PYTHON BASIC COURSE

DR. SAW MYAT SANDAR

STUDENT GUIDE BOOK 1/12/24 LEVEL 1

Table of Contents

Introduction to Python

Python is a high-level, general-purpose programming language. Its design philosophy emphasizes code readability with the use of significant indentation.

It supports multiple programming paradigms, including structured, object-oriented and functional programming. Python is a popular programming language. Python can be used on a server to create web applications.

Code Editor for Python (IDE)

1. IDLE

https://www.python.org/downloads/

2. VS Code

https://code.visualstudio.com/download

3. Thonny

   https://thonny.org/

Python Syntax

```
print("Hello, World!")
```

Python Indentation

Indentation refers to the spaces at the beginning of a code line. Where in other programming languages the indentation in code is for readability only, the indentation in Python is very important. Python uses indentation to indicate a block of code.

```
if 5 > 2:
    print("Five is greater than two!")
```

Python Comments

- Comments can be used to explain Python code.
- Comments can be used to make the code more readable.
- Comments can be used to prevent execution when testing code.
- Comments starts with a #, and Python will ignore them:

Single Line Comment

```
#This is a single comment
```

Multiline Comment

```
“””
This is a Muliline comments
written in more than just one line “””
```


Variable

Variables are containers for storing data values. Python has no command for declaring a variable.

A variable is created the moment you first assign a value to it. Variables do not need to be declared with any particular type.

A variable can have descriptive name (age, carname, total_volume).

Rules for Python variables:

- A variable name must start with a letter or the underscore character and cannot start with a number
- contain alpha-numeric characters and underscores (A-z,0-9,and_)
- Variable names are case-sensitive (age, Age and AGE are three different variables)

Example1 Variable and Data Type 


```
a=10
b=20
c=a+b
print(c)

d=5
e="John"
f=d+e #error
f=str(d)+e #concat
print(f)
print(type(d)) #int
print(type(e)) #string
print(type(f)) #string

firstNumber=10
secondNumber=20
thirdNumber= firstNumber + secondNumber
print(thirdNumber)
print(thirdnumber) #error case sensitive

x,y,z="Orange","Banana","Cherry"
print(x)
print(y)
print(z)

num1,num2,num3=23,34,56
print(num1)
print(num2)
print(num3)
```


```
print(num1,x)
print(str(num1)+x)

xx=yy=zz="Orange"
print(xx)
print(yy)
print(zz)

fruits=["apple","banana","cherry"]
print(fruits)
a,b,c=fruits
print(a)
print(b)
print(c)

aa="Python"
bb="is"
cc="awesome"
print(aa,bb,cc)
print(aa+bb+cc)
```

Result

```
>>> %Run eg1.py
30
5John
<class 'int'>
<class 'str'>
<class 'str'>
30
Orange
Banana
Cherry
23
34
56
23 Orange
23Orange
Orange
Orange
Orange
['apple', 'banana', 'cherry']
apple
banana
cherry
Python is awesome
Pythonisawesome
```

Local Variable and Global Variable

Global variables are created in the main program and can be accessed from any part of the program.

Local variables are created inside a function and are only accessible in scope within the function in which they are declared.

Example2 Global Local Variable 02

```
q="awesome"#global variable
def myfun():
    global cc
    cc="very good"
    q="good"#local variable
    print("1. Python is "+q)
    print(cc)
myfun()
print("2. Python is "+q)
print(cc)
```

Result

```
>>> %Run eg2.py
 1. Python is good
 test
 2. Python is awesome
 test
```

Data Type

In programming, data type is an important concept. Variables can store data of different types, and different types can do different things. Python has the following data types built-in by default, in these categories:

1. Text Type	:	str
2. Numeric Types	:	int, float, complex
3. Sequence Types	:	list, tuple, range
4. Mapping Type	:	dict
5. Set Types	:	set, frozenset
6. Boolean Type	:	bool
7. Binary Types	:	bytes, bytearray, memoryview
8. None Type	:	NoneType

Example3 Data Type 03

```
a = 1
b = 35656222554887711
c = -3255522
print(type(a)) #int
print(type(b))
print(type(c))

d = 1.10
e = 1.0
f = -35.59
print(type(d)) #float - real
print(type(e))
print(type(f))

g = 35e3
h = 12E4
i = -87.7e10
print(type(g),g) #float - real
print(type(h),h)
print(type(i),i)

j = 3+5j
k = 5j
l = -5j
print(type(j)) #complex
print(type(k))
print(type(l))
```

Basic Data Types in Python:

1. Integer
2. Real
3. String
4. Boolean

Result

```
>>> %Run eg3.py
 <class 'int'>
 <class 'int'>
 <class 'int'>
 <class 'float'>
 <class 'float'>
 <class 'float'>
 <class 'float'> 35000.0
 <class 'float'> 120000.0
 <class 'float'> -877000000000.0
 <class 'complex'>
 <class 'complex'>
 <class 'complex'>
```

String

Strings in python are surrounded by either **single quotation marks**, or **double quotation marks**. **Strings** in Python are arrays of bytes representing unicode characters. However, **Python does not have a character data type**, a single character is simply a string with a length of 1. **Square brackets** can be used to access elements of the string. Since strings are arrays, we can use for loop.

To get the **length** of a string, use the ***len()*** function.

To check if a certain phrase or character is present in a string, we can use the keyword ***in***. To check if a certain phrase or character is NOT present in a string, we can use the keyword ***not in***.

Method	Description
capitalize()	Converts the first character to upper case
center()	Returns a centered string
count()	Returns the number of times a specified value occurs in a string
encode()	Returns an encoded version of the string
endswith()	Returns true if the string ends with the specified value
find()	Searches the string for a specified value and returns the position of where it was found
format()	Formats specified values in a string
index()	Searches the string for a specified value and returns the position of where it was found
isalnum()	Returns True if all characters in the string are alphanumeric
isalpha()	Returns True if all characters in the string are in the alphabet
isdecimal()	Returns True if all characters in the string are decimals
isdigit()	Returns True if all characters in the string are digits
islower()	Returns True if all characters in the string are lower case
isnumeric()	Returns True if all characters in the string are numeric
isspace()	Returns True if all characters in the string are whitespaces
istitle()	Returns True if the string follows the rules of a title
isupper()	Returns True if all characters in the string are upper case
join()	Joins the elements of an iterable to the end of the string
lower()	Converts a string into lower case
lstrip()	Returns a left trim version of the string
replace()	Returns a string where a specified value is replaced with a specified value
rsplit()	Splits the string at the specified separator, and returns a list
rstrip()	Returns a right trim version of the string
split()	Splits the string at the specified separator, and returns a list
startswith()	Returns true if the string starts with the specified value
strip()	Returns a trimmed version of the string
title()	Converts the first character of each word to upper case
translate()	Returns a translated string
upper()	Converts a string into upper case

Example4 String 04

```
z="Hello" #single line string
print(z)

a = """My name is Saw Myat Sandar.
I teach Computer Science.
I love Python.
This is multiline string"""
print(a)

a="Hello"
print(a)
print(a[0])#H
print(a[1])#e
print(a[2])
print(a[3])
print(a[4])
for x in "banana":
    print(x)

firstName="Saw Myat"
print(len(firstName))

txt1 = "The best things in life are free!"
print("free" in txt1) #True
print("expensive" in txt1) #False

if "free" in txt1:
    print("Yes, 'free' is present.")

txt2 = "The best things in life are free!"
if "expensive" not in txt2:
  print("No, 'expensive' is NOT present.")

a=" Hello: Good: Morning "
print(a.upper())
print(a.lower())
print(a.strip())
print(a.replace("H","J"))
print(a.split(":"))

age=37
txt="You are {} year old"
```

```
print(txt.format(age))

age1=input("How old are you?")
txt1="You are {} year old"
print(txt1.format(age1))

txt2="We are the so called \"Vikings\" from the nort."
print(txt2)
txt3="We are the so called \'Vikings\' from the nort."
print(txt3)
txt4="We are the so called Viking\n from the nort."
print(txt4)
txt5="We are the so called Viking\t from the nort."
print(txt5)
mytxt = "I love apple, apples are my favorite fruit"
x = mytxt.count("apple")
print(x)
y=mytxt.count("love")
print(y)
```

Result

```
>>> %Run eg4.py
Hello
My name is Saw Myat Sandar
I teach Computer Science
I love Python.
This is multiline string
Hello
H
e
l
l
o
b
a
n
a
n
a
8
True
False
Yes, 'free' is present.
No, 'expensive' is NOT present.
```


```
 HELLO: GOOD: MORNING
 hello: good: morning
Hello: Good: Morning
 Jello: Good: Morning
[' Hello', ' Good', ' Morning ']
You are 37 year old
How old are you?10
You are 10 year old
We are the so called "Vikings" from the nort.
We are the so called 'Vikings' from the nort.
We are the so called Viking
 from the nort.
We are the so called Viking      from the nort.
2
1
```

Slicing

You can return a range of characters by using the slice syntax.
Specify the start index and the end index, separated by a colon, to return a part of the string.
By leaving out the start index, the range will start at the first character: print(b[:5])
By leaving out the end index, the range will go to the end: print(b[2:])

Example5 Slicing 05

```
b = "Hello, World!"
print(b)
print(b[0])
print(len(b))
print(b[2:5])#index,length
print(b[1:6])
print(b[2:])#start from 2 up to end
print(b[:5])#from start to less than 5
print(b[-5:-2])#length,index
```

Result

```
>>> %Run eg5.py
 Hello, World!
 H
 13
 llo
 ello,
 llo, World!
 Hello
 orl
```

Random

Python does not have a random() function to make a random number, but Python has a ***built-in module called random*** that can be used to make random numbers:

Function	Description
random.randint(0, 9)	Returns any random integer from 0 to 9
random.randrange(20)	Returns a random integer from 0 to 19
random.randrange(2, 20)	Returns a random integer from 2 to 19.
random.randrange(100, 1000, 3)	Returns any random integer from 100 to 999 with step 3. For example, any number from 100, 103, 106 … 994, 997.
random.randrange(-50, -5)	Returns a random negative integer between -50 to -6.

Example6 Random 06

```
import random
a=random.randrange(1,5)
print(a)

b=random.randrange(0,9)
print("Random number between 0 and 9:",b)

c=random.randint(1,5)
print(c)

mylist=["apple","banana","cherry"]
print(random.choice(mylist))

mylist2=["apple","banana","cherry"]
print(random.sample(mylist2,k=2))

print(random.random())

random.seed(10)
print(random.random())

random.seed(5)
print(random.random())
```

Result

```
>>> %Run eg6.py
2
Random number between 0 and 9: 3
4
apple
['apple', 'banana']
0.8793499144590566
0.5714025946899135
0.6229016948897019
```

Input Output

Python allows for user input. That means we are able to ask the user for input. The method is a bit different in Python 3.6 than Python 2.7.

Python 3.6 uses the ***input()*** method. Python 2.7 uses the raw_input() method.

The Python ***print()*** function is often used to output variables.

Operator

Operators are used to perform operations on variables and values.

In the example below, we use the + operator to add together two values:

Python divides the operators in the following groups:

1. Arithmetic operators (+,-,*,/,%,**,//)
2. Assignment operators(a=b)
3. Comparison operators (==,!=,>,>=,<,<=)
4. Logical operators (AND, OR, NOT)

Relational operators in python are:

Equal To(==), Not Equal To(! =), Greater Than(>), Less Than(<), Greater Than or Equal To(>=), and Lesser Than or Equal To(<=).

Example7 Operator 07

```
#example7
firstNumber=int(input("Enter first number:"))
secondNumber=int(input("Enter second number:"))

add=firstNumber+secondNumber
print("Addition is", add)

sub=firstNumber-secondNumber
print("Sub is", sub)

mul=firstNumber*secondNumber
```

```
print("Mul is", mul)

div=firstNumber/secondNumber
print("Div is", div)

mod=firstNumber%secondNumber
print("Mod is", mod)

flo=firstNumber//secondNumber
print("Floor is", flo)

exp=firstNumber**secondNumber
print("Exp is", exp)
```

Result

```
>>> %Run eg7.py
 Enter first number:3
 Enter second number:2
 Addition is 5
 Sub is 1
 Mul is 6
 Div is 1.5
 Mod is 1
 Floor is 1
 Exp is 9
```

Conditional Statement

These conditions can be used in several ways, most commonly in "if statements" and loops.

An "if statement" is written by using the ***if*** keyword.

The ***elif*** keyword is python's way of saying "if the previous conditions were not true, then try this condition".

The ***else*** keyword catches anything which isn't caught by the preceding conditions.

The ***and*** keyword is a logical operator, and is used to combine conditional statements: Returns True if both statements are true (eg x < 5 and x < 4)

The ***or*** keyword is a logical operator, and is used to combine conditional statements: Returns True if one of the statements is true (eg x < 5 or x < 4)

The ***not*** keyword is a logical operator, Reverse the result, returns False if the result is true (eg not(x < 5 and x < 10))

You can have if statements inside if statements, this is called ***nested if*** statements.
if statements ***cannot be empty***, but if you for some reason have an if statement with no content, put in the ***pass*** statement to avoid getting an error.

Example8 BMI Conditional 08

```
#example8
weight=float(input("Enter weight(lb):"))
height=float(input("Enter height(in):"))
bmi=(703*weight)/(height**2)
print(bmi)

if bmi<18:
    print("Under weight")
elif bmi>=18 and bmi<25:
    print("Normal")
elif bmi>=25 and bmi<30:
    print("Over weight")
elif bmi>=30 and bmi<35:
    print("Obese")
else:
    print("Extremely Obese")
```

Result

```
>>> %Run eg8.py
 Enter weight(lb):135
 Enter height(in):62
 24.689125910509887
 Normal
```

Example9 Exam Result 09

```
#example9_exam
myan=int(input("Enter myanmar mark:"))
eng=int(input("Enter english mark:"))
math=int(input("Enter math mark:"))

#if not(myan<40 or eng<40 or math<40):
if myan>=40 and eng>=40 and math>=40:
    print("Pass")
else:
    print("Fail")
```

Result

```
>>> %Run eg9.py
 Enter myanmar mark:70
 Enter english mark:80
 Enter math mark:90
 Pass
```

Example10 Exam 10

```
#example10_exam
myan=int(input("Enter myanmar mark:"))
eng=int(input("Enter english mark:"))
math=int(input("Enter math mark:"))
if myan<40 or eng<40 or math<40:
    print("Fail")
else:
    print("Pass")
```

Result

```
>>> %Run eg10.py
 Enter myanmar mark:75
 Enter english mark:85
 Enter math mark:35
 Fail
```

Example11 Guessing Game 11

```
#example11_guessingGame
#mystNum=3
import random
mystNum=random.randint(1,10)
guess=int(input("Enter guess number:"))
if guess>10:
    print("Too High")
else:
    if guess==mystNum:
        print("Correct")
    else:
        print("Incorrect",mystNum)
```

Result

```
>>> %Run eg11.py
 Enter guess number:6
 Incorrect 7
```

Control Flow

Python has two primitive loop commands:

1. While Loop ➔ while True ➔ anotherGo=True ➔ password
2. For loop ➔ Array

While Loop

With the while loop we can execute a set of statements as long as a condition is true.

With the break statement we can stop the loop even if the while condition is true:

With the continue statement we can stop the current iteration, and continue with the next:

With the else statement we can run a block of code once when the condition no longer is true:

Example12 While Loop 12

```
#example12
i=1
while i<=5:
    print(i)
    i+=1

a=10
while a<=15:
    print(a)
    if a==12:
        break
    a+=1

b=20
while b<=25:
    b+=1
    if b==22:
        continue
    print(b)

c=30
while c<=35:
    c+=1
    if c==32:
        continue
```

```
    print(c)
else:
    print("Finished")

n=3
i=1
while i<=n:
    j=1
    while j<=i:
        print("*",end=" ")
        j+=1
    print()
    i+=1
```

For Loop

A for loop is used for iterating over a sequence (that is either a list, a tuple, a dictionary, a set, or a string).

The for loop does not require an indexing variable

break statement - **stop** the loop

continue statement - **skip**, and continue with the next:

The range(6), which means values from 0 to 6 (but not including 6) increments by 1 (by default)

range(2, 6), which means values from 2 to 6 (but not including 6) increments by 1 (by default)

range(2, 30, 3) , which means values from 2 to 30 (but not including 30), increments by 3

Example13 For Loop

```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
    print(x)

fruits = ["apple", "banana", "cherry","kiwi"]
for y in fruits:
    if y=="cherry":
        break
    print(y)

fruits = ["apple", "banana", "cherry","kiwi"]
for y in fruits:
    if y=="cherry":
        continue
    print(y)
```

```
for x in range(6):
    print("X value",x)

for y in range(2,6):
    print("Y value",y)

for z in range(2,30,3):
    print("Z value",z)
for x in range(6):
    if x==3: break
    print(x)
else:
    print("Finally Finished")

for x in range(6):
    if x==3: continue
    print(x)
else:
    print("Finally Finished")

color = ["red", "big", "tasty"]
fruits = ["apple", "banana", "cherry"]
for x in color:
    for y in fruits:
        print(x,y)
    print(" ")

#pyramid
for i in range(5):
    for j in range(5, i, -1):
        print(" ", end="")
    for k in range(i + 1):
        print("* ", end="")
    print()
for i in range(5):
    for j in range(i):
        print(" ", end="")
    for j in range(i, 5):
        print("* ", end="")
    print()
```


Function

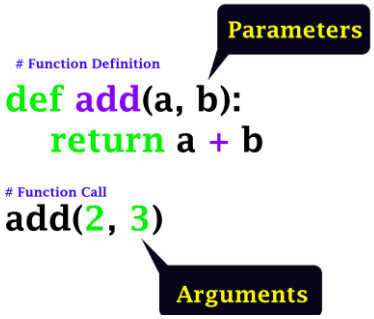
A function (sub program) is a block of code which only runs when it is called.

You can pass data, known as ***parameters***, into a function.

A function can return data as a result.

In Python a function is defined using the ***def*** keyword

To call a function, use the ***function name*** followed by ***parenthesis***:



Example14 Function 14

```
#example14
def my_function(fname):
    print(fname,"Smith")
#main
my_function("John")

def my_function2(food):
    for x in food:
        print(x)

fruits=["apple","banana","cherry"]
cake=["cheese","milk","chocolate","strawberry"]
breakfast=["egg","orange juice"]
my_function2(fruits)
my_function2(cake)
my_function2(breakfast)

def my_function(country = "Norway"):
  print("I am from " + country)

my_function("Sweden")
my_function("India")
my_function()
my_function("Brazil")

#Recursion
```

```
def sum(k):
  if(k > 0):
    result = k + sum(k - 1)
    print(result)
  else:
    result = 0
  return result
print("\n\nRecursion Example Results")
sum(5)
```

sum(5)

k>0 result=k+sum(k-1)

5>0 result=5+sum(4) ➔ r=5+10=15

4>0 result=4+sum(3) ➔ r=4+6=10

3>0 result=3+sum(2) ➔ r=3+3=6

2>0 result=2+sum(1) ➔ r=2+1=3

1>0 result=1+sum(0) ➔ r=1+0=1

0>0 ➔ result=0

Array

Python does not have built-in support for Arrays, but Python ***Lists*** can be used instead.

To work with arrays in Python you will have to import a library, like the ***NumPy*** library.

Arrays are used to store multiple values in one single variable:

An array is a special variable, which can hold more than one value at a time.

Method	**Description**
append()	Adds an element at the end of the list
clear()	Removes all the elements from the list
copy()	Returns a copy of the list
count()	Returns the number of elements with the specified value
extend()	Add the elements of a list (or any iterable), to the end of the current list
index()	Returns the index of the first element with the specified value
insert()	Adds an element at the specified position
pop()	Removes the element at the specified position
remove()	Removes the first item with the specified value
reverse()	Reverses the order of the list
sort()	Sorts the list

There are **four** collection data types in the Python programming language:

List is a collection which is ***ordered*** and ***changeable***. Allows ***duplicate*** members.

Tuple is a collection which is ***ordered*** and ***unchangeable***. Allows ***duplicate*** members.

Set is a collection which is ***unordered***, ***unchangeable****, and ***unindexed***. ***No duplicate*** members.

Dictionary is a collection which is ***ordered**** and ***changeable***. ***No duplicate*** members.

Example15 List

```
thislist = ["apple", "banana", "cherry", "apple", "cherry"]
print(thislist)
print(len(thislist)) #length of list
print(type(mylist)) #type of list

#data type
list1 = ["apple", "banana", "cherry"]
list2 = [1, 5, 7, 9, 33]
list3 = [True, False, False]
list4 = ["abc", 34, True, 40, "male"]
print(type(list1),type(list2),type(list3),type(list4))

#con
list5=list(("apple", "banana", "cherry"))
print(list5,type(list5))

#access item
thislist = ["apple", "banana", "cherry","orange", "kiwi", "melon",
"mango"]#index start from 0
print(thislist[1])
#negative index
print(thislist[-1])
#range
print(thislist[2:5])#start from index 2 and less than index 5 - 234
print(thislist[:5])#start from index 0 and less than index 5 - 01234
print(thislist[2:])#start from index 2 to end

#change
thislist = ["apple", "banana", "cherry", "orange", "kiwi", "mango"]
thislist[1:3]=["strawberry","watermelon"]
print(thislist)
thislist[1:2]=["durin","lime"]
print(thislist)
thislist[1:3]=["grape"]

#Append Items
thislist = ["apple", "banana", "cherry"]
thislist.append("orange")
print(thislist)

#Insert Items
thislist = ["apple", "banana", "cherry"]
```

```
thislist.insert(1, "orange")
print(thislist)

#Remove Specified Item
thislist = ["apple", "banana", "cherry"]
thislist.remove("banana")
print(thislist)

#Remove Specified Index
thislist = ["apple", "banana", "cherry"]
thislist.pop(1)
print(thislist)

thislist = ["apple", "banana", "cherry"]
thislist.pop()#removes the last item.
print(thislist)

#del keyword
thislist = ["apple", "banana", "cherry"]
del thislist[0]
print(thislist)

thislist = ["apple", "banana", "cherry"]
del thislist
#print(thislist)

#Clear the List
thislist = ["apple", "banana", "cherry"]
thislist.clear()
print(thislist)

#for loop
thislist = ["apple", "banana", "cherry"]
for x in thislist:
  print(x)

thislist = ["apple", "banana", "cherry"]
for i in range(len(thislist)):
  print(thislist[i])

#while loop
thislist = ["apple", "banana", "cherry"]
i = 0
```


```
while i < len(thislist):
  print(thislist[i])
  i = i + 1

thislist = ["apple", "banana", "cherry"]
[print(x) for x in thislist]

#sorting
thislist = ["orange", "mango", "kiwi", "pineapple", "banana"]
thislist.sort()
print(thislist)

thislist = ["orange", "mango", "kiwi", "pineapple", "banana"]
thislist.sort(reverse=True)
print(thislist)
```

Example16 Tuple

```
thistuple = ("apple", "banana", "cherry", "apple", "cherry")
print(thistuple)
print(len(thistuple))
print(type(thistuple))

#data type
tuple1 = ("apple", "banana", "cherry")
tuple2 = (1, 5, 7, 9, 3)
tuple3 = (True, False, False)
tuple4 = ("abc", 34, True, 40, "male")
print(type(tuple1),type(tuple2),type(tuple3),type(tuple4))

thistuple = tuple(("apple", "banana", "cherry")) # note the double
round-brackets
print(thistuple)

#slicing
thistuple = ("apple", "banana", "cherry", "orange", "kiwi", "melon",
"mango")
print(thistuple[1])
print(thistuple[-1])
print(thistuple[2:5])#234
print(thistuple[2:])
print(thistuple[:4])

#update
```

```
#add new item into tuple - tuple cannot change
x=("apple","banana","cherry")
y=list(x)#convert from tuple to list

y[1]="kiwi"
x=tuple(y)#convert from list to tuple
print(x)

#add
thistuple=("apple","banana","cherry")
y=list(thistuple)
y.append("orange")
thistuple=tuple(y)
print(thistuple)

thistuple = ("apple", "banana", "cherry")
y = ("orange",)
thistuple += y
print(thistuple)

#remove
thistuple = ("apple", "banana", "cherry")
y = list(thistuple)
y.remove("apple")
thistuple = tuple(y)
thistuple = ("apple", "banana", "cherry")
del thistuple
#print(thistuple)

#Loop
thistuple=("apple","banana","cherry")
for x in thistuple:
    print(x)

thistuple=("apple","banana","cherry")
for x in range(len(thistuple)):
    print(thistuple[x])

thistuple=("apple","banana","cherry")
i=0
while i<len(thistuple):
    print(thistuple[i])
    i=i+1
```


```
#Method
thistuple=(1,5,7,5,7,5,4,6,8,5)
x=thistuple.count(5)
print(x)

thistuple=(8,5,7,5,7,5,4,8,7,5)
x=thistuple.index(8)#index method finds the first occurrence of the
specified value.
print(x)
```

Example17 Set

```
#Set_unordered_unchangeable_notDuplicate
thisset={"apple","banana","cherry","apple"}
print(thisset)
print(type(thisset))
print(len(thisset))

thisset={"apple","banana","cherry",True,1,2}#True    and    1    are
considered the same value
print(thisset)

#AccessSet
thisset={"apple","banana","cherry"}
print(thisset)
for x in thisset:
    print(x)
print("banana" in thisset)

#add
thisset.add("orange")
print(thisset)

#update
thisset={"apple","banana","cherry"}
tropical={"kiwi","durin","mango"}
thisset.update(tropical)
print(thisset)

#Add any
thisset={"apple","banana","cherry"}
mylist=["papaya","mango"]
thisset.update(mylist)
```


```
print(thisset)

#remove
thisset={"apple","banana","cherry"}
thisset.remove("banana")
print(thisset)

thisset = {"apple", "banana", "cherry"}
thisset.discard("banana") #remove
print(thisset)

thisset = {"apple", "banana", "cherry"}
x=thisset.pop()#random
print(x)
print(thisset)

thisset = {"apple", "banana", "cherry"}
thisset.clear()
print(thisset)

thisset = {"apple", "banana", "cherry"}
del thisset#completely delete thisset
print(thisset)#error

#method
set1 = {"a", "b", "c"}
set2 = {1, 2, 3}
set3=set1.union(set2)
print(set3)

set1 = {"a", "b", "c"}
set2 = {1, 2, 3}
set1.update(set2)
print(set1)

x = {"apple", "banana", "cherry"}
y = {"google", "microsoft", "apple"}
x.intersection_update(y)
print(x)

x = {"apple", "banana", "cherry"}
y = {"google", "microsoft", "apple"}
z = x.intersection(y)
```


```
print(z)

x = {"apple", "banana", "cherry"}
y = {"google", "microsoft", "apple"}
x.symmetric_difference_update(y)
#The symmetric_difference_update() method will keep only the elements
that are NOT present in both sets.
print(x)

x = {"apple", "banana", "cherry", True}
y = {"google", 1, "apple", 2}
z = x.symmetric_difference(y)#True and 1 is considered the same value:
print(z)

x1 = {"apple", "banana", "cherry"}
y1 = {"google", "microsoft", "facebook"}
z1 = x1.isdisjoint(y1)
print(z1)

x2 = {"a", "b", "c"}
y2 = {"f", "e", "d", "c", "b", "a"}
z2 = x2.issubset(y2)
print(z2)

x3 = {"f", "e", "d", "c", "b", "a"}
y3 = {"a", "b", "c"}
z3 = x3.issuperset(y3)
print(z3)
```

Example18 Dictionary

```
#example18
thisdict={
"brand":"Suzuki",
"model":"Swift",
"year":2020
}
print(thisdict)
print(thisdict["brand"])
print(len(thisdict))
print(type(thisdict))

#constructor
thisdict2=dict(name="john",age=36,country="Norway")
print(thisdict2)

#method
thisdict={
"brand":"Suzuki",
"model":"Swift",
"year":2020
}
x=thisdict["model"]
print(x)

#get()
xx=thisdict.get("model")
print(xx)

#key()
car = {
"brand": "Ford",
"model": "Everest",
"year": 1964
}
y=car.keys()
print(y)
car["color"]="white"
print(y)

#values()
car = {
"brand": "Ford",
```


```
"model": "Everest",
"year": 1964
}

z=car.values()
print(z)
car["color"]="red"
print(z)

#item
car = {
"brand": "Honda",
"model": "Fit",
"year": 2000
}
a=car.items()
print(a)
car["year"]=2023
print(a)
car["color"]="blue"
print(a)

#update
thisdict={
"brand":"Honda",
"model":"fit",
"year":2010
}
thisdict["year"]=2023
print(thisdict)

#update
thisdict.update({"year":2025})
print(thisdict)

#Add New
car={
"brand":"Suzuki",
"model":"Swift",
"year":2010
}
car["color"]="red"
print(car)
```


```
#update new
car2={
"brand":"Suzuki",
"model":"Swift",
"year":2010
}
car2.update({"color":"silver"})
print(car2)

#remove
car={
"brand":"Suzuki",
"model":"Swift",
"year":2010
}
car.pop("model")
print(car)

#pop
car2={
"brand":"Suzuki",
"model":"Swift",
"year":2010
}
car2.popitem()
print(car2)

#clear
car={
"brand":"Suzuki",
"model":"Swift",
"year":2010
}
car.clear()
print(car)

#del
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
```


```
del thisdict["model"]
print(thisdict)

#del all
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
del thisdict
print(thisdict)#error
```

File Handling

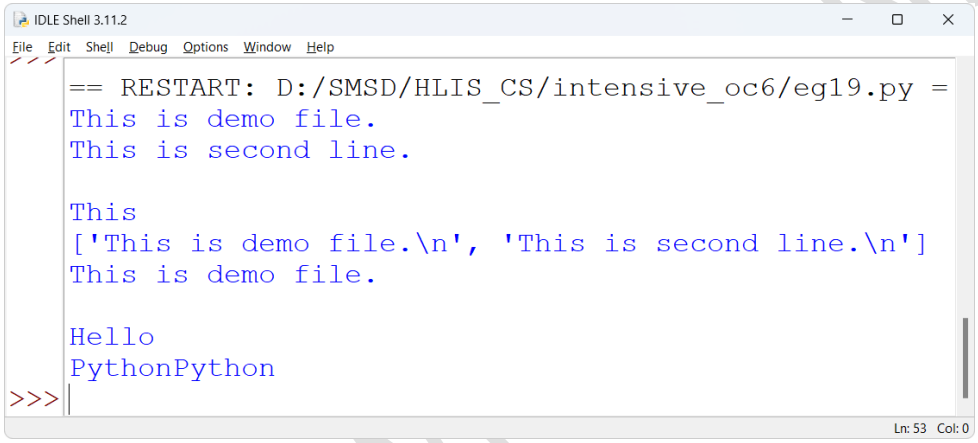
File handling is an important part of any web application.
Python has several functions for creating, reading, updating, and deleting files.
The key function for working with files in Python is the ***open()*** function.
The ***open()*** function takes ***two parameters***; **filename**, and **mode**.
There are ***four*** different methods (modes) for opening a file:
"r" - Read - Default value. Opens a file for reading, error if the file does not exist
"a" - Append - Opens a file for appending, creates the file if it does not exist
"w" - Write - Opens a file for writing, creates the file if it does not exist
"x" - Create - Creates the specified file, returns an error if the file exists
"a" - Append - will ***append*** to the end of the file
"w" - Write - will ***overwrite*** any existing content
To ***delete*** a file, you must import the ***OS*** module, and run its os.remove() function:

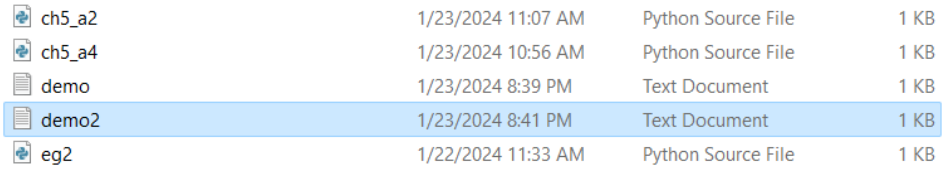
Example19 Read File ♬ 19

```
#example19_readfile
#read
f=open("demo.txt","r")
print(f.read())

f=open("demo.txt","r")
print(f.read(5))

f=open("demo.txt","r")
print(f.readlines())

f=open("demo.txt","r")
print(f.readline())

#write
ff=open("demo2.txt","w")
ff.write("Hello")
ff.close()

f=open("demo2.txt","r")
print(f.read())
```

```
#append
f3=open("demo3.txt","a")
f3.write("Python3")
f3.close()

f3r=open("demo3.txt","r")
print(f3r.read())

#read using loop
myfile = open("demo.txt", "r")
for line in myfile:
    print(line)
myfile.close()
```

Result





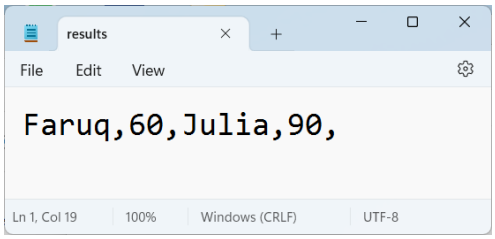
♬ 20

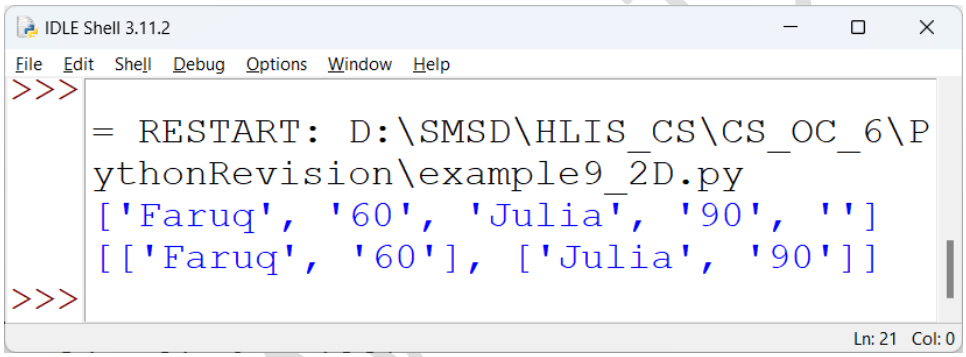
Example20 Two Dimension

```
#example20_2D – ch9 activity 25
#write
Slist=[['Faruq',60],['Julia',90]]
myFile=open("results.txt","w")
for x in range(len(Slist)):
    myFile.write(Slist[x][0]+',')
    myFile.write(str(Slist[x][1])+',')
myFile.close()

#read
Blist=[]
```

```
myFile=open("results.txt","r")
Blist=myFile.read().split(',')
myFile.close()
print(Blist)

newList=[]
for index in range(0,len(Blist)-1,2):
    newList.append([Blist[index],Blist[index+1]])
print(newList)
```

Result





Sorting and Searching

Bubble Sort

```
#example71_bubblesort
def bubble_sort(arr):
    swapped=False
    for n in range(len(arr)-1,0,-1):#(8-1,0,-1)=(7,0,-1)
        for i in range(n):
            if arr[i]>arr[i+1]:
                swapped=True
                arr[i],arr[i+1]=arr[i+1],arr[i]

        if not swapped:#not true = flase - no swap
            return

arr=[39,12,28,85,72,10,2,18]
print("Unsorted list is", arr)
bubble_sort(arr)
print("Sorted list is", arr)
```

Result

```
>>> %Run bubble_sort.py
 Unsorted list is [39, 12, 28, 85, 72, 10, 2, 18]
 Sorted list is [2, 10, 12, 18, 28, 39, 72, 85]
```

Merge Sort

```
def merge_sort(arr):
    if len(arr) <= 1:
        return arr

    # Split the array into two halves
    mid = len(arr) // 2
    left_half = arr[:mid]
    right_half = arr[mid:]

    # Recursive call to sort each half
    left_half = merge_sort(left_half)
    right_half = merge_sort(right_half)

    # Merge the sorted halves
    return merge(left_half, right_half)

def merge(left, right):
    merged = []
    left_index = 0
    right_index = 0

    # Compare elements from both arrays and merge them
    while left_index < len(left) and right_index < len(right):
        if left[left_index] < right[right_index]:
            merged.append(left[left_index])
            left_index += 1
        else:
            merged.append(right[right_index])
            right_index += 1
    # Append the remaining elements from both arrays (if any)
    merged += left[left_index:]
    merged += right[right_index:]

    return merged
my_list = [38, 27, 43, 3, 9, 82, 10]
sorted_list = merge_sort(my_list)
print(sorted_list)
```

Result

```
>>> %Run merge_sort_simple.py
 [3, 9, 10, 27, 38, 43, 82]
```


Linear Search

```
def linear_search(arr,key):
    for i in range(0,len(arr)):
        if arr[i]==key:
            return i
    return -1

arr=[1,3,5,4,7,9]
key=int(input("Enter search number:"))
result=linear_search(arr,key)
if result==-1:
    print("search no found")
else:
    print("search item found",result)
```

Result

```
>>> %Run linear_search.py
 Enter search number:4
 search item found 3
```

Binary Search

```
def binary_search(arr,key):
    low=0
    high=len(arr)-1 #6-1=5
    mid=0
    while low<=high:
        mid=(low+high)//2
        if arr[mid]<key:
            low=mid+1
        elif arr[mid]>key:
            high=mid-1
        else:
            return mid
    return -1

arr=[1,3,4,5,7,9]
key=int(input("Enter search number:"))
result=binary_search(arr,key)
if result==-1:
    print("search no found")
else:
    print("search item found",result)
```


Result

```
>>> %Run binary_searcj.py
 Enter search number:4
 search item found 2
```

References

https://www.w3school.com
https://www.onlineexambuilder.com/index.php?r=exam/quiz&language=en&PHPSESSID=new