* Function And Return *

1. A Function is a reusable Block of Code Do A Task
2. A Function Run when You Call it
3. A Function Accept Element To Deal with Called [Parameters]
4. " " Can do The Task without Returning data
5. " " " Return Data After Job is Finished
6. " " Create to prevent DRY " Don't Reapet Yourself "
7. " " Accept Elements when You Called it Called [Arguments]
8. There's A Built in-Function and user defined Functions
9. A Function is For All Team And All apps

syntax

```
def function_name (        ):      ← Parameters
    BlockofCode
```

ex:

```
a, b, c = "Ahmed", "osanaa", "sayed"
def say_hello(n):          → Parameter
    print(f"Hello {n}")    → Task
    return value           → المتغير الراجع

say_Hello("Ahmed")         → Ahmed is argument
```

def → Function keyword "define"

say_hello → Function name

* Function Call

1. nameofFunction (Par1 , Par2 , Par3)

   لو دا كدا هتمرر عن طريق ال Position

2. nameofFunction (nameof=value, nameof=value, ...)

   (nameof = Par1, Par2)

   ← لو كدا هتمرروا حطه ازاي

tuple لصورة ← ى

* Function Packing , unPacking Arguments   *Args , **Args

ى صورة dict ← ط

ـ ستخدم عند عدم معرفة عدد الـ argument للدالة "*"

ـ " أيضا لعمل unPacking للـ Tuples والـ set والـ list

ـ ملاحظة الـ "**" نستخدم لعمل unPacking للـ dictionary

mylist

```
def say_Hello( *People ):
    for name in Peoples:
        Print(f"Hello {name}")

say_Hello("osama", "Ahmed", "sayed")

def say_details(name , *skills):
    Print(f"Hello {name} your skill is : ")
    for skill in skills:
        print(skill)

say_details("osama", "Html", "css", "Js")
```

* Function Default Parameters

ـ وظيفة الـ defaultPar ، يحط قيمة لو حصل Parameter ولو الـ User لم يعطيه قيمة

ـ ملاحظة الـ default Parameter يكتب في آخر الـ Parameters

```
def say_Hello( name , age = "unknown", contry = "unknown"):
    Print(f"Hello {name} your Age is {Age}
        and your contry is {contry}")

say_Hello( "osama", 36, "Egypt")
say_Hello( "Ramy", 28 )
say_Hello ("Ramy")
```

* Function scope *

[1] Global متغير ف اي Function وموجود مع ال Function

[2] local .. داخل Function وموجود ع ال Function اخرى

ـ لو حط متغير اي متغير Global يوضع كلمة global قبله

ـ ال Function تعطي أولوية الى ال local variable ع global "unboundlocalerror" مثلاً ويكون ... arg

* ال Global مفيد لتغيير اي change اي reassigning Function is scope

* Function Recursion *

ـ هو استخدام الدالة لنفسها

ed:
```
def Cleanword(word) :
    if len(word == 1) :
        return word
    print(f"print start function {word}")
    if word[0] == word[1] :
        print(f"print before condition {word}")
        return Cleanword(word[1:])
    print(f"print Before Return {word}")
    return word[0] + Cleanword[1:])
```

* lambda "Anonymous Function" *

[1] it has no name
[2] You can call it inline without define it
[3] You can ~~call~~ use it in Return data From Another Function
[4] lambda used for simple Function and Def مهم Handle the large Tasks
[5] lambda is one single expression not Block of Code
[6] lambda Type is function

ed (parameters, return)
```
Hello = lambda name, age : f"Hello {name} Your age is :{age}"
Print (hello ("Ahmed", 30))
```

* File Handling *

[1] "a" Append → open file for appending value
Creat file if not Exists

[2] "r" Read → "Default value" open file for Read
and Give Error if file is not Exists

[3] "w" write → open file for writing, creat file if not Exists
[4] "x" Creat → Creat file, Give Error if file Exists

→ import os
os.getcwd() → main current working directory
os.path.abspath(__file__) → المسار الكامل للملف اللي بكتب فيه
os.path.dirname(paths) → Directory for the opened file
os.path.abspath(__file__)
os.chdir(path) → Change Current working Directory
Ex: os.chdir(os.path.dirname(os.path.abspath(__file__)))

open(r" Path ") افتح الملف
raw يعني الـ Compiler لما يقرأ الـ Path
يمنع الـ escape sequence

→ هنا اديتها 2 Parameter ← لاحظ انا ممكن تتبدلها
open(r"Path", "mode")
المسار الملف ← نوع الفتح r, a, w, x

* File Handling "Read File" *

```
myFile = open("D:\Python\Files\osama.txt", "r")
```

[1] print(myFile) # File Data object

myFile.name ←
.mode
.encoding

[2] print(myFile.read()) → هيقرأ محتويات الملف كله

print(myFile.read(n)) → عدد الحروف اللي هيقرأها

[3] print(myFile.readline(n)) → عدد الحروف اللي هيقرأها في السطر

print(myFile.readline()) → هيقرأ سطر كامل

list ← نوع السطر الرابع يكون

← ممكن loop على الفايل واوقف عند حاجة معينة

ex:
```
for line in myFile :
    print(line)
    if line.startwith("07"):
        break
```

[4] myFile.close() → close the file

* File Handling "write and append in file"

"w" , "a" , "r+"

هيعمل الحاجة مكان المؤشر حاليا ← هيمسح القديم كله ويكتب من جديد

```
myFile.write( value )
myFile.writelines( list )
```
list بياخد

* File Handling "important info"

→ import os

myFile.truncate(number) ← اقتطاع عدد حروف معينة وحذف الباقي

myFile.tell() → يرجع مكان المؤشر

myFile.seek(value) → يودي المؤشر لمكان معين

os.remove("Path") → يحذف file معين

* Bult in functions *

[1] all(iterable) → لو كل العناصر True هيرجع True ولو عنصر واحد False هيرجع False

[2] any(iterable) → لو عنصر واحد True هيطلع True ← لو كلها صفر يعني False

[3] bin(number) → يرجع الرقم ثنائي

[4] id(object) → يرجع الid الobject في الmemory

[5] sum(iterable, start) → بيجمع مجموعة عناصر

← لو حطينا له قيمة هيجمع الstart مع sum بتاع الiterable

بعد الفاصلة "optional" عدد ارقام

[6] round(number, numofdigit) → بيقرب رقم

[7] range(start, end, step) → optional "default = 1"

optional "Default = 0"

و هي notinclude

* Modules *

1. module is A file Contain A set of functions
2. You Can import module in your APP To help You
3. You Can import multiple modules
4. You Can creat your own modules
5. modules saves your Time

* show All function Inside module

```
dir(Modulename )
```

* import main module

```
import name of module
```

ـ تقدر تستخدم الدوال في ال module

```
nameof module.name of function( )
```

* import one or more function from module

```
from nameofmodule import nameof function1 ,nameof function2 ...
```

ـ مكتبة random لعمل ارقام عشوائية

1. random() → float number
2. randint(start,end) → random integer (rang)

```
* import sys
  sys.path.append(r"Path")
  print(sys.path)
```

ـ ملاحظة Path of module
1. في ملف اللغة

* Alias "as" عمل اسم مستعار

```
import random as ra
from random import random as ra
```

ـ ملاحظة لو اسند من *
هيجيب جميع دوال الملف
From name of modules Import *

* install External Package *

1 modul → Function ملف واحد فيه مجموعة

2 Package → file ال modules فولدر فيه مجموعة
الخاصة بال Package دي

2 External Package Downloaded from the Internet

3 You can install Package with python Package manger PiP

4 PiP install the Package and it's Dependencies

5 modules List "https://docs.python.org/3/py-modindex.html"

6 Package and modules Directory "https://pypi.org/"

7 PiP manual "https://pip.pypa.io/en/stable/reference/pip_install/"

اوامر الـ PiP في ال terminal

1 PiP --version اصدار ال PiP

2 PiP list لبشوف الـ Package

3 PiP install nameofPackage لتنزيل الـ Package

4 PiP " " == version اصدار معين

5 " " " >= version " اواعلى منه

6 PiP install nameofPackage --upgrade للتحديث

7 " " " --user " ولوفي مشكلة في user

ex:

```
import termcolor
import pyfiglet
print(dir(pyfiglet))
print(pyfiglet.figlet_format("Elzero"))
print(termcolor.colored("Elzero", color="Yellow"))
print(termcolor.colored(pyfiglet.figlet_format("Ahmed"), color="Yellow"))
```

```
pyfiglet.figlet_format("Text")   # ASCII Art
termcolor.colored("Text", color="color")   # Text with color
```

* Built in Function * "Map"

[1] map Take A Function + iterable iterator
[2] map Called map because it map the function on Every element
[3] the Function Can be Pre-defined Function or lambda Function

syntax: map(Function, iterable)
يطبق الدالة على كل عنصر في الـ iterable

* "Filter"

[1] Filter Take A Function + iterator
[2] Filter Run A Function on Every element
[3] Function Can be pre-defined Function or lambda Function
[4] Filter out All Elements For which the Function return True
[5] The Function need To Boolean value

syntax: Filter(Function, iterable)
function تعريف عادي
ينفذ على العناصر لو رجعت True
اعرض العناصر اللي لو مرت على الدالة هترجع True

* Reduce *

[1] Reduce Take A Function + iterator
[2] Reduce Run A Function on first and second element and Give Result
[3] The Run Function on result And third element
[4] " " " " " " Fourth " And So on
[5] Till one element is left And this is the result of the Reduce
[6] the Function Can be Pre-defined Function or lambda Function

syntax: reduce (Function, iteratable)

المسوحة ضوئيا بـ CamScanner