Data structure And Algorithms

نبدأ مراجعة سريعة كدا على بايثون → Python

• مميزات لغة بايثون

1 سهلة

2 Cross Plat form ⇐ يعني ايه؟ تعمل على جميع الأنظمة
↳ windows - mac - Lunix - mobile Phone

3 ستخدم في العديد من المجالات مثل web application - machine Learning - mobile application ⟵ ليه؟ عشان الكم الهائل من المكتبات Libraries
.... إلخ.

• أول سطر كود

```
• Print ("Hello world")
```

• فنكشن اسمها Print ( ) ⟵ للطباعة

```
Print ( أي حاجة جواها )

Print ("salan")
Prinb (12345)
Print (True)      or      print (false)
```

← مينفعش اكتب جملتين . Print في نفس السطر (X)

```
Print ("I Love Python") ~~P~~ Print ("I Love Php")   X
                                                     ✓
```

و

← لغة Python حساسة ⇐ للحروف الكابيتال والسمول capital و small

Variables المتغيرات

← هو متغير يتم إعطاؤه قيمة ويمكنك تغيير تلك القيمة إلى قيمة أخرى.

← المتغير يتخزن في ال Memory

ازاى اكتب متغير فى لغة Python :-

name = "salah" ✓

ـــــ قواعد تسمية المتغير

① لا يجوز البدء برقم   X 11salah '

② لا يجوز تسمية المتغير بـ شئ built in مثلا   X Print, for, in

③ لا يتم البدء بالحروف الخاصه $, # ,......   X $Ahmed

——————

ازاى اعمل تعليق   Comment

ــــ من خلال رمز #

#Print ("I love Python")

——————

concatination   الربط   أى نص يكتب بين علامتين التنصيص

name1 = 'salah" ⟶ # string

name2 = 'hassan" ⟶ # string   الجزء

print (name1 + name2) ⟶ salah hassan

ممكن كتابة concatination علي الصورة دى برده

print (name1 + " " + name2)

print ('salah' + "hassan")

print (20 + years) ⟹ # error

مينفعش اربط string بـ int

الحل

print (str (20) + "years")

⟶ # 20 years   concatination

. مينفعش اعمل concatination بين number و string

أنواع البيانا ــــ (النصوص - الارقام - .....)

نص ⟶ string

رقم ⟶ int و float, double

منطق ⟶ True - False ⟵ (Bolean)

List ⟶ [ ]

dictionary { }   Tuples = ( )

String

```
X = 'salah"
y = 'salah'
print (x)
print (y)
```

→ salah
salah

← مش هيفرق النص سواء [single / double qoutes]

• طب لو انا كتب سطر طويل اووى مثلا اعلانة (استخدم \ backslash)

```
x = "salahhassan\
    footballbasket...."
```

• مينفعش احط [double / single qoutes] جو [double / single qoutes]

~~طب~~ طب الحل اعمل اى؟

استخدم \

```
X = "salah\" hassan\ وو وو
```

\n ← سطر جديد

\ ← سطر طويل (مثل الاعلان)

لكن ينفع احط [double qoute] جو [single qoute] والعكس صحيح

```
X = "salah 'hassan'"
```

\

طب لو عايز استغنى عن (back slash) اعمل ايه ⟸

استخدم """ ''' Tripple qoutes

لـ بستخدم لطباعه أكثر من سطر

```
X = '''salah
hassan
a
"b"   ← عادى
c
'1'
2
"3"   ← عادى
ووو
```

طب لو عايز اطبع جزء من النص اعمل ايه ⇐ بص للكود

```
X='salah hassan"
Print ( X [0] )  ⟶  [S] ← يطبع
           index

Print ( X [0:5] ) →  Salah
```

← البداية

← لغاية

مع الاستثناء ٥

آخر حرف (s) اسم

x~~~~~~~~~~~~~~~~x

• فيه شوية دوال Function اقدر استخدمها مع string

↳ built in من اللغة

```
name =" salah"
Print (name.upper()) ⟶ # SALAH
```

عشان ~ الدالة upper

• upper ← تحول الحروف إلى capital

• Lower ← تحول الحروف إلى small

```
Print (name. Lower()) ⟶ # salah
```

```
name= 'Salah"
Print (name. is upper ()) ⟶ # false
print (name.is Lower()) ⟶ # True
```

• is upper ← يعني ⇐ هل الحروف الموجودة capital ← True / false

• islower ← يعني ⇐ هل الحرف الموجودة small ← true / false

```
name = 'salah'
print (name. capitalize ()) ⟶ # Salah
```

• capitalize ← تجعل أول حرف capital [من الجملة كلها]

• كابتلس ~~title~~ title ← تجعل كل أول حرف من الجملة [capital]

```
name = "salah hassan"
Print (name. capitalize()) ⟶ #Salah hassan
print (name. title()) ⟶ # Salah Hassan
```

• Split ← Function

.الوظيفة تحول string إلى ← list ← []

مثال

→ Program_lan = 'html css Php C++ Python"

Print (Program_Lan . Split())

فاضي يتم طباعة كل عنصر لـ هيئة list

الخرج → ['html', 'css', 'Php', 'c++', 'python']

Program lan = "html-css-Php-C++-Python"

print (Program_lan . split ("-",2))

يعني هيخلي أول قيمتين كل واحدة لوحدها والباقي
كله كـ (قيمة واحدة)

الخرج → ['html', 'css', 'Php-C++-Python']

ملاحظة اسمها ← (rsplit) ← تعمل عكس split

قطع من الآخر #

بص دي كدا

lan = "html css R Php js"

print (Lan [3]) → # L

print (lan.index(m)) → # 2

موقع (M)

طب لو عايز اعرف عدد الحروف الموجودة في المتغير
← استخدم Len()

X = "Salan hassan"
0 1 2 3 4 5 6 7 8 9 10 11

print (X.Len( )) → # 11

في داله اسمها replace تبدل حرف مكان حرف (أو) كلمة مكان كلمة

X = "salan hassan"

print(X.replace("s", "A")) →الخرج # Aalan ha AAan

print (X.replace ("salan", "Ahmed")) →الخرج # Ahmed hassan numbers الأرقام

integer → 14

float → 14.5

complex → 7+9j

طب لو عايز اعرف نوع المتغير (الرقم) بالبايثون

Print (type (14)) → # int

مفتاح اسمها Function دوال ← بستخدم مع الأرقام

- Print (round (7,3)) → # 7 تقريب لأقرب رقم
- Print (round (7,5)) → # 8

لو الرقم بعد العلامة ٠، ١، 2، 3، 4

لأكبر رقم لو سواء الرقم بعد العلامة 5، 6، 7، 8، 9

- Print (floor (7,9)) → # $\boxed{7}$ يطلع للرقم الأقل

Print (ceiL(5,1)) → # $\boxed{6}$ يطلع للرقم الأعلى

Print (Pow (2,3)) → # $\boxed{8}$ الأس

or Print (2*2*2) → $\boxed{8}$

Print abs((-2)) → $\boxed{2}$: القيمة المطلقة [دائما موجب]

Print (max (4, 5)) → #5 الرقم الأعلى

Print (min (4,5)) → # 4 الرقم الأقل

Input الإدخال

x= input ('enter your lan1: ")
z= input ("enter your lan2: ")
print ("i love" + x)
print ("i love + z)

→ enter your lan1 : Php
→ enter your lan2 : C#
الخرج ⇒ i Love Php
i Love C#

*~~~~~~~~*

List [] → تكتب بداخلها بيانات رقمية / منطقية / نصية
0,1,2 True False "Salah"

~~List~~ ●

يمكن كتابة List بداخل List

Lista=["ahmed",1, True, False, 10, 7, [1,2,3[1, 2,3,4]]]
print (Lista) → # يطبع List كلها
Print (Lista(0)) → # ahmed' يطبع ahmed بس

← يمكن تغيير قيمة بداخل List لقيمة أخرى

Lista=["salah", 1, 2, 3, 4, 5]
● Lista [0] = "Hassan"
Print (Lista) ⇒ # ["Hassan", 1, 2, 3, 4, 5]

*~~~~~~~~*

شوية دوال هنستخدمهم مع List

1 append ← تستخدم لإضافة عنصر بداخل ال List ويتم اضافة من آخر List

x=[1,2,3]
x. append(4)
Print(x) → # [1,2,3,4]

2 insert ← تأخذ قيمتين // حيث يتم اضافة العنصر أي مكان (جوا List)
x = [1,2,3]
x. insert (1, "salah")
print(x) → # [1,"salah", 3, 4

3] extend ⟵ دمج 2 list مع بعض

```
x = [1,2,3]
z = [4, "salah", "hassan"]
x.extend(z)
print(x) —الخرج→ # [1,2,3,4,"salah",'hassan']
```

4] sort ⟵ [a→z, 0→9] ترتيب الارقام / العناصر تصاعدياً

```
x = [4,12,31,5,9,0,3]
z = ["salah", "Ali", "tamer"]

x.sort()
print(x) ——→ # [0,3,4,5,9,12,31]
```

sort 0→31 تصاعدي

```
z.sort
print(z) ——→ # ['Ali','salah','tamer']
```

sort a→z تصاعدي

```
z.sort(reverse=True)
print(z) ——→ ['tamer', 'salah', 'Ali']
```

5] remove ⟵ ازالة لعنصر بداخل list

6] clear ⟵ ازالة لجميع عناصر list

```
x = [1,2,3,4,salah]
x.remove(4)
x.clear()
print(x) —الخرج→ # [1,2,3,'salah']
print(x) —الخرج→ [ ]
```

⟵ list فارغة

Tuples → ()

لا يمكن التعديل عليها
تأخذ رقام وحروف و....

```
tuples = (1,2,3,4,5,6)
print(tuples) ──────→ # (1,2,3,4,5)
tuples[0]=5 ──────→ # Type error
```

dictionary → {}

كل key له معنى
كله ومعناها
'name': ahmed
↑ Key ↑ المعنى
تأخذ رقام , حروف و....

```
info = { "name": ahmed,
"age" : 25
"country": "egypt" }
Print (info) ──الخرج──→ # {'name': 'ahmed' , 'age': 25 , 'country': 'egypt'}
print (info["name"]) ──→ # ahmed
Print (info ["age"]) ──→ # 25
print (info[country"]) ──→ # egypt
Print(info.get ("id", "Not found")) ──→ # Not found
```

ستخدم كله get عشان يعمل الأمر None أو [أي أمر انت كاتبه زي
not found, ...
في حالة ~~عدم~~ وجود قيمة ليست بداخل dictionary ..
خصائصها غير مرتبة

Set → { }

```
Myset={"ali", "Ahmed", "hassan", "salah"}
print (my set) →# {'Ahmed', 'ali', 'salah', 'hassan'}
               →# { 'salah', 'Ahmed', 'hassan', 'ali' }
               →# { 'hassan', 'ali', 'salah', 'Ahmed'}
```

الخرج غير مرتب
مختلف في كل مرة

معرفش اجيب index

```
Print (myset[0]) ──→ error
print (my set [0:2]) ──→ error
my set = {1,2,3, "sala", [1,2,3]} ──→ error
myset = {1,2,3, "salah" {1,2,3}} ──→ error
my set = {1,2,3, (1,2,3)} ──→
```

مينفعش اكتب { } أو [ ] جو set
لكن ينفع اكتب ( ) Tuples جو set

بعض الدوال المستخدمة مع : Set

1 union او □|□ — دمج two List

```
a={1,2,3}
b={4,5,6}
print(a.union(b)) → {1,2,3,4,5,6}
print(a|b) → {1,2,3,4,5,6}
```

2 add ← اضافة عنصر بداخل Set

```
a={1,2,3}
a.add(4)
print(a) → {1,2,3,4}
a.  a.dd(4,5)  X error
```

```
a={1,2,3,4}
a.add(5)
a.add(6)     } ← اضافة عنصرين
print(a) → {1,2,3,4,5,6}
```

3 remove ← مسح عنصر بداخل Set

```
a={1,2,3,4}
a.remove(3)
print(a) → {1,2,4}
```

```
a={1,2,3,4}
a.remove(7)
print(a) → error
```

عشان 7 مش موجوده بداخل Set

4 discard ← مسح عنصر بداخل set

```
a={1,2,3,4,5}
a.discard(4)
print(a) → {1,2,3,5}
```

```
a={1,2,3,4,5}
a.discard(7)
print(a) → {1,2,3,4,5}
```

لم يعطي error

الفرق بين discard و remove ←

وجه الشبه: كلاهما يمسح عنصر من set

وجه الاختلاف:

discard	remove
لم يعطي error في حالة اعطاءه رقم لا يوجد بداخل set	يعطي error في حالة اعطاءه رقم لم يوجد بداخل set

5 clear ← مسح جميع عناصر set

```
a={1,2,3}
a.clear()
print(a) → set()
```

عمل function في Python

Parameters: قيم تأخذها الدالة

اسم الدالة

```
def   first function (name,age):
      print ("my name is":+name + "yourage is:"+str (age))

first function("salah",20)
```

```
def  calc  (num1, num2)
     return num2 + num2

print (calc (10,5))
```

return → آخر سطر في function

return → تستخدم لإرجاع قيمة

If لو

```
egyptian=True
if egyptian== True
    print(i am egyptian)
```

→ النتيجة i am egyptian

```
if not egyptian:
if egyptian:
```

→ # flase = قيمة egyptian

→ # True = قيمة egyptian

If , else

```
email= "x@gmail.com"
pussward = 12345678
if email = "x@gmail.com"  and  pasward == 12345678:
    print ("welcom")
elif email=="x@gmail.com" and pasward != 12345678:
    print ("invalid passward")
elif email!="x@gmail.com" and pasward==12345678:
    print ("invalid email")
else ~~email!=~~
    print ("invalid passward and gmail")
```

and → لازم الشرطين يتحققوا

or يكفي شرط واحد

النتيجة → # welcom

Lops → (while)

قارن

```
i = 1
while i <=10
    print(i)
    i + = 1        →
```

```
1   البداية
2
3
4
5
6
7
8
9
10
```

```
i = 1
while i <=10
    i + = 1
    print(i)       →
```

```
2   البداية
3
4
5
6
7
8
9
10
```

من أمر اسمه continue.

```
i=1
while i<=10
    i+=1
    if i == 8
        continup   →
    print(i)
```

```
2
3
4
5
6
7
9
10
```

عند ازاله أمر continue

و نستبدله بـ أمر break

يكون الناتج (الخرج)

2, 3, 4, 5, 6, 7

لعند 8 يخرج وينتهي البرنامج

for

```
For  name  in "salan":
    print ("salan" )   →
```

```
s
a
l
a
n
```

```
Lan= ["java", "python", "C#", "C++", "javascript"]
For Lista in range (len(Lan)):
    Print (Lan [Lista])   →
```

```
Java
Python
C#
C++
javascript
```

← وبعدين الآلة حاسبة والمستخدم دخل نص ← البرنامج هيضرب (مش هيكمل)

طب أنا عايز يظهر رسالة Invalid value عشان يبقى البرنامج موجود ويكمل عادي

← من خلال أمر try: except:

```
try:
    num = int(input("enter avalue"))
     print(num)
excpect:
     print("invalid value")
```

الأول → enter anumber: salah
Invalid value

الثاني → enter avalue: 12345
12345

قراءه / التعديل في الملفات من ملف البايثون

← لو أنا عندي ملف وعايز استخدم ملف تاني مثلاً هيد html أو نص أو.....
اعمله ازاي

← اعمل الملف الأول

test.txt ⇐ اكتب فيه اي حاجه ⇐ salah
hassan
ali
Ahmed

طب أنا هجيب الملف اللي عملته في بايثون (الملف الأصلي اللي أنا كتبت فيه الكلام)

open تاخذ قيمتين

* files= open ("test.txt" و"r")

w ← كتابه ع الملف

r → read قراءه للبيانات ,

a → apond اضافه شيء في آخر الملف ,

r+ → قراءه / كتابه في نفس الوقت
معاً

```
files= opon ("test.txt" , "r")
print (files.readable()) → True
 File.close()
```

readable → True / false قيمة boolean

readLine ← قراءة الملف سطر سطر

readLines ← قراءه كل سطور الملف

read ← قراءة الملف كما هو [زي ما هو]

formating

```
name="salan"
age="20'
Country="egypt'
```

Confatination
الدرسه
الطريقه
مختلفه

```
# Print ("my name is"+ name+"my age is"+str(age)+
        "my Nationality is'+ (cauntry)

# Print ("my name is %s my age is%s my nationality
   is%s" % (name, age, country))
```

%s s — string نص

%d — أرقام

%f — float

```
print ("my age Is %f "%age) → 20,000000
Print("my age Is %.3f"%age) → 20,000
```

```
name="salan"
age="20'
```

formating نفس الشكل من concatination

```
#Print ("my name is"+name+"my age is"+str(age))
#Print ("my name is %s my age is %f " %(name,age)
#Print ("my name is {  } my age is { }".format(name,20)
```

```
a=1
b=2
c=3
Print ("the nums are [a}{b}{c}" ) →  a
                                     b
                                     c
Print (f"the nums are {a}{b}{c}") →  1
                                     2
                                     3
Print (f"the nums are {a}.{b:3f}{c:2f}" →   1
                                            2,000
                                            3,00
```

Modules

اعمل ملف X.py اكتب فيه دوالمنز

```
def CallName(name):
    print(f"my name is {name}")
CallName("ahmed")
```

```
import X
X.CallName("ahmed")
```

الملف الاعلى بيعمل/المستدعي الدالة من الملف الأعلى

الخرج → My name is ahmed

طب لو عايز أكثر من دالة بعمل ايه

```
~~impo~~ from X import Callname
Call name ("ahmed")
```

Modules جاهزة من البايثون

math → import math
print (math.ceil( ))

ceil → ceil(10,1) → 11 ارجاع العدد الأكبر
floor → floor(7,9) → 7 ارجاع العدد الأصغر

```
from math import ceil
print(ceil(7,9)) → # 7
```

```
import random
print ( random.randint( 1, 10 ) ) → randint ارجاع رقم عشوائي بين رقمين
```

الخرج 7 و لو شوية → 8 و .....

```
From random import * → All
print (randint(5,15)) → 6
                        7
                        10
                        12
                        8
                        :
```

عمل استدعاء للكل (*)

OOP ←

Object-orlented-Programming

ده

كل ما هو شيء

دي object له

Action
افعال

Attributes
صفات

Function
دوال

Variables
متغير

طريقة كتابة الكود

object عبارة عن

دوال متغيرات

ازاي اعمل class

```
Class car:
    def __init__(self):
        ~~pr~~ print("hello word")
car()
```