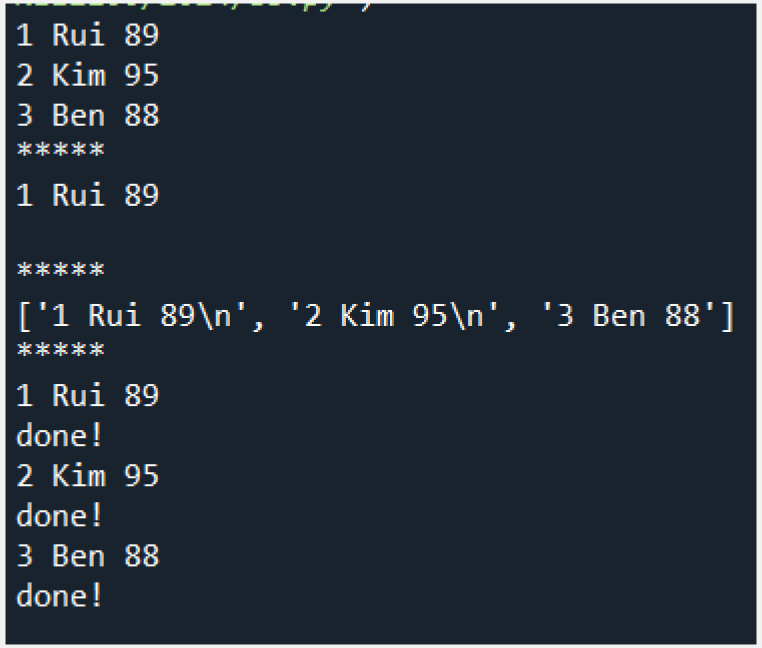
< Read a file>

Different reading options:

file.read() - file's entire contents as a string
file.readline() - next one line from file as a string
file.readlines() - file's contents as a list of lines



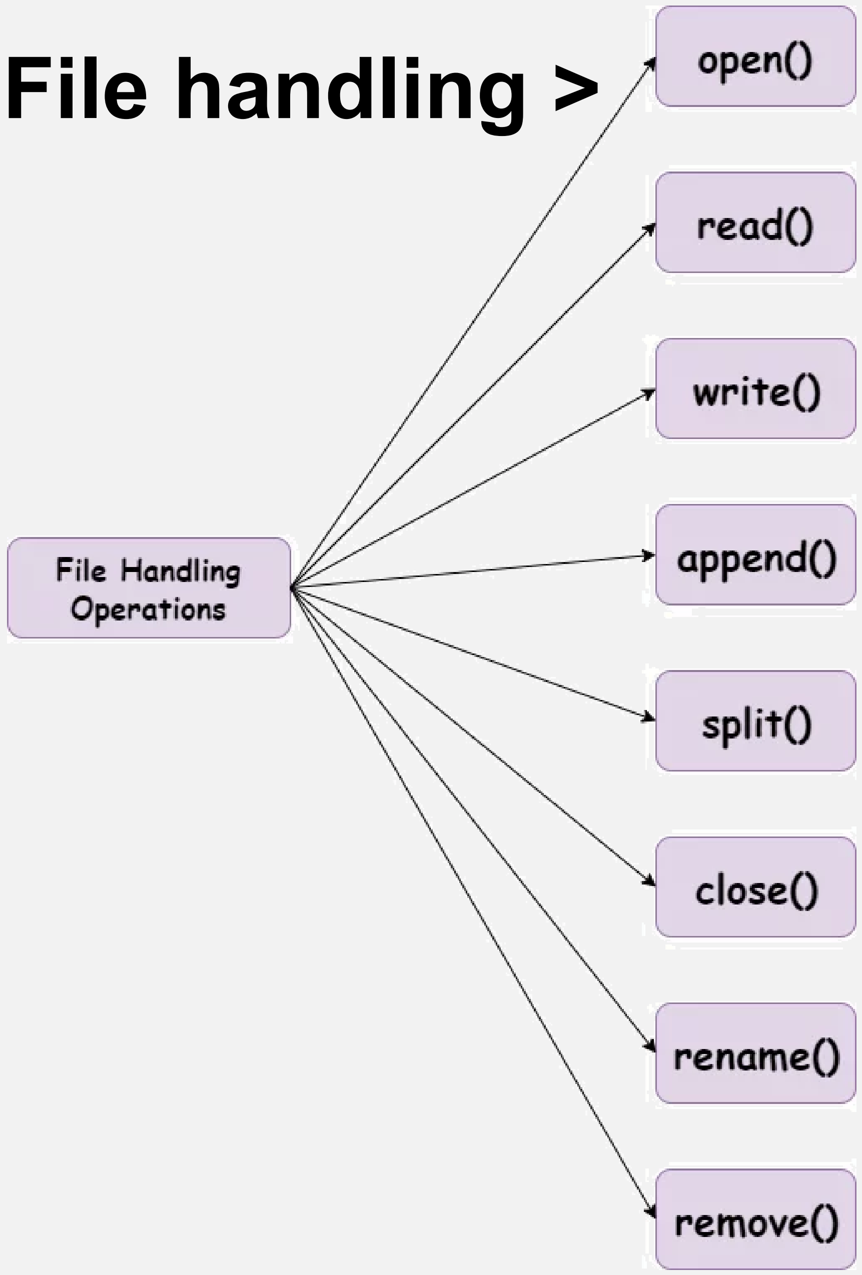
```
#%% different reading options
file = open("C:/Users/e5107499/Desktop/NEE2106/2024/MyFile.txt","r")
print(file.read())
print("*****")
file = open("C:/Users/e5107499/Desktop/NEE2106/2024/MyFile.txt","r")
print(file.readline()) # read one line
print("*****")
file = open("C:/Users/e5107499/Desktop/NEE2106/2024/MyFile.txt","r")
print(file.readlines()) # read all as a list
print("*****")
file.close()

file = open("C:/Users/e5107499/Desktop/NEE2106/2024/MyFile.txt","r")
for line in file :
    print(line.strip())
    print("done!")
```

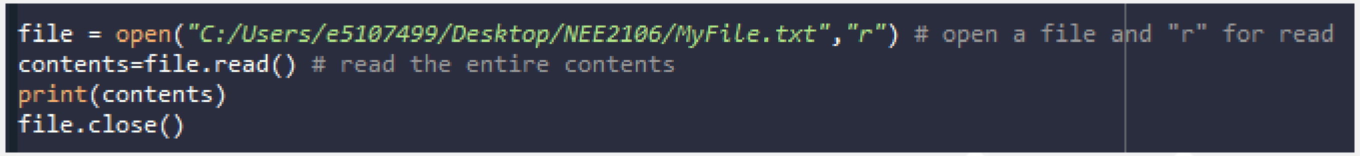


```
1 Rui 89
2 Kim 95
3 Ben 88
*****
1 Rui 89

*****
['1 Rui 89\n', '2 Kim 95\n', '3 Ben 88']
*****
1 Rui 89
done!
2 Kim 95
done!
3 Ben 88
done!
```

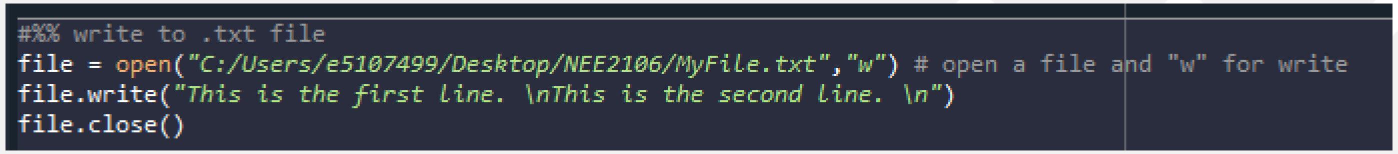
© 2024 Victoria University. CRICOS No. 00124K (Melbourne), 02475D (Sydney and Brisbane), RTO 3113, TEQSA No. PRV12152, Provider Category: Australian University

< File handling >

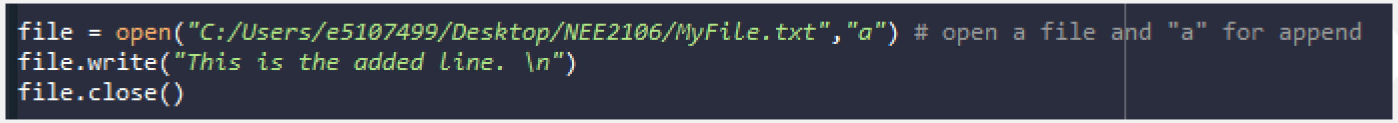




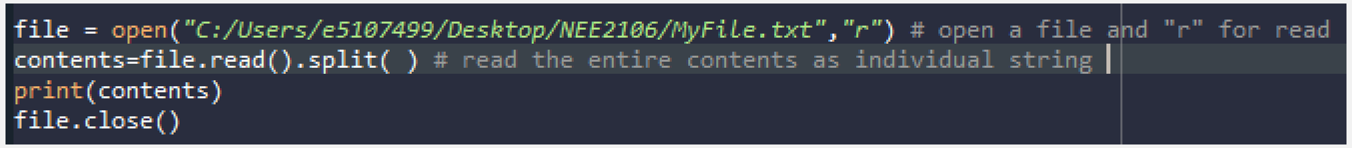
```
file = open("C:/Users/e5107499/Desktop/NEE2106/MyFile.txt","r") # open a file and "r" for read
contents=file.read() # read the entire contents
print(contents)
file.close()
```



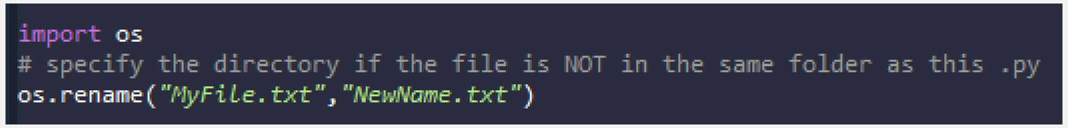
```
#%% write to .txt file
file = open("C:/Users/e5107499/Desktop/NEE2106/MyFile.txt","w") # open a file and "w" for write
file.write("This is the first line. \nThis is the second line. \n")
file.close()
```



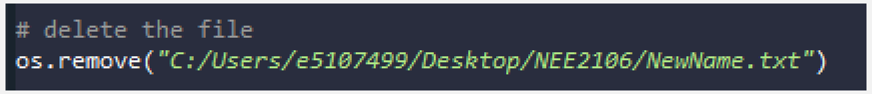
```
file = open("C:/Users/e5107499/Desktop/NEE2106/MyFile.txt","a") # open a file and "a" for append
file.write("This is the added line. \n")
file.close()
```



```
file = open("C:/Users/e5107499/Desktop/NEE2106/MyFile.txt","r") # open a file and "r" for read
contents=file.read().split( ) # read the entire contents as individual string
print(contents)
file.close()
```



```
import os
# specify the directory if the file is NOT in the same folder as this .py
os.rename("MyFile.txt","NewName.txt")
```



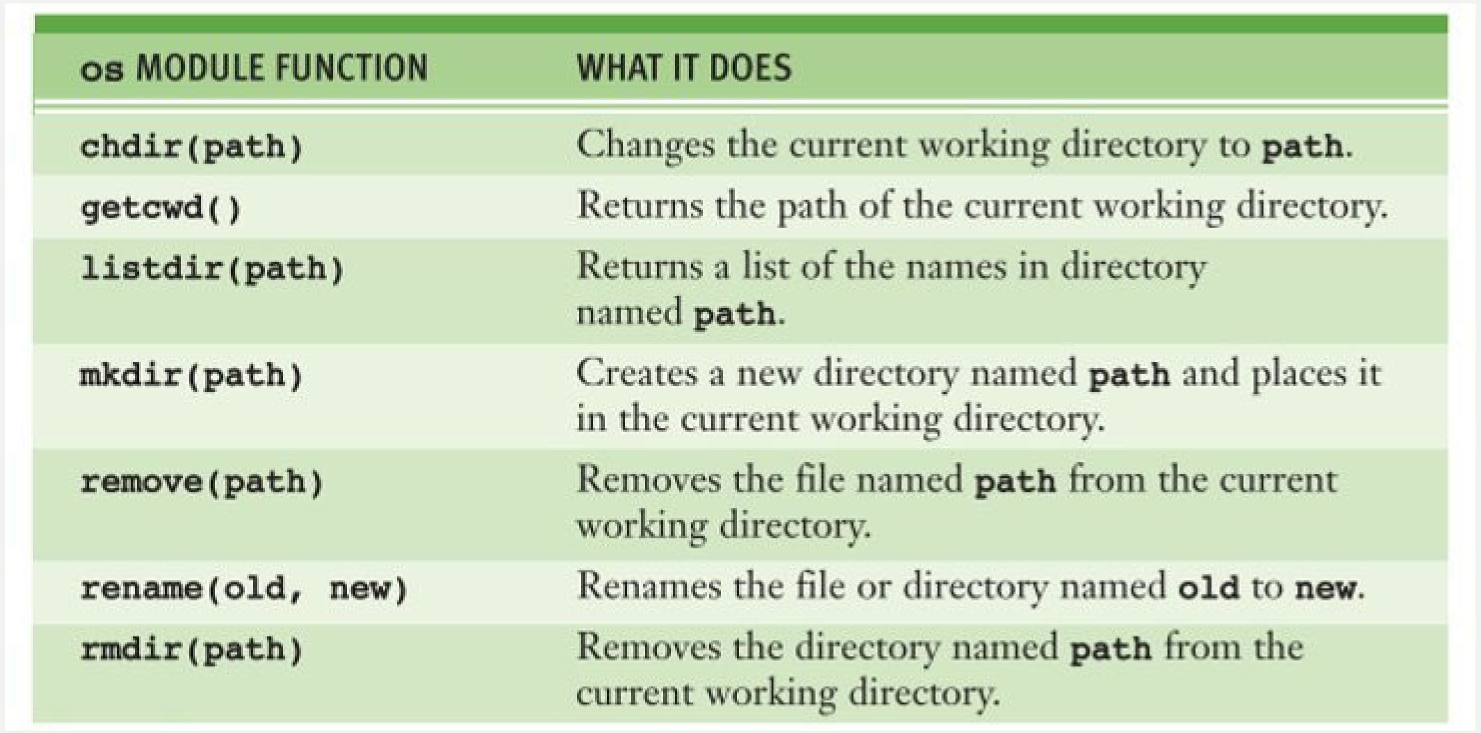
```
# delete the file
os.remove("C:/Users/e5107499/Desktop/NEE2106/NewName.txt")
```

Ref:
Python File Handling
Python os Module

VICTORIA UNIVERSITY
MELBOURNE AUSTRALIA

© 2024 Victoria University. CRICOS No. 00124K (Melbourne), 02475D (Sydney and Brisbane), RTO 3113, TEQSA No. PRV12152, Provider Category: Australian University

< os Module>



os MODULE FUNCTION	WHAT IT DOES
`chdir(path)`	Changes the current working directory to **path**.
`getcwd()`	Returns the path of the current working directory.
`listdir(path)`	Returns a list of the names in directory named **path**.
`mkdir(path)`	Creates a new directory named **path** and places it in the current working directory.
`remove(path)`	Removes the file named **path** from the current working directory.
`rename(old, new)`	Renames the file or directory named **old** to **new**.
`rmdir(path)`	Removes the directory named **path** from the current working directory.

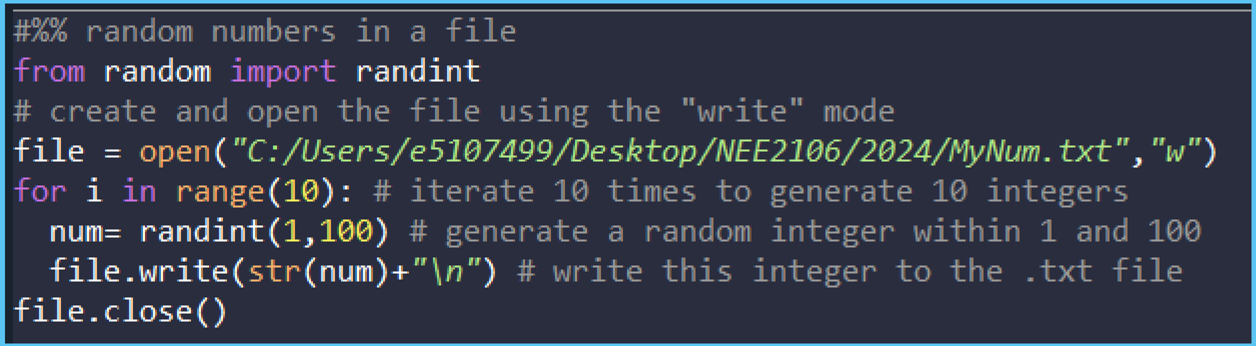
Ref:
Python File Handling
Python os Module

VICTORIA UNIVERSITY
MELBOURNE AUSTRALIA

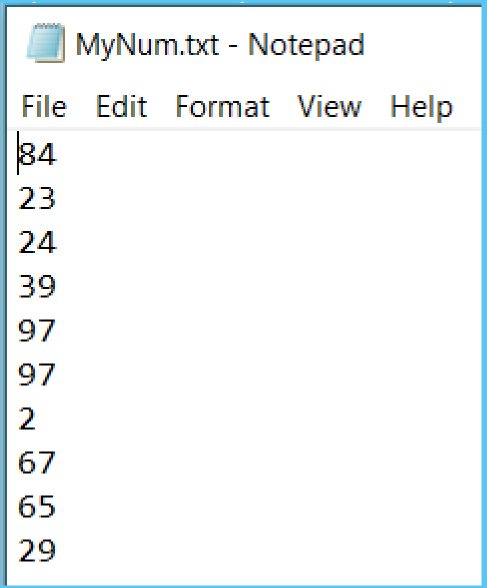
© 2024 Victoria University. CRICOS No. 00124K (Melbourne), 02475D (Sydney and Brisbane), RTO 3113, TEQSA No. PRV12152, Provider Category: Australian University

< Example>

Generate 10 random integers from 1-100 using the Python module "**random**" and write the numbers to a text file named **MyNum.txt**.

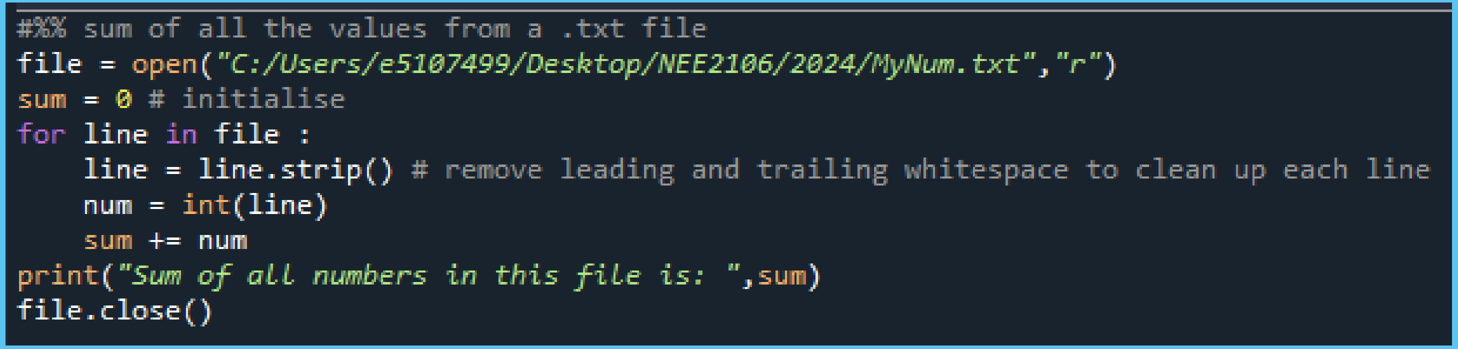


```
#%% random numbers in a file
from random import randint
# create and open the file using the "write" mode
file = open("C:/Users/e5107499/Desktop/NEE2106/2024/MyNum.txt","w")
for i in range(10): # iterate 10 times to generate 10 integers
  num= randint(1,100) # generate a random integer within 1 and 100
  file.write(str(num)+"\n") # write this integer to the .txt file
file.close()
```

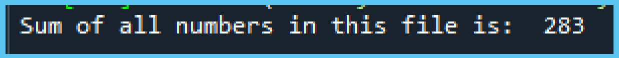


```
84
23
24
39
97
97
2
67
65
29
```

Read all the numbers in **MyNum.txt** to a Python variable and calculate the sum of all the values.



```
#%% sum of all the values from a .txt file
file = open("C:/Users/e5107499/Desktop/NEE2106/2024/MyNum.txt","r")
sum = 0 # initialise
for line in file :
    line = line.strip() # remove leading and trailing whitespace to clean up each line
    num = int(line)
    sum += num
print("Sum of all numbers in this file is: ",sum)
file.close()
```



```
Sum of all numbers in this file is:  283
```

© 2024 Victoria University. CRICOS No. 00124K (Melbourne), 02475D (Sydney and Brisbane), RTO 3113, TEQSA No. PRV12152, Provider Category: Australian University