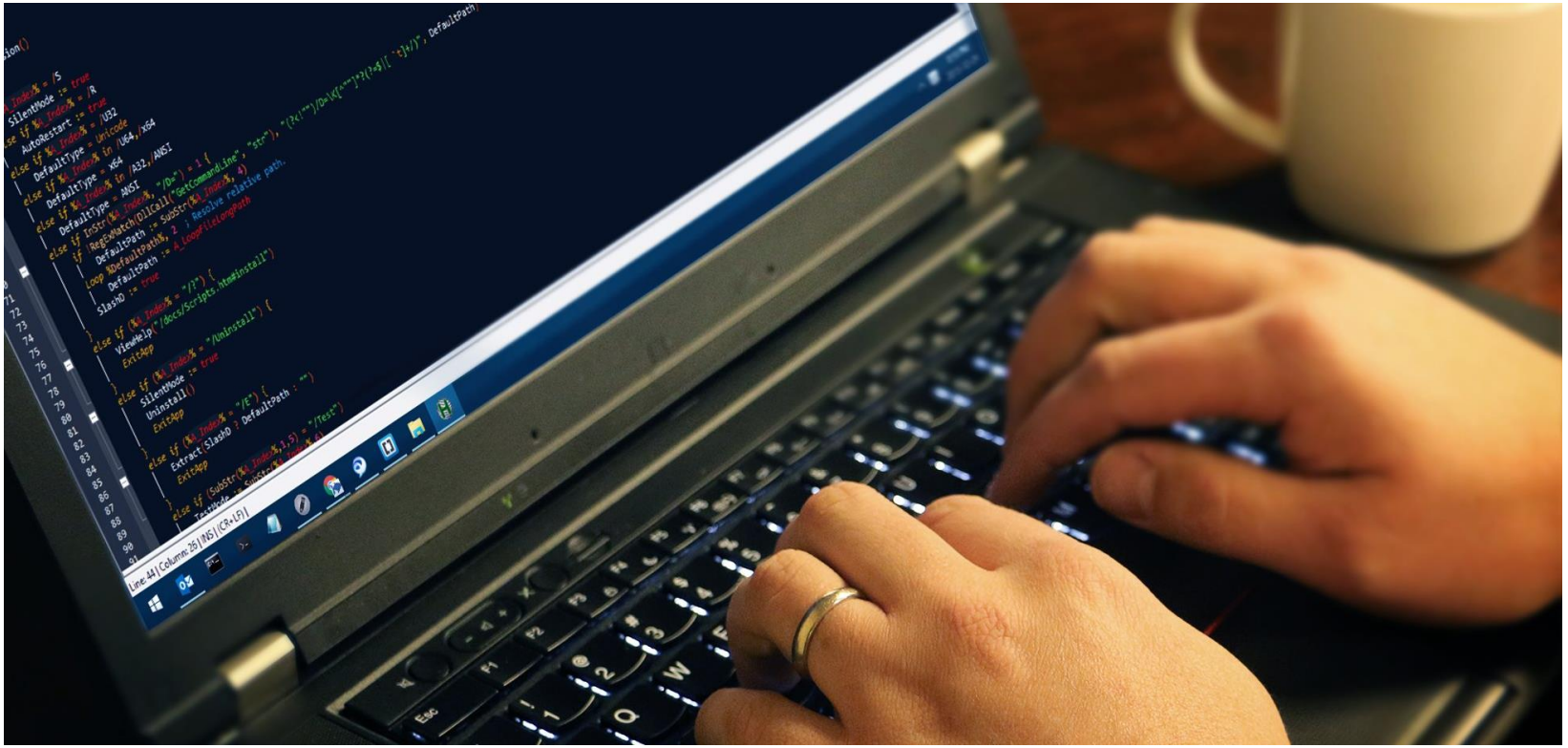


Course summary and review

Introduction to Programming: Python

Ashok Kumar Veerasamy
Xiaobo Bi
Abdur Rahman

1. First Program

```
# My first program
print("Hello Python")
# print("2+3+5="+(2+3+5))
print("2+3+5="+str(2+3+5)) # 2+3+5=10
print("2+3+5=", 2+3+5)  # 2+3+5= 10

# hint: sometimes may be you don't need to delete your statements,

you only need to change it to comment.
```

2. Variables and Expression

- **Data type (basic type)**

Integer: 0, 4,8923423,-232 , -45

Float: 23.45, 0.2342214, -1.00, 0.23, -0.342342

String: "Lut" , "name\tmark" # enclosed by " " , ' ' , …

Boolean: **True**, **False**

- **Different type have different storage and operation form.**

23 is different from "23" # 23 -> 0001 0101 binary form, "23" -> ASCII

"ABC" > "z" ……

- **Variable naming rule**

In python, variable names can use letters, digits, and the underscore symbol, but they can't star with a digit.

Case sensitive. varC is different varc

Can't use system keyword.

Arithmetic operators and expressions

Symbol	Operator	Example	Result
-	Negation	-5	-5
+	Addition	11 + 3.1	14.1
-	Subtraction	5 - 19	-14
*	Multiplication	8.5 * 4	34.0
/	Division	11 / 2	5.5
//	Integer Division	11 // 2	5
%	Remainder	8.5 % 3.5	1.5
**	Exponentiation	2 ** 5	32

Table 1—Arithmetic Operators

Waring:
= is not Equality in Python

Precedence	Operator	Operation
Highest	**	Exponentiation
	-	Negation
	*, /, //, %	Multiplication, division, integer division, and remainder
Lowest	+, -	Addition and subtraction

Table 2—Arithmetic Operators Listed by Precedence from Highest to Lowest

Type-convertion

- type()

```
>>> type(input("user input is:"))
 user input is:891
<class 'str'>
```

```
>>> type([1,2])
<class 'list'>
```

- int()

```
5 print(int(2.6))
6 print("256")
7 print(int("2.71828"))
```

```
2
256
```

```
  print(int("2.71828"))
ValueError: invalid literal for int() with base 10: '2.71828'
```

```
>>> int(input("please input an integer"))
```

- float()

```
10 print(float(4))
11 print(float(2.5))
12 print(float("2.55"))
```

```
4.0
2.5
2.55
```

```
>>> float(input("please input an integer"))
```

- str()

```
1 print("2+3+5="+(2+3+5))
```

```
  print("2+3+5="+(2+3+5))
TypeError: can only concatenate str (not "int") to str
```

```
1 # print("2+3+5="+(2+3+5))
2 print("2+3+5="+str(2+3+5))
3 print("2+3+5=", 2+3+5)
```

```
2+3+5=10
2+3+5= 10
```

String handle

- String handing in Pytyon
  - print(“name\tmark”)
  - print(“name\nmark”)
  - print( ) *# a new line*
  - print(“he said \“victory in sight.\””)
- Index of string (0……n-1) **[ ]**
  - str1= “hi hello”
  - n= len(str1) *# the last character in string is str1[n-1]*
  - str1[3:] str[:3]
  - str[0:n-1]

String function

```
str="aAbBcC"
str.index()
str.find()
str.capitalize()
str.count()
str.upper()
str.lower()
str.replace()
```

```
>>> str="aAbBcC"
>>> str.index("a")
0
>>> str.index("A")
1
```

```
>>> str.find("a")
0
>>> str.find("D")
-1
```

```
>>> str.index("D")
  Traceback (most recent call last):
    File "<pyshell>", line 1, in <module>
  ValueError: substring not found
```

```
>>> str
'aAbBcC'
>>> str.capitalize()
'Aabbcc'
```

```
>>> str.count("a")
1
>>> str.lower().count("a")
2
```

```
>>> str.replace("a","X")
'XAbBcC'
>>> str
'aAbBcC'
>>> str.replace("d","E")
'aAbBcC'
```

import datetime

```
1 from datetime import datetime
2 print(datetime.now())
3 print(datetime.now().year)
4 print(datetime.now().month)
5 print(datetime.now().day)
```

```
2021-12-11 17:51:45.813246
2021
12
11
```

```
1 import datetime
2 print(datetime.date.today())
3 print(datetime.date.today().year)
4 print(datetime.date.today().month)
5 print(datetime.date.today().day)
```

```
2021-12-11
2021
12
11
```

3.Selection / Conditional statements

Bool Type : Ture and False

Relational Operators

Symbol	Operation
>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to
==	Equal to
!=	Not equal to

Table 6—Relational and Equality Operators

Boolean Operators

not, and, or

Enter a year to determine whether it is a normal year or a leap year.

```
year=int(input("Please input a year:"))
if (year%4==0 and year%100!=0) or (year%400==0):
    print(year," is leap year.")
else:
    print(year," is normal year.")
```

```
#Enter an integer and judge whether it is even or odd ?
x = int(input ("Please input an integer:"))
if x % 2 == 0:
    print(x, 'is even')
else:
    print(x, 'is odd')
```

3.Selection / Conditional statements

```python
# area of the circle
import numpy
r=int(input("Enter the radius value:"))
if r>=1:
    CirArea=numpy.pi*(r**2)
    print("The area of circle is:", CirArea)
else:
    print("Radius value is zero of below")
```

```python
num1=int(input("Please input 1st integer:"))
num2=int(input("Please input 2nd integer:"))
if num1>num2: print("Decending order:",num1,num2)
else: print("Decending order:",num2,num1)
```

```python
if number%2 == 0:
    print (number, "is even")
print (number, "is odd")
```

```python
if number%2 == 0:
    print (number, "is even")
else:
    print (number, "is odd")
```

3.Selection / Conditional statements

Enter the length of the three sides of the triangle to determine whether a triangle can be formed, and then calculate the area.

```
import math
a=float(input("1st side length: "))
b=float(input("2nd side length: "))
c=float(input("3rd side length: "))
if a>0 and b>0 and c>0:
    if (a+b)>c and (a+c)>b and (b+c)>a:
        p=(a+b+c)/2
        s=math.sqrt(p*(p-a)*(p-a)*(p-c))
        print("The area of tringle is:", s)
    else:
        print("Can't form a tringle")
else:
    print("Input length must be greater than 0.")
```

3.Selection / Conditional statements

**Table 2: Grade calculation for CT60A0203

Scores / Points in %	Grade
0 to 49	0
50 to 59	1
60 to 69	2
70 to 79	3
80 to 92	4
93 to 100	5

```
1  score=float(input("Please input the score:"))
2  if 93<=score<=100:
3      print("The corresponding grade is:",5)
4  elif 80<=score<=92:
5      print("The corresponding grade is:",4)
6  elif 70<=score<=79:
7      print("The corresponding grade is:",3)
8  elif 60<=scoer<=69:
9      print("The corresponding grade is:",2)
10 elif 50<=score<=59:
11     print("The corresponding grade is:",1)
12 else:
13     print("The corresponding grade is:",0)
```

3.Selection / Conditional statements menu based coding

- Write a program to prompt the user to enter any two numbers as input and perform the arithmetic calculations based on user choice.

example_if3.py | example_if4.py

```
1  #Simple menu based coding
2  print ("Welcome to arithmetic calculations")
3  print (" 1. Addition\n","2. Subtraction\n", "3. Multiplication\n","4. Division\n","5. Exit\n")
4  option = int(input("Select your option [from 1 - 5]"))
5
6  if  option<1 or option>=5:
7      print ("Bye Bye")
8  else:
9      a = float(input("Enter the first number: "))
10     b = float(input("Enter the second number: "))
11
12 if option == 1:
13     print("a+b =",a+b)
14
15 if option == 2:
16     print("a-b =",a-b)
17
18 if option == 3:
19     print("a*b =",a*b)
20
21 if option == 4:
22     print("a/b =",a/b)
```

- 1- Addition
- 2- Subtraction
- 3 –Multiplication
- 4. Division
- 5. Exit

Shell

```
Welcome to arithmetic calculations
 1. Addition
 2. Subtraction
 3. Multiplication
 4. Division
 5. Exit

Select your option [from 1 - 5]4
Enter the first number: 50
Enter the second number: 5
```

4. Iteration and iterative staements

- *for* loop # loop over a range of numbers

```
for «variable» in «list»:
    «block»
```

- *while* loop # looping until a condition is reached

```
while «expression»:
    «block»
```

```
1  # printing odd numbers using for loop
2  for i in range (1, 10):
3    if i%2 == 1:
4        print(i)
5
```

Shell

```
Python 3.7.9 (bundled)
>>> %Run ex8_for.py

  1
  3
  5
  7
  9
```

```
7  # printing odd numbers using while loop
8
9  x = 1
10 while x<10:
11     print (x)
12     x = x+2
```

Shell

```
Python 3.7.9 (bundled)
>>> %Run ex8_for.py

  1
  3
  5
  7
  9
```

4. Nested loops

Nested loops: It consists of an outer loop and one or more inner loops. Each time the outer loop is repeated, the inner loops are reentered and started new.

```
1
12
123
1234
```

```
for i in range(1,5):
    for j in range(1,i+1):
        print(j,end="")
    print("\n")
```

```
for i in range(1,5):
    for j in range(1,i+1):
        print(j,end="")
    print()
```

```
1
12
123
1234
```

```
for i in range(1,10):
    for j in range(1,i+1):
        print(str(i)+"*"+str(j)+"="+str(i*j)+" ",end="")
    print()
```

```
1*1=1
2*1=2 2*2=4
3*1=3 3*2=6 3*3=9
4*1=4 4*2=8 4*3=12 4*4=16
5*1=5 5*2=10 5*3=15 5*4=20 5*5=25
6*1=6 6*2=12 6*3=18 6*4=24 6*5=30 6*6=36
7*1=7 7*2=14 7*3=21 7*4=28 7*5=35 7*6=42 7*7=49
8*1=8 8*2=16 8*3=24 8*4=32 8*5=40 8*6=48 8*7=56 8*8=64
9*1=9 9*2=18 9*3=27 9*4=36 9*5=45 9*6=54 9*7=63 9*8=72 9*9=81
```

4. Controlling Loops Using break and continue

Write a program that accpets any integer as input and print whether it is a prime number or not.

```python
#use else-segement
n=int(input("Enter an integer(>=2):"))
for i in range(2,n):
    if n%i==0:
        print(n,"is not a prime number.")
        break
else:
    print(n,"is a prime number.")
```

```python
# use flag
n=int(input("Enter an integer(>=2):"))
flag=True
for i in range(2,n):
    if n% i==0:
        flag=False
        break
if n==2 or flag==True:
    print(n,"is a prime number.")
else:
    print(n,"is not a prime number.")
```

```python
#use break
n=int(input("Enter an integer(>=2):"))
for i in range(2,n):
    if n%i==0:break
if n==2:
    print(n,"is a prime number.")
elif i==n-1:
    print(n,"is a prime number.")
else:
    print(n,"is not a prime number.")
```

5. Procedure / function definition and calling

- ***format***

```
def «function_name»(«parameters»):
    «block»
```

```
>>> def convert_to_celsius(fahrenheit):
...     return (fahrenheit - 32) * 5 / 9
...
>>> convert_to_celsius(80)
26.666666666666668
```

```
#BMI calculatioin function/procedure
def BMI(w,h):
    bmi = (w*0.45359237)/ ((h*0.0254)**2)
    print("your BMI is:",bmi)


#main program
w = float(input("Enter your weight in pounds:"))
h = float(input("Enter your height in inches:"))
BMI(w,h)
```

```
def BMI1(w,h):
    bmi = (w*0.45359237)/ ((h*0.0254)**2)
    return bmi


#main program
w = float(input("Enter your weight in pounds:"))
h = float(input("Enter your height in inches:"))
b = BMI1(w,h) # or print(BMI1(w,h))
print("your calculated BMI is:",b)
```

5. Procedure / function definition and calling

- ***example (in quiz)***

Please note that you only need to submit the statements for defining sub progress. Do not submit program or statements that are not required by the topic.

File "LibRec.txt" is the library borrowing records, the information include student ID, name, borrowing date and degree kinds. Here is the preview of part of the file (not include the field names of the first row).

Borrowing

ID	Name	Date	Degree
210001	Vivian	2021-02-02	ph. D.
210002	Nancy	2021-06-01	MD
210003	Wilson	2021-8-11	MD
210004	James	2021-08-05	BD
210005	Bruce	2021-05-07	BD
210006	Taylor	2021-04-10	ph. D.
210007	Kelly	2021-03-21	ph. D.
210008	Lane	2021-07-05	ph. D

Degree kind means student for bachelor's degree (BD), student for master's degree (MD) and student for doctor's degree (ph.D.). The free borrowing time duration of students for BD is 2 months, students for MD is 3 months and students for ph.D. is 4 months. For example, if the borrowing date is 2021-09-09 of a student for BD, then due date for return book is 2021-11-09; If he/she is a student for MD, the due date for return book is 2021-12-09; If he/she is a student for ph.D. the due date for return book is 2022-01-09. If the book is returned overdue, 0.1 euro will be charged for each overdue day.

(1) Define a sub function **DueDate(*Borrowingdate*, *Degree*)**, according to borrowing date and student degree to calculate the due date for return book and **return** the due date. The type of arguments (***Borrowingdate*** and ***Degree***) and return value is string. For example, if we call it as follows:

```
def DueDate(borrDate, Degree):
    x=borrDate.split("-")
    year=int(x[0])
    month=int(x[1])
    day=int(x[2])
    if Degree=="BD":
        month+=2
    elif Degree=="MD":
        month+=3
    elif Degree=="ph.D.":
        month+=4
    if month>12:
        month=month%12
        year+=1
    return str(year)+"-"+str(month)+"-"+str(day)
```

5. Procedure / function definition and calling

- ***example (in quiz)***

```
def DueDate(borrDate, Degree):
    x=borrDate.split("-")
    year=int(x[0])
    month=int(x[1])
    day=int(x[2])
    if Degree=="BD":
        month+=2
    elif Degree=="MD":
        month+=3
    elif Degree=="ph.D.":
        month+=4
    if month>12:
        month=month%12
        year+=1
    return str(year)+"-"+str(month)+"-"+str(day)
```

```
def OverDueCharge(duestr):
    duedate=datetime.date(*map(int,duestr.split('-')))
    overdays=(datetime.date.today()-duedate).days
    if overdays>0:
        fine=overdays*0.1
    else:
        fine=0
    return fine
```

```
>>> Xdue=DueDate("2021-11-21","ph.D.")
>>> print(OverDueCharge(Xdue))
0
```

6. File I/O processing in Python

- ***read data from file***

```
>>> with open("LibRec.txt","r") as f1:
        for rec in f1:
            print(rec)
```

```
f1=open("LibRec.txt","r")
for rec in f1:
    print(rec,type(rec))  # get the type of rec
    recitem=rec.split(",")
    print(recitem, type(recitem))
f1.close()
```

```
210001,Vivian,2021-08-02,ph.D.
 <class 'str'>
['210001', 'Vivian', '2021-08-02', 'ph.D.\n'] <class 'list'>
210002,Nancy,2021-09-01,MD
 <class 'str'>
['210002', 'Nancy', '2021-09-01', 'MD\n'] <class 'list'>
```

6. File I/O processing in Python

- ***read data from file***

```
f1=open("LibRec.txt","r")
content=f1.read()
print(content, type(content))
for rec in content:
    print(rec,type(rec))
f1.close()
```

```
210001,Vivian,2021-08-02,ph.D.
210002,Nancy,2021-09-01,MD
210003,Wilson,2021-10-11,MD
210004,James,2021-11-05,BD
210005,Bruce,2021-08-07,BD
210006,Taylor,2021-09-10,ph.D.
 <class 'str'>
2 <class 'str'>
1 <class 'str'>
0 <class 'str'>
0 <class 'str'>
0 <class 'str'>
1 <class 'str'>
, <class 'str'>
V <class 'str'>
i <class 'str'>
v <class 'str'>
i <class 'str'>
```

```
f1=open("LibRec.txt","r")
content=f1.readlines()
print(content, type(content))
for rec in content:
    print(rec,type(rec))
f1.close()
```

```
['210001,Vivian,2021-08-02,ph.D.\n', '210002,Nancy,2021-09-01,MD\n', '21
0003,Wilson,2021-10-11,MD\n', '210004,James,2021-11-05,BD\n', '210005,Br
uce,2021-08-07,BD\n', '210006,Taylor,2021-09-10,ph.D.\n'] <class 'list'>
210001,Vivian,2021-08-02,ph.D.
 <class 'str'>
```

6. File I/O processing in Python

- ***write data to file***

```
f1=open("numRec.txt","w")
for i in range(6):
    f1.write(str(i))
f1.close()
```

012345

```
f1=open("numRec.txt","w")
for i in range(6):
    f1.write(str(i)+"\n")
f1.close()
```

0
1
2
3
4
5

students.txt

```
s101,78,3
s102,0,0
s103,58,1
s104,64,2
s105,46,0
s106,93,5
s107,85,4
```

```
f1 = open("students.txt") # read student data
f2 = open("revisedStudents.txt","w") # write into new file
for student in f1: # loop to read data -> end of file
    s = student.split(",") #split data separated by comma
    if int(s[2])>0: #the index of grade in a row is 2
        f2.write(s[0]+","+s[1]+","+s[2].strip()+",P"+"\n")
    else:
        f2.write(s[0]+","+s[1]+","+s[2].strip()+",F"+"\n")
        #strip cut the "\n"
f1.close()
f2.close()
f3 = open("revisedStudents.txt") # read data
print(f3.read())
```

revisedStudents.txt

```
s101,78,3,P
s102,0,0,F
s103,58,1,P
s104,64,2,P
s105,46,0,F
s106,93,5,P
s107,85,4,P
```

*7. Data structures in Python **List** and Turple*

- **List**
  - A **dynamic** sequence of zero or more consecutive items.
  - Content of lists can be **changed**, **re-arranged** during execution.
  - **Index** of list elements

data:	[	“atom‘	“cat”	“deep”	“moron”	“quick”	“reason”	“Zoo”	]
Index:		0	1	2	3	4	5	6	

print(data[0]) # return a value “atom”

print(data[1:3]) # return a list [“cat”,“deep”,“moron”]

print(data[:]) # return the list data

- **len(data)** # find the length of list

```
f1=open("LibRec.txt","r")
content=f1.readlines()
for i in range(len(content)):
    print(content[i])
f1.close()
```

exp List combining and multiplying and more..

```
#combining lists
a = [1, 2, 3]
b = [6, 7, 8]
a = a + b
print (a)  # returns [1, 2, 3, 6, 7, 8]
```

```
# What is the output of the following code?
b = [6, 7, 8]
c = b * 3
print (c) # returns [6, 7, 8, 6, 7, 8, 6, 7, 8]
print (c.count(6)) # returns 3
```

```
#Modifiying  the value of a certain index
myList = [1, 2, 3, 4, 5, 7]
myList[2] =  36
print (myList)     # would print  [1, 2, 36, 4, 5, 7]
```

Adding, inserting and deleting values in the list

```
x = [1, 3, 4]
x.append(5)  # 5 will be added at the end of the list
print (x) # would print [1, 3, 4, 5]
```


```
x = [1, 3, 4]
x.append(5)  # 5 will be added at the end of the list
print (x) # would print [1, 3, 4, 5]
```


```
x.insert (1, 2) # insert 2 at index 1
print (x)  # would print [1, 2, 3, 4, 5]
x.insert (9,46)
print(x) # [1, 2, 3, 4, 5, 46]
```

```
# Item in list - in operator
a =  [1, 2, 3, 4, 5]
print (7 in a) # returns False
print  (3 in a) # returns True
```

Adding, inserting and deleting values in the list

```
x = [1, 2, 3, 4, 5, 2]
x.remove (2) # remove value 2 from the list not by index
print (x)  # would print [1, 3, 4, 5, 2]

# so to delete a value if it exists in the List
def deleteItem (the_list, item):
    for x in the_list:
        if item in the_list:  # using in operator here
            the_list.remove(item)
x = [1, 2, 3, 4, 5, 2]
deleteItem(x,2) #
print(x) # [1, 3, 4, 5]
```

Sorting elements of list

```
1  List1 = [1,4,-23.9,89]
2  List2 = ["asd","akb@234","45"]
3
4  print(sorted(List1)) # printing sorted form but no change in the list
5  print(List1) # print as it is (line 1)
6  List2.sort() # sort the values of List2 - gets changed
7  print(List2) #
8
9  #descending order
10 print ("sorting elements in descending order")
11 print(sorted(List1,reverse=True))
12 print (List1)
13 List2.sort(reverse=True)
14 print(List2)
```

Shell

```
Python 3.7.9 (bundled)
>>> %Run examplesorty.py

  [-23.9, 1, 4, 89]
  [1, 4, -23.9, 89]
  ['45', 'akb@234', 'asd']
  sorting elements in descending order
  [89, 4, 1, -23.9]
  [1, 4, -23.9, 89]
  ['asd', 'akb@234', '45']

>>>
```

7. Data structures in Python List and **Tuple**

- ***Tuple***
  - It is a collection of ordered items,
  - Tuple is like Lists [ ] but values for tuple are listed within ( ) and items/objects in the tuple are not changeable.

```
tuple1=(2,3,4,5,-6)
tuple2=("orange","apple",-45,24.5)
tuple3=tuple1+tuple2
print(tuple3)
```

```
(2, 3, 4, 5, -6, 'orange', 'apple', -45, 24.5)
```

```
tuple1=(2,3,4,5,-6)
list1=[2,3,4,5,-6]

list1[3]=100
print(list1)
```

```
[2, 3, 4, 100, -6]
```

```
tuple1[3]=100
print(tuple1)
```

```
    tuple1[3]=100
TypeError: 'tuple' object does not support
item assignment
```

Tuple Accessing and deleting tuple elements

```
tup1=(45,89,34,-20,23,56)
print(tup1[2])  # 34
print(len(tup1)) # 6
print(min(tup1)) # -20
print(max(tup1)) # 89
print(tup1[1:4]) # (89, 34, -20)
print(tup1[:]) #(45, 89, 34, -20, 23, 56)
print(tup1[len(tup1)-1]) # last item in tuple 56
```

del( ) Removing individual tuple elements is not possible but deleting an entire tuple is possible.

```
tup1=(45,89,34,-20,23,56)
print(tup1) # (45, 89, 34, -20, 23, 56)
del(tup1)
print(tup1)
```

```
    print(tup1)
NameError: name 'tup1' is not defined
```

Accessing and deleting tuple elements

- Accessing elements of tuple

```
tup1=(45,-20,23,56)
for i in tup1:
    if i>0:
        print(i)
```

```
for i in range(len(tup1)):
    if tup1[i]>0:
        print(tup1[i])
```

```
45
23
56
```

- Deleting elements of tuple

```
tup1=(45,-20,23,56)
list1=list(tup1) # convert tuple as list
for i in list1:
    if i<0: # removing negative values
        list1.remove(i)
tup1=tuple(list1) # reassigning tup1 from list1
print(tup1) # (45, 23, 56)
```

- Sort the elements of tuple

```
tup1 = (16,41, -5.6, 8.912)
# displays in sorted form but tup1 order have no changes.
print(sorted(tup1)) # Ascending order.
print(sorted(tup1, reverse=True)) #Descending order.
```

Type of return value is: List

```
[-5.6, 8.912, 16, 41]
[41, 16, 8.912, -5.6]
```

Accessing and deleting tuple elements

- Accessing elements of tuple

```
tup1=(45,-20,23,56)
for i in tup1:
    if i>0:
        print(i)
```

```
for i in range(len(tup1)):
    if tup1[i]>0:
        print(tup1[i])
```

```
45
23
56
```

- Deleting elements of tuple

```
tup1=(45,-20,23,56)
list1=list(tup1) # convert tuple as list
for i in list1:
    if i<0: # removing negative values
        list1.remove(i)
tup1=tuple(list1) # reassigning tup1 from list1
print(tup1) # (45, 23, 56)
```

- Sort the elements of tuple

```
tup1 = (16,41, -5.6, 8.912)
```

```
>>> tup1.sort()
 Traceback (most recent call last):
   File "<pyshell>", line 1, in <module>
 AttributeError: 'tuple' object has no attribute 'sort'
```

Dictionary

- Format of dictionary
  - ***{key1:value1, key2:value2, ……..keyn: valuen]***

- Dictionary elements are defined inside **{ }**. Names of key must be unique, but values can be duplicated. That is , Dictionaries cannot have two values with the same key
- What will happen if we use the name of the key more than once?

```
scores={"x001":90, "x002":80, "x003":85,
        "x004":95, "x005":75, "x006":90, "x001":95}
print(scores)
```

```
{'x001': 95, 'x002': 80, 'x003': 85, 'x004': 95, 'x005': 75, 'x006': 90}
```

Dictionary

- Format of dictionary
  - ***{key1:value1, key2:value2, ……..keyn: valuen]***
- Dictionary elements are defined inside **{ }**. Names of key must be unique, but values can be duplicated. That is , Dictionaries cannot have two values with the same key
- What will happen if we use the name of the key more than once?

- The How to define a dictionary
  - ***StuSco={ }***
  - ***StuSco=dic( )***
- **Add element to dictionay (example)**
  - StuSco[**key]= value # samekey :update different key: add**

Access the elements of ditionary

function *.key( ) .values( ) .items( )* of a dictionary

```
scores={"x001":90, "x002":80, "x003":85,
        "x004":95, "x005":75, "x006":90}
print(scores)
print(scores.keys())
print(scores.values())
print(scores.items())
print("---------------")
for key, val in scores.items():
    if val>=90:
        print(key,val)
```

```
{'x001': 90, 'x002': 80, 'x003': 85, 'x004': 95, 'x005': 75, 'x006': 90}
dict_keys(['x001', 'x002', 'x003', 'x004', 'x005', 'x006'])
dict_values([90, 80, 85, 95, 75, 90])
dict_items([('x001', 90), ('x002', 80), ('x003', 85), ('x004', 95), ('x005', 75),
('x006', 90)])
---------------
x001 90
x004 95
x006 90
```

Access the elements of dictionary

```
Quizscores = {"x123" : 17, # here x123 is a key and 17 is a value..
              "x124" : 54,
              "x125" : 62,
              "x126" : 75,
              "x129" : 92,
              "x127" : 62}

#iterating
for k in Quizscores: # iterating
    print (k, Quizscores[k]) # here key is a index
print("----------------------")
#iterating and print the key that in index 3
print([key for key in Quizscores.keys()][3])
print("----------------------")
#similarly print value that in index  4
print([value for value in Quizscores.values()][4])
print("----------------------")
#how about printing both fetching from specific index key and value
print(([key for key in Quizscores.keys()][2], [value for value in Quizscores.values()][2]))
print("----------------------")
#Is it possible to fetch different index and value. Why not?
print(([key for key in Quizscores.keys()][2], [value for value in Quizscores.values()][1]))
```

```
x123 17
x124 54
x125 62
x126 75
x129 92
x127 62
----------------------
x126
----------------------
92
----------------------
('x125', 62)
----------------------
('x125', 54)
```

update, and delete items of dictionary

```
scores={"x001":90, "x002":80}
x={"x003":75,"x004":85}
scores.update(x)
print(scores) {'x001': 90, 'x002': 80, 'x003': 75, 'x004': 85}
x={"x001":95,"x005":80}
scores.update(x)
print(scores) {'x001': 95, 'x002': 80, 'x003': 75, 'x004': 85, 'x005': 80}
```

```
del scores["x005"]
print(scores)  {'x001': 95, 'x002': 80, 'x003': 75, 'x004': 85}
x=scores.pop("x004")
print(scores,x) {'x001': 95, 'x002': 80, 'x003': 75} 85
x=scores.popitem()
print(scores,x) {'x001': 95, 'x002': 80} ('x003', 75)
del scores
print(scores)     print(scores)
NameError: name 'scores' is not defined
```

8. sets & set theory

It is used to store multiple items in a single variable.
A set is a collection which is **unordered**, **unchangeable**, and **unindexed + no duplicates**.
Can be used for get rid of the duplicate values in a sequence.
The elements for set is defined within { }

- set operations and operators

Method Call	Operator
set1.difference(set2)	set1 - set2
set1.intersection(set2)	set1 & set2
set1.issubset(set2)	set1 <= set2
set1.issuperset(set2)	set1 >= set2
set1.union(set2)	set1 | set2
set1.symmetric_difference(set2)	set1 ^ set2

add, update, and delete elements of set

```
Quiz1 = {"Ashok","Kamal","Liu","Chen","Lev","Ajay","Ren","Wu","Zhao"}
Quiz1.add("Andrei") # adding/appending element
print(Quiz1)
print("**************")

#adding another set elements
Quiz1_deferred = {"Shu","Gao","Wang"}
Quiz1.update(Quiz1_deferred)
print(Quiz1)

print("**************")
name = input("Enter the name to be removed from Quiz1:")

if name in Quiz1:
    Quiz1.remove(name)

print("New list after removal:",Quiz1)
```

add/append

Update

Remove

add, update, and delete elements of set

Method	Description
S.add(v)	Adds item v to a set S—this has no effect if v is already in S
S.clear()	Removes all items from set S
S.difference(other)	Returns a set with items that occur in set S but not in set other
S.intersection(other)	Returns a set with items that occur both in sets S and other
S.issubset(other)	Returns True if and only if all of set S's items are also in set other
S.issuperset(other)	Returns True if and only if set S contains all of set other's items
S.remove(v)	Removes item v from set S
S.symmetric_difference(other)	Returns a set with items that are in exactly one of sets S and other—any items that are in both sets are *not* included in the result
S.union(other)	Returns a set with items that are either in set S or other (or in both)

List[ ]_ Tuple( )_ Set{ }_ Dictionary{ }

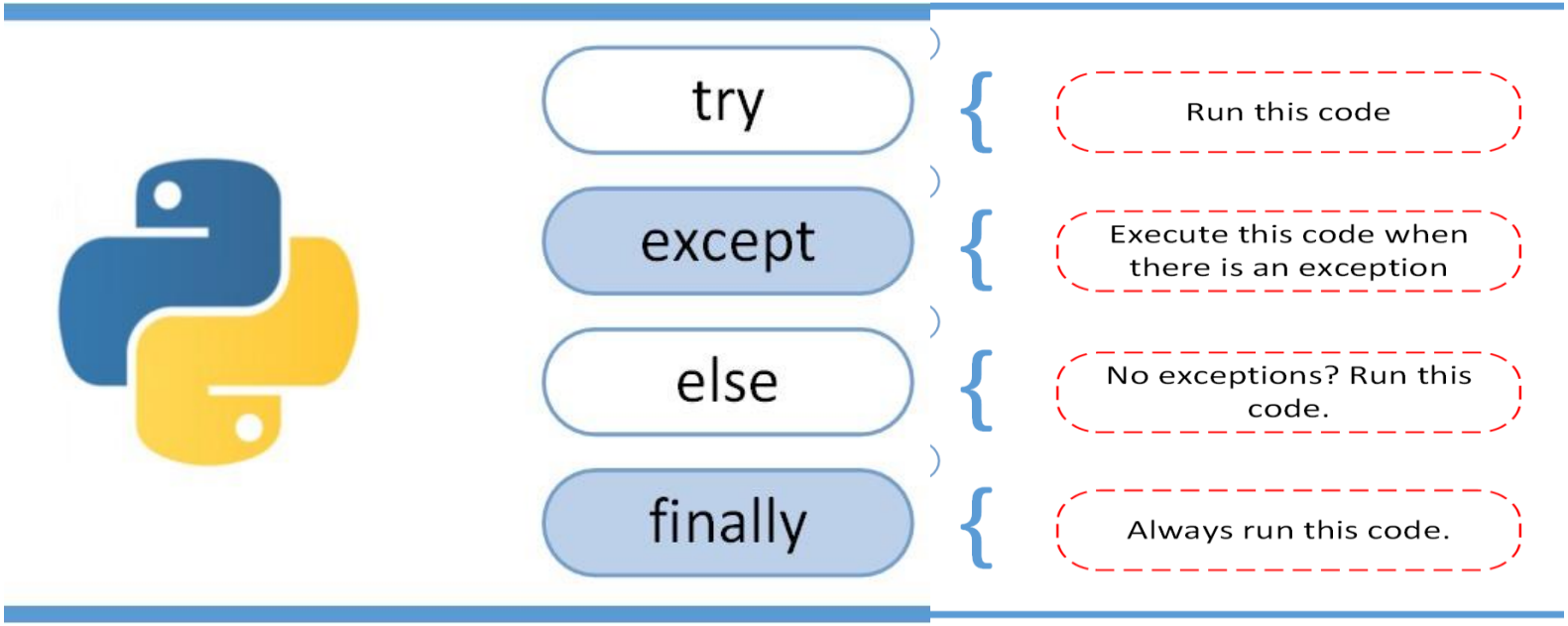
List [ ]	Tuple ( )	Set { }	Dictionary { }
Collection of **ordered** data which can be non-homogeneous data structure	Collection of **ordered** data which can be non-homogeneous data structure	Collection of **unordered** data which can be non-homogeneous data structure	Collection of **ordered** data [key and value] which can be non-homogeneous data structure but key value pair
Example: L1 = [1, 2, 3, 4] L2 = [3.4, -9.0, 0.23] L3 =["A","12B","z123"] L4 =["A",True,12.4,-90, "789"]	T1 = (1, 2, 3, 4) T2 = (3.4, -9.0, 0.23) T3 =("A","12B","z123“) T4 =("A",True,12.4,-90, "789“)	S1 = {1, 2, 3, 4} S2 = {3.4, -9.0, 0.23} S3 ={"A","12B","z123“} S4 ={"A",True,12.4,-90, "789“}	D1 = {"Name" : "Cho", "Height" : 168.0, "Age" : 42, "Email" : "Ch20@gmail.com", "Status" : True}
Can have **duplicate** elements	Can have **duplicate** elements	**Can not** have duplicate elements	**Can not** have duplicate keys but values
To create an empty list - Example **List1 = [ ]**	To create an empty tuple - Example **Tuple1 = tuple( )**	To create an empty set - Example **Set1 = set( )**	To create an empty dictionary - Example **Dict1 = { }**
Mutable, so elements in the list can be changed	**Immutable, so elements in the tuple can not be changed**	**Mutable, so elements in the set can be changed**	Mutable. But Keys are not duplicated

9. Exception classes in Python

What is Exception then?

- Errors detected **during execution** are called **exceptions**
- Example: ZeroDivisionError, FileNotFoundError, TypeError… …
- Program **gets terminated abruptly** if the program is executed with execution errors.

Here, we use Python exception handling technique: **try-except-finally**



User / custom defined exception

```
#user defined exception
class valueSmallError(Exception):
    pass
class valueBigError(Exception):
    pass
#main program
n = 10 #guess number
while True:
    try:
        x = int(input("Enter your guess number:"))

        if x<n:
            raise valueSmallError
        elif x>n:
            raise valueBigError
    except ValueError:
        print("The input must be integer only")

    except valueSmallError:
        print("The value is small try again")

    except valueBigError:
        print("The value is big try again")
    else:
        print("Your guess is correct")
        break
```

User / custom defined exception

```
#user defined exception
class valueSmallError(Exception):
    def __init__(self, arg): # define parameter to refer the message
        self.msg = arg     # copy the message come from raise
class valueBigError(Exception):
    def __init__(self, arg): # define parameter to refer the message
        self.msg = arg     # copy the message come from raise
#main program
n = 10 #guess number
while True:
    try:
        x = int(input("Enter your guess number:"))

        if x<n:
            raise valueSmallError("The value small try again")
        elif x>n:
            raise valueBigError("The value is big try again")
    except ValueError:
        print ("The value must be integer")
    except valueSmallError as error:
        print(error)
    except valueBigError as error:
        print(error)
    else:
        break
print("Your guess is correct")
```

User / custom defined exception

```python
# Custom exception
class InvalidAgeError(Exception):
  def __init__(self, arg): # define parameter to refer the message
    self.msg = arg     # copy the message come from raise

def vote_eligibility(age):
  if age < 18:
    raise InvalidAgeError("Person not eligible to vote, age is " + str(age))
  else:
    print('Person can vote, age is', age)

# calling function
try:
  vote_eligibility(22)
  vote_eligibility(14)
except InvalidAgeError as error:
  print(error)

#https://www.netjstech.com/2019/09/user-defined-exceptions-in-python.html
```

Shell

```
Python 3.7.9 (bundled)
>>> %Run example8.py

 Person can vote, age is 22
 Person not eligible to vote, age is 14
```

10.typical structure for simple python program.

```
# 1.import libraries necessary.
import datetime
import time
import numpy as np
      ......
```

```
# 2. define sub progress (sub function / sub prcedure)
def show(x):
    if len(x)==0:
        return "Without any accessories,0.0"
    if len(x)!=0:
        ......
```

```
def updatetxt(a):
    try:
        f1=open(a,"r")
        ......
```

```
# 3. main program (call sub progress defined)
while True:
    show(x)
    .....
    updatetxt(a)
    ......
```

Python extend learning and application

- DATA ANALYTICS AND PYTHON PROGRAMMING
- ALGORITHMS, PSEUDOCODE, AND LANGUAGE PROCESSORS
- DATA/INFORMATION STORAGE FORMATS
- INTERFACES AND DOCUMENTATION
- DATA MINING IN AI

Thanks!