Python lists, Tuples, sets

collection: Single variable used to store Multiple values

List = [ ] ordered, changeable and doublicate ok

fruits = ["apple", "Orange", "Banana", "Coconut"]

fruits[i] ← To get the element with the referred index from i=0 to i=3

fruits [: :] if step = -1 it will reverse List

start end ~~end~~ step

[1] len(fruits): return Length of list

[2] "variable" in fruits ← returns True if the variable in the List false otherway

[3] fruits.append(" "): To add element in the end of list

[4] fruits.remove(" "): delete element

[5] fruits.insert(index, " "): add element in desired index

[6] fruits.Sort(): Sort elements

[7] fruits.reverse(): reverse elements

[8] fruits.clear(): clean all the list

[9] fruits.index(" "): returns index of the element

[10] fruits.Count(" "): returns the number of occurance of this element

(21)

Set = { } unordered and immutable, but Add/Remove ok. No doublicates

Fruits = { "apple", "orange", "banana", "coconut" }

Print(fruits) : Print elements in different order every Time

We can't Use (fruits.index() / fruits[i])

1. fruits. add (" ")
2. fruits. remove (" ")
3. Fruits. pop () : delete a random element

مقدرش استخدم اى داله فيها ترتيب او حتى اى اجيب index

لانها مش مترتبة وملهاش علاقة بترتيب دخول العناصر اللى انت بتكتبها

Tuple = ( ) : ordered and unchangeable. Duplicates ok
Faster

Fruits = ("apple", "orange", "banana", "coconut")

~~Fruits print()~~

Print(fruits) : Print element with '—'
'apple'

we don't have many methods.

1. " " in Fruits
2. fruits. index (" ")
3. Fruits. Count (" ")

~~Two 2D list~~

fruits = []
vegitubles = []
meats = []

groceries = [fruits, oranges, meats]

To make 2D list we use morethan one 1D List
first list represent first row first element in every list
will represent first column.
To deal with 2D list we use nested loop

for collection in groceries :
    for food in collection :

We can do ~~test~~
2D Tuple; sets
(( )) {{ }}

25 dictionarry: acollection of Values pairs ordered and changeable. No doufLicates { Key : value }

capitals = { "USA" : "washington D.C",
"Russia" : "MOSCOW" }

Some methods

1) capitals.get("USA") : retuns the value with This
capitals.get(" ↑ ") : it returns None
(variable doesn't exist)

2) capitals.update({ "Germany" : "Berlin" }) we use it add new element OR change exist element
capitals.update ({ "USA" : "Dectorit" })

3) capitals.pop("China") : To remove element
4) capitals.popitem ( ) : remove latest Key value.
5) capitals.clear ( ) : remove all elements
6) capitals.keys ( ) : will return keys dict_keys (['USA', 'RUSSIA'])
7) capitals.values ( ) : dict_values (['was-', 'Mos-'])

Good!

[28] capitals.item() : will return our dictionary as 2D Tuple
dict_items ([('USA' : 'Wasgnton D.C'), ('Russia', 'Moscow.')])

To generate a random number we need to import random.

① random.randint(1,6) : return random int number between 1,6
② random.random() : return float number between 0,1
③ random.choice( ) : Choose value from a collection
Tuple or list or set
④ random.Shuffle( ) : Shuffle a collection and return

[31] F: Functions

Function: A block of reusable code

→ def name ( ):
To invoke it name()

You can add argument by

def name (name):

To invoke function you need to use matching numbe of arguments and in the same order

→ return : Statement used to end a function and send a result back to the caller

print("1","2","3","4","5"
,sep="-")
1-2-3-4-5

32

default argument : default value of a certain Parameter
it is used when argument omitted

def net_Price(List_Price, discount, tax) ← without default
def net Price (List_price, discount=0, tax=.05)
default value of argument

ده ممكن نستغني عنه
net-Price(List_Price) or net-Price(List_Price, discount).

important note: or all
((any default argument should be written after positional argument))

↳ To make your function flexiable reduce number of arguments

33

→ Keyword argument : an argument Preceded by an identifier
helps with readability, order doesn't matter

def hello (greeting, title, first, Last)
hello ("Hello", title="Mr.", Last="Eltohamy",
First="Mohamed")
Hello Mr Mohamed eltohame.

→ Positional arguments should be written before Keyword argument

Arbitarary:

Tuple *args : allows you to pass multiple non-key arguments.

dic **kwargs : allows you to pass multiple keyword-arguments

*unPacking operators

```
→def (a, b)
    return a+b        Positional.
Print (add(1,2))

def (*args):  → is a TUPLE
For arg in args:
    Total += arg
    return total
```

} this can take too many arguments

→ dictionary.

```
def Print_address (** kwargs):
```

** kwargs should Follow *args

kwargs.get('—')

↑ name of argument

35

→ module : a file containing code you want to include in your program use 'impor' to include it
(buitin - your own)

↳ import math.

Print(math.Pi) to access value of Pi

↳ import math as m.

Print(m.pi)

↳ From math import Pi

Print (Pi)

How to create a module.
① we open a new ~~file~~ python File.
② i will define some Functions and values.
③ import name of my module.

36 Variable Scope : where variable is visible and accessable
Scobe resultion - (Local – Enclosed – Global – Built in) (LEGB)
1 2 3 4

```
def Func1():
    a=1
    Print(a)
```
} Scobe OF a

Global Scope: out of any Fundion

```
F Func2()
    b=2
    Print(b)
```
} Scobe OF b

37

exception handling:

exception: events detected during excution that interrupt The Flow OF aprogramer

try:
    My code that May has excaption
except Exception:
    Print ("Something went wrong")

We have many specific exceptions
→ except ZeroDivisionError as e:
→ except ValueError as e:

We should catch specific exception then Put except Exception

Finally: handles وظيفتها

    Print ("This will always excute")

38 File detection.

import os
Path = "Path"

1. os.Path.exists(Path): returns if Path exist or not
2. os.Path.isFile(Path): returns if this is File or not
3. os.Path.isdir(Path): it is a folder or not

\n -- new line char

39 Python read a File

- with open('text.txt') as File:
  - ↑ if this in the Folde of Proje
  - if not we need to write Path of File

- Print(File.read()) → This method closed File automatically

→ File.closed : returns if File closed or open

40 Python write a File

- text = " ↑ My text
- With open('File Path', 'w') as file. } this override my text
  File.write(text)

With open('File Path', 'a') as File. ← To append on last text

41 Python Copy a File

1. copyFile() : copy contents of a File
2. copy() : copyFile() + Permission mode + destination can be a dictionary
3. copy2() : copy() + Copies metadata (File's creation and modification times)

```
import shutil
shutil.copyFile('Path of src','Path of dst')
```

#12

Move File or directory

```
import os
Source = "Source path"
destination = "dst path"

try:
  if os.path.exists(destination):
    Print("There is already File there)
  else:
      os.replace(Source, destination)
except FileNotFound
```

#13 delete a File

~~import os~~
~~import shutil~~

os.remove() only delete files

os.rmdir(Path) delete directory "empty Foredur" doesn't has files

Shutil.rmtree(Path) delete directory has files.