Python

اعداد وتقديم: م. محمد أسامة بيطار

الفصل الأول:

المحتويات

❖ التعليمات الأساسية في بايثون ومقابلاتها

اسم التعليمة	Python
تعليمة القراءة	x = input ("enter x ")
تعليمة الكتابة	Print ("x is ", x)
اسناد القيمة لمتحول	Y = x +5
التعليمة الشرطية	If (condition is true): pass Elif (condition is true): Pass Else: Pass
التعليمة التكرارية for	For x in range (0,5): Pass
التعليمة التكرارية while	While (condition is true): Pass

✎ الفاصلة المنقوطة في نهاية كل تعليمة برمجية " غير مطلوبة " في بايثون، توضع فقط في حال تمت كتابة تعليمات في نفس السطر.

✎ تحدد كتل التعليمات في بايثون من خلال "المحاذاة alignment"

✎ يتم استخدام التعليقات في لغة بايثون من خلال استخدام"#".

✎ اما ازا كان التعليق متعدد السطور فيتم استخدام (""التعليق "").

◆ تعليمة القراءة:

تقوم تعليمة القراءة بقراءة الدخل على شكل سلسلة نصية ولتحويل الى النمط الذي نريد نستخدم اسم النمط الذي نريده على الشكل الاتي:

X = input (" enter x : ")	
Float	X = float (input (" enter x : "))
Int	X = int (input (" enter x : "))
Bool	X = bool (input (" enter x : "))

✎ أي انه وعند القراءة من المستخدم تخزن على شكل string

✎ (ملاحظة مهمة) عند التحويل لنمط boolيكون الخرج على الشكل الاتي:

أي متحول يحوي على قيمة يكون او قيمته العددية غير معدومة عندها التحويل يعطي قيمة true

أي متحول فارغ القيمة او قيمته العددية هي صفر يكون تحويله يعطي القيمة false

◆ تعليمة الكتابة:

يقبل التابع ()print عدة معاملات (أنماط المعطيات المختلفة , توابع) نقوم بالفصل بين المعاملات في

تابع الكتابة من خلال استخدام "," ولدمج السلاسل الحرفية يمكن استخدام "+"

◆تعليمة الاستاد:

عملية الاسناد يمكن ان تكون عملية اسناد متعددة أي اسناد قيمة واحدة لعدد من المتحولات.

- من مزايا بايثون ان عملية الاسناد تقوم بتحديد نمط المتحول ضمنيا.
- من المهم الانتباه الى انه يمكن ان تتغير قيمة المتحول عند كل عملية اسناد جديد.
- الاسناد المزود بعملية: (مراجعة للفكرة)

هنا يجب الانتباه الى الشكل العام له:

X = x op y → x op=y

◆ ترتيب الأولويات في العمليات الرياضية على الشكل الاتي:

حيث () هي اقوى عملية و - هي ذات الأولوية الأدنى

التعابير الرياضية في بايثون:

()	الاقواس في العمليات الرياضية
*	الضرب
**	القوة
/	القسمة
%	باقي القسمة
+	الجمع
-	الطرح

نتيجة جمع أو طرح أو ضرب عددين صحيحين هو عدد صحيح. أما إذا كان أحد العددين حقيقيا float فتكون النتيجة هي عدد حقيقي.

نتيجة استخدام عملية القسمة / لعدد على عدد آخر هي عدد حقيقي،

ويمكن الحصول على ناتج القسمة "الصحيحة" لعدد صحيح على عدد صحيح باستبدال عملية القسمة "/" بالعملية "//"

◆ التعليمة الاسناد الشرطية:

تذكر انه لا يوجد داعي للأقواس المتعرجة للدخول الى block ويكتفى باستخدام المحاذاة الداخلية

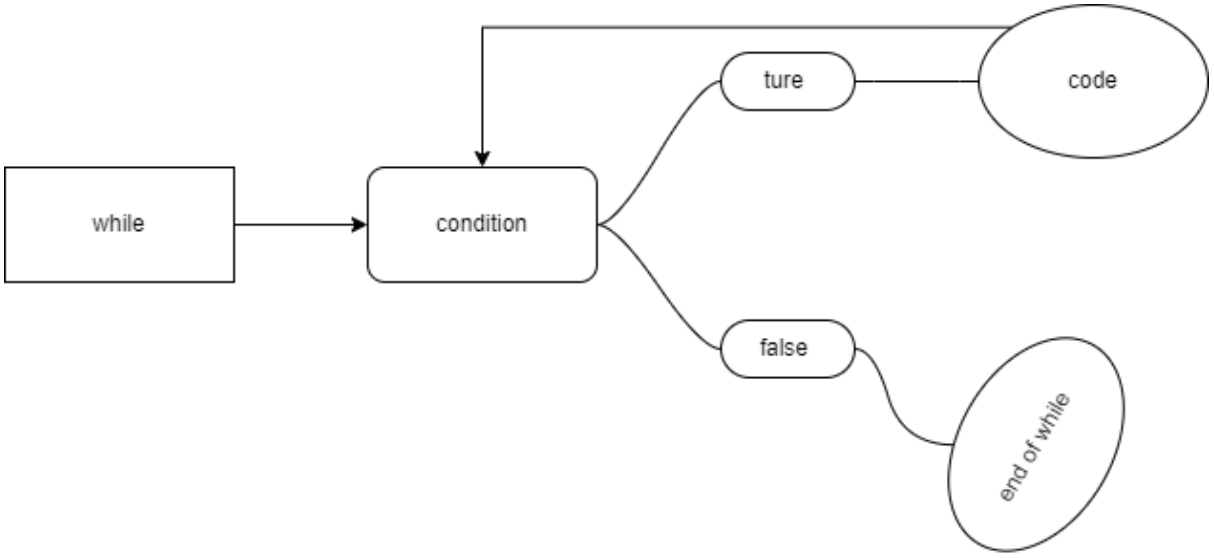
◆ تعليمة الاسناد الشرطية:

If (expression) value1 else value2

أي انه وفي حال تحقق الشرط expression يتم اسناد القيمة value1 والا فيتم اسناد القيمة value2

◆ التعليمة التكرارية الشرطية while:

الشكل القواعدي لتعليمة التكرار الشرطية



◆ التعليمة التكرارية بعداد for:

✎ وهنا يجب أولا فهم التابع ()range:

هو تابع يقوم بإعطاء قائمة (list) من العداد الموجودة ضمن المجال المدخل لهذا التابع (سوف تتم دراسته بشكل مفصل لاحقا)

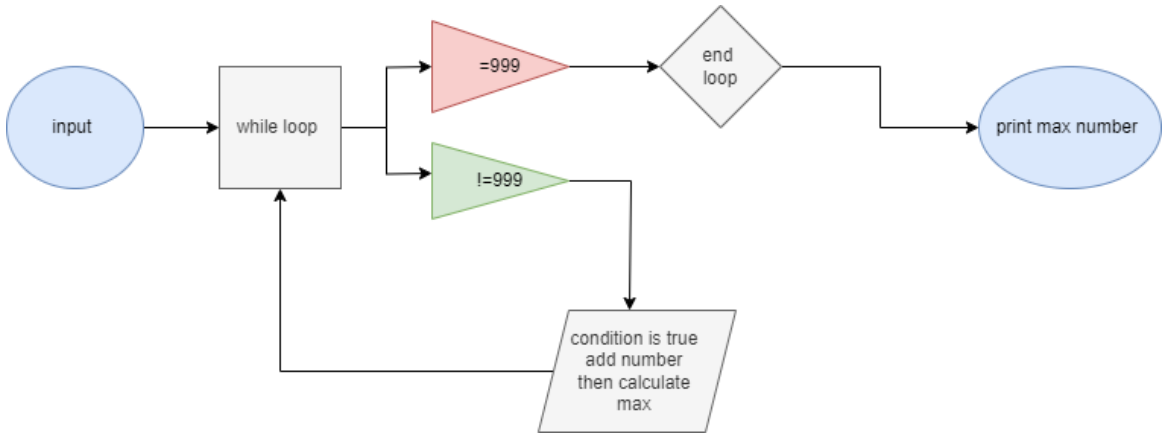
Range (star , end , step)

Start	عدد صحيح يدل على بداية المجال
End	عدد صحيح يدل على نهاية المجال
Step	عدد صحيح يمثل الخطوة وقيمته الافتراضية هي 1

❖ امثلة برمجية:

◆ أولا: معرفة القيمة العظمة بين القيم المدخلة من المستخدم

رسم توضيحي للفكرة العامة:

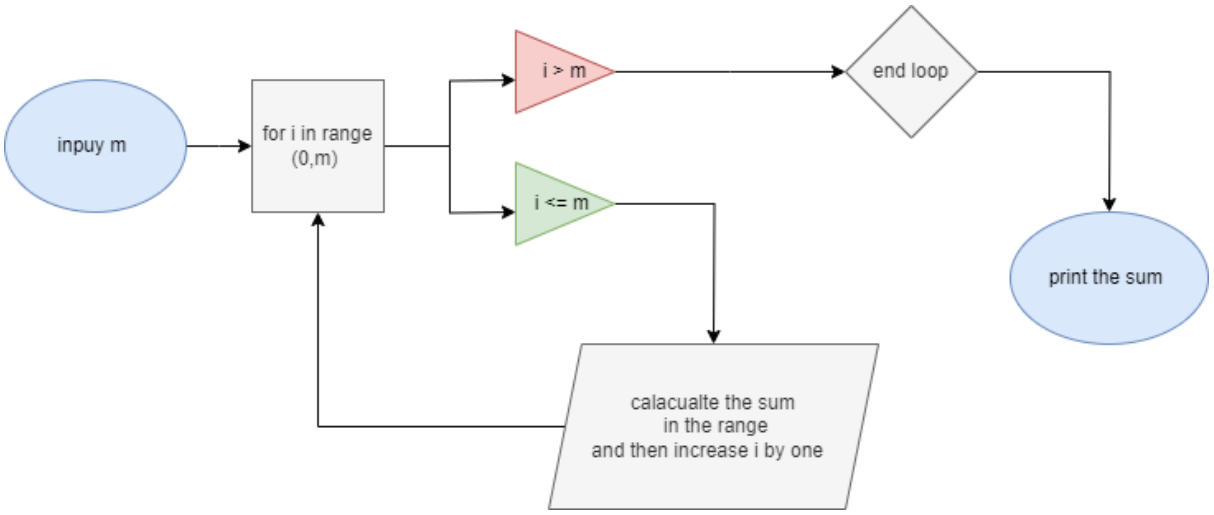


الكود المستخدم:

```
a =int (input(" a= "))
max = a
while a != 999:
    if a > max :
        max = a
    a = int(input(" a= "))
if max !=999:
    print("max =",max)
```

◆ ثانيا: حساب مجموع الاعدد من 1....الى m حيث m يعطى كدخل:

رسم توضيحي:



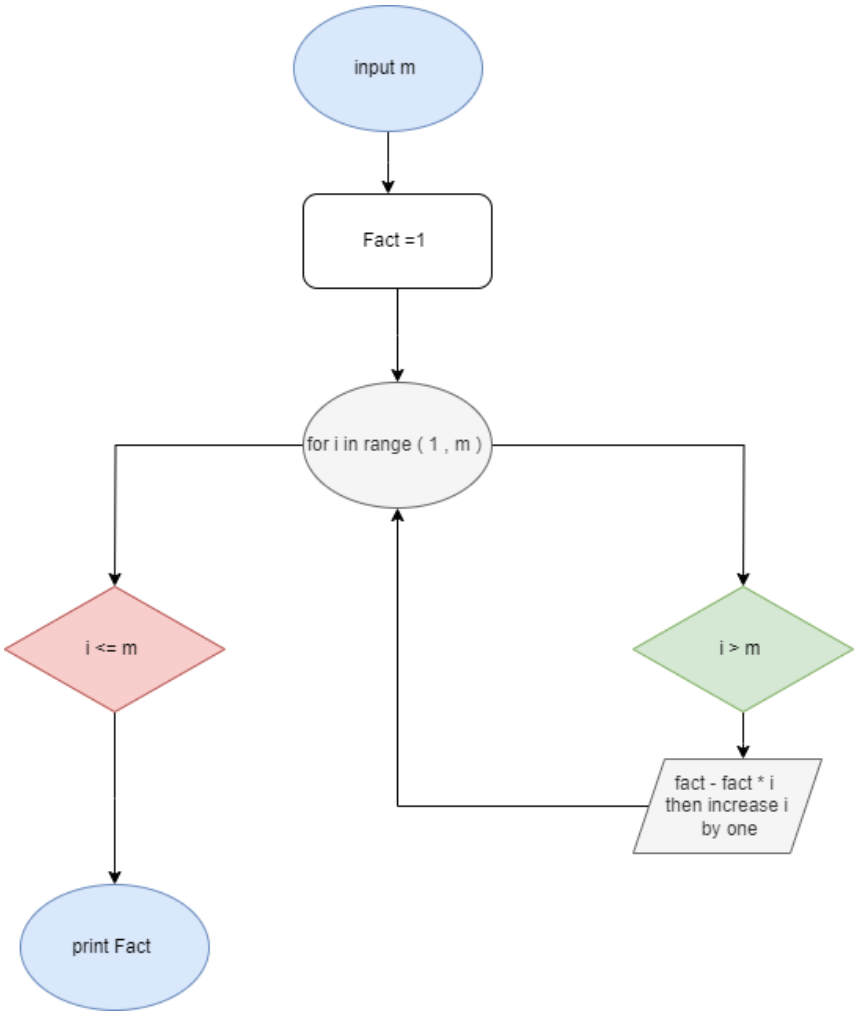
الكود المستخدم:

```
#Read M
print(" Input M : ")
```

```
SM = input(); M = int(SM)
s = 0
#compute the Sum in s
for i in range(1,M+1):
    s = s + i
#Write s
print("1+2+...+" ,M , " = " ,s)
```

◆ ثالثا: حساب قيمة العاملي

رسم توضيحي للفكرة:

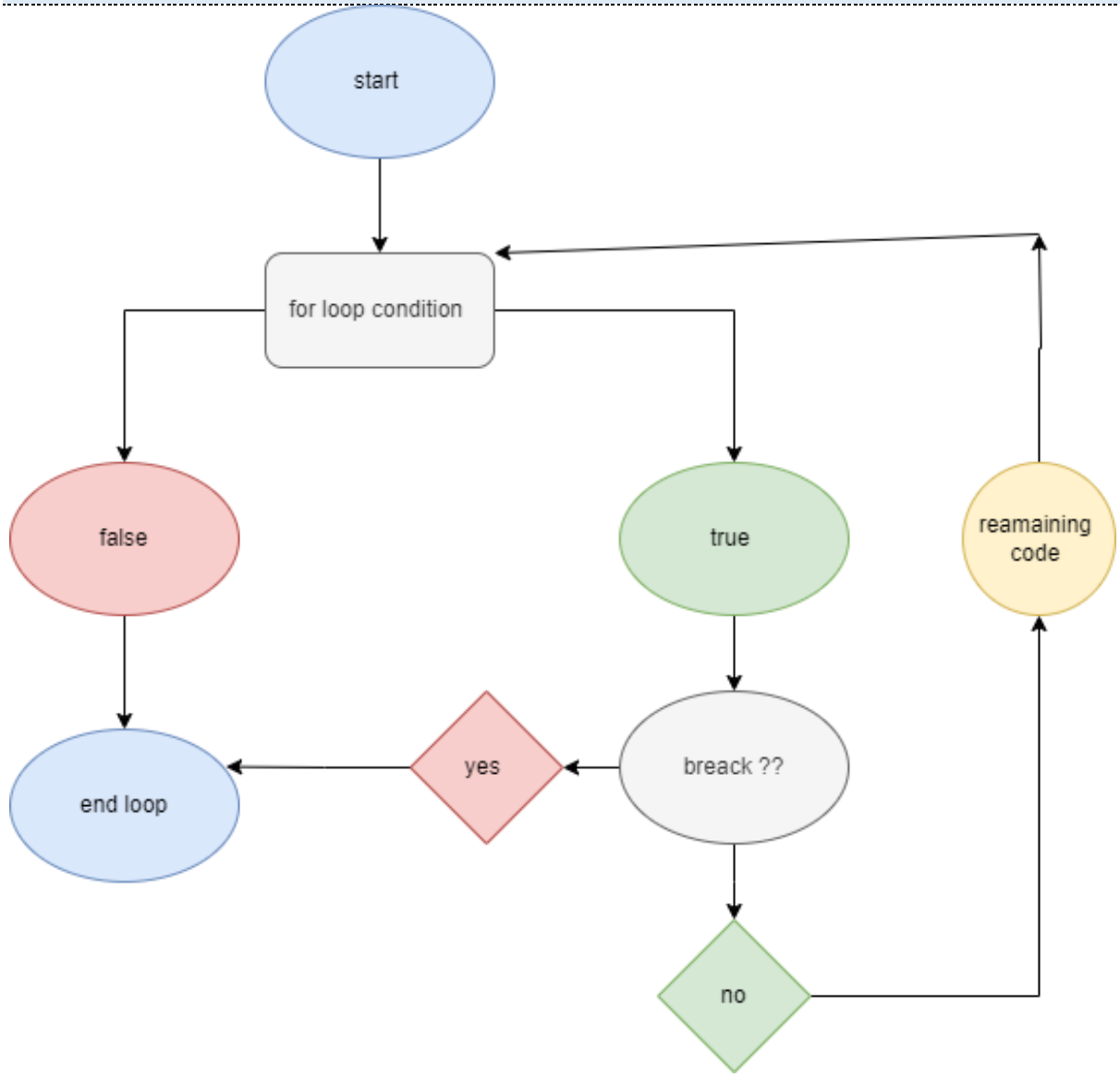


الكود المستخدم:

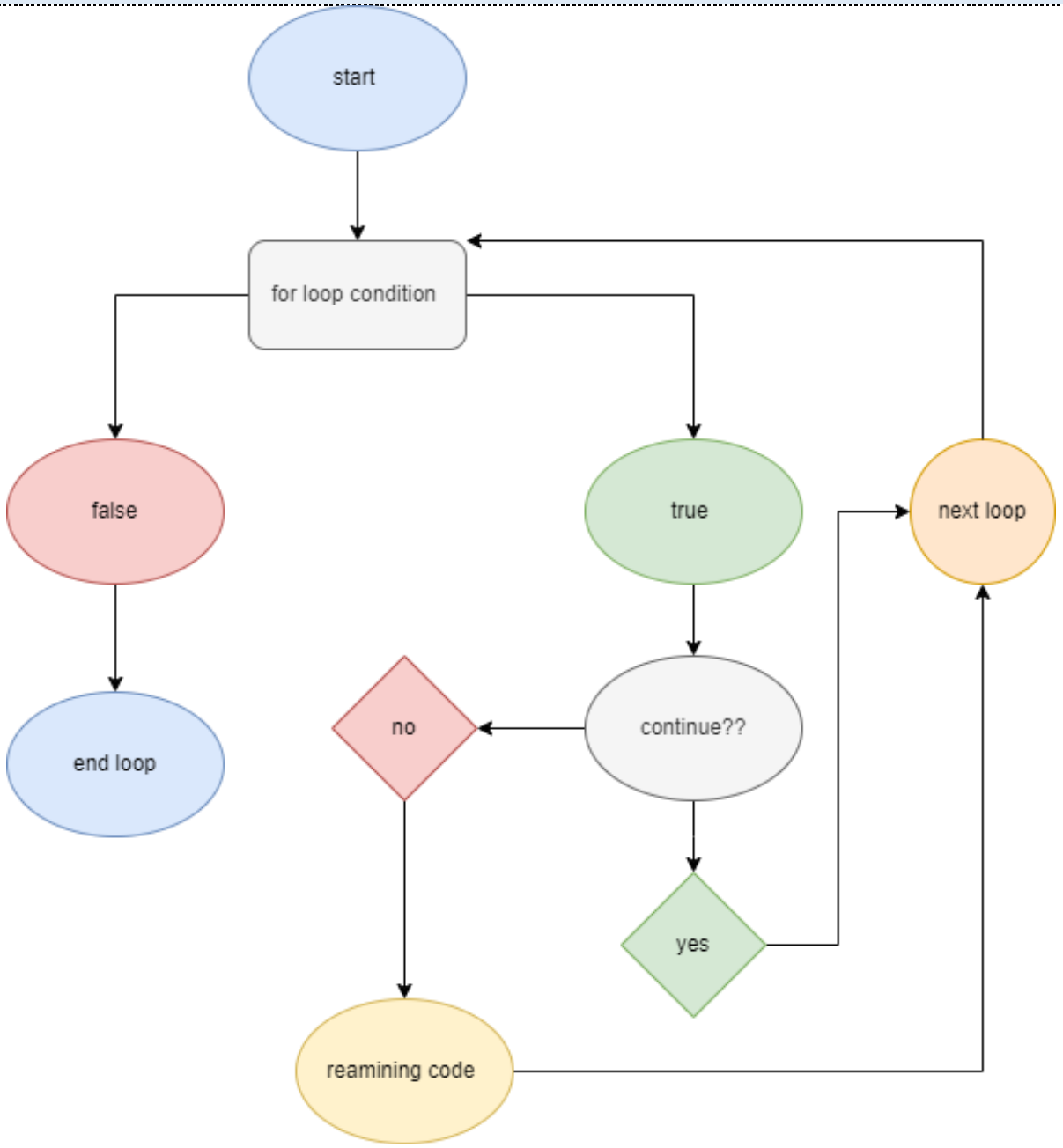
```python
#Read M
print(" Input M : ")
SM = input(); M = int(SM)
#variables initialization
s = 1
#compute the M! in s
for i in range(1,M+1):
    s = s * i
#Write s
print(" " ,M,"! = ",s)
```

❖ كسر البرمجة المهيكلة في بايثون:

◆ تعليمة break:



◆ تعليمة continue:



◆ ملاحظات:

✎ لا وجود لتعليمة do while

✎ لا وجود لتعليمة switch case

❖ أنماط المعطيات الأساسية في بايثون:

ان الأنماط البسيطة في بايثون هي:

Int, float, complex, string, Boolean

اما الأنماط المركبة فهي:

List, set, tuple, dictionary

✎ في بايثون لا يمكن معرفة نمط المتحول قبل ان نقوم بإسناد قيمة له، حيث انه أيضا تتغير أنماط المتحولات بتغير نمط القيمة المسندة له في كل مرة من الاستخدام أي انه لا يوجد نمط دائم للمتحول في بايثون بل هي قابلة للتغير

✎ طريقة الاستفسار عن نمط المتحول تكون من خلال استخدام التابع:

Type ()

◆ التوابع الرياضية الجاهزة للاستخدام مع الأنماط العددية:

القيمة المطلقة	Abs(number)
التقريب	Round(number)
القوة	Pow(number, power)
الجمع	Sum(num1,num2,num3.....)
القيمة الصغرى	Min(num1,num2,num3.....)
القيمة العظمة	Max(num1,num2,num3.....)

❖ العمليات المنطقية:

وهي العمليات المنطقية المعروفة

And, or, not

اما العمليات المنطقية على البنات: bitwise operation

تسمح لنا لغة بايثون بكتابة الأعداد بالنظام الثنائي والثماني والست عشري من خلال استخدام التوابع الاتية:

التحويل الى النظام الست عشري	Hex()
التحويل للنظام الثماني	Oct()
التحويل للنظام الثنائي	Bin()

◆ العمليات على البنات:

And	0&0 =0	0&1 =0	1&0 =0	1&1 =1
Or	0|0=0	0|1=1	1|0=1	1|1=1
xor	0^0=0	1^0=1	0^1=1	1^1=0

صورة توضيحية من مرجع خارجي:

OPERATOR	NAME	DESCRIPTION	SYNTAX
&	Bitwise AND	Result bit 1,if both operand bits are 1;otherwise results bit 0.	x & y
|	Bitwise OR	Result bit 1,if any of the operand bit is 1; otherwise results bit 0.	x | y
~	Bitwise NOT	inverts individual bits	~x
^	Bitwise XOR	Results bit 1,if any of the operand bit is 1 but not both, otherwise results bit 0.	x ^ y
>>	Bitwise right shift	The left operand's value is moved toward right by the number of bits specified by the right operand.	x>>
<<	Bitwise left shift	The left operand's value is moved toward left by the number of bits specified by the right operand.	x<<

❖ فقط في بايثون:

التابع eval() : يقوم هذا التابع بحساب قيمة أي تعبير رياضي مدخل من قبل المستخدم

مدخلات هذا التابع: عبارة عن تعبير رياضي بشكل سلسلة محرفيه

تجربة للتابع:

```
x = 5
print(eval('x == 4'))
x = None
print(eval('x is None'))
```

❖ تمارين الفصل:

◆ التمرين الأول: كتابة جدول الضرب المدرسي

شرح الفكرة العامة : علينا في هذا البرنامج الطلب من المستخدم للعدد الذي يريد معرفة جدول ضربه m

ومن ثم حساب جدول الضرب الخاص بهذا العدد من خلال استخدام التعليمة التكرارية while

وشرط تكرارها هو (i<=10)حيث من خلال هذه الحلقة سوف نقوم بطباعة القيمة

Print (i*m)

الكود المستخدم:

```
print("input k in [2..10] = ")
k=int(input())
# Multiplication Table of number k
i = 1
while i <= 10:
    print(k , " x " , i , " = " , k * i)
    i = i+1
```

تعديل الكود: لكي يقوم بطباعة الجدول المدرسي للأعداد من 1 -10 من تلقاء نفسه:

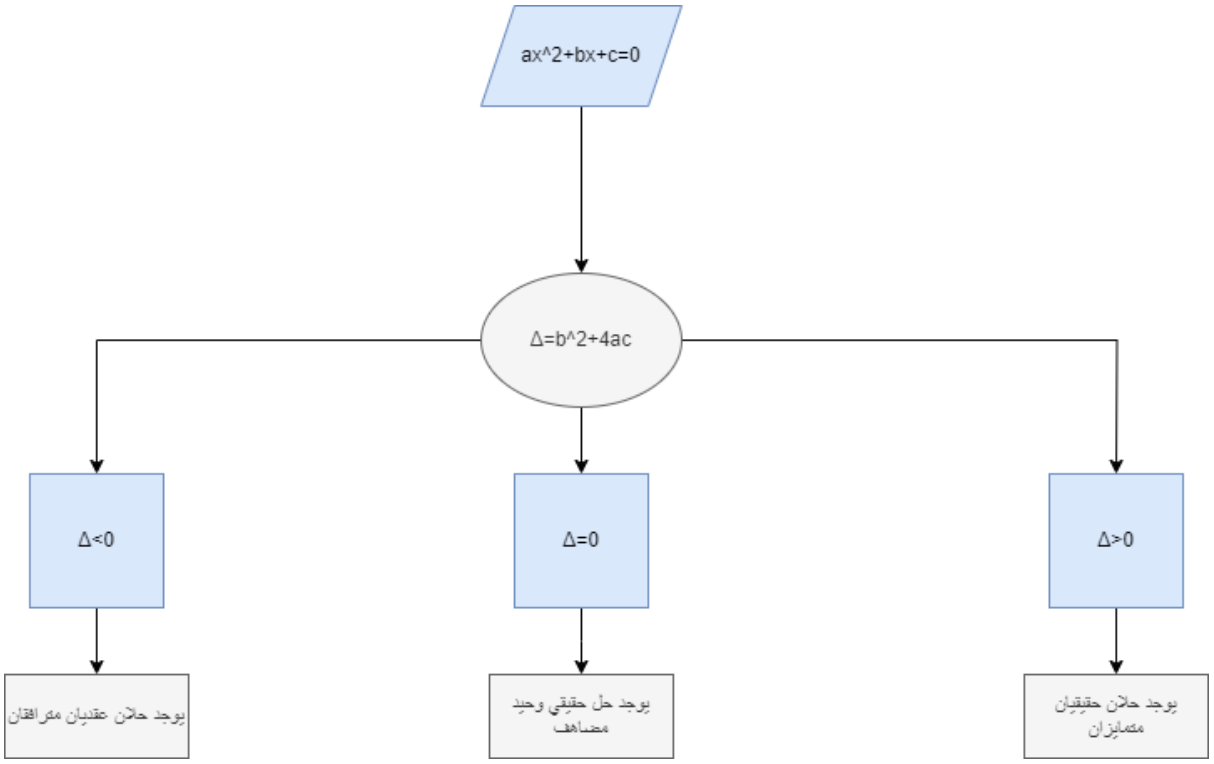
نحتاج من اجل ذلك عدادين هما للقيمة أولى والثانية (i*j) (العدد المضروب والعدد المضروب به)

الكود المستخدم:

```
# Multiplication Table of number k
for i  in range (1,10) :
    for j in range (1,10):
        print(i , " x " , j , " = " , i * j)
        j = j+1
    i = i +1
```

◆ التمرين الثاني: حل المعادلات من الدرجة الثانية بشرط (a!=0)

شرح الفكرة العامة:



أولا سوف يتم طلب قيم كل من a , b ,c من المستخدم وبناء عليه سوف يتم حساب قيمة دلتا التي سوف تقودنا من بعدها الى ثلاث شروط نقوم بحساب قيمة الجذر حسب الشرط المحقق من ضمن الشروط و من ثم نقوم بطباعة الجذر الخاص بالمعادلة

الكود المستخدم:

```
import math
# Read a, b, c
Sa = input(" Input a = "); a = float(Sa)
Sb = input(" Input b = "); b = float(Sb)
Sc = input(" Input c = "); c = float(Sc)
delta = b* b - 4 * a * c
# compute the equation roots
if delta< 0:
    print(" No Real Solution ")
if delta == 0:
    print(" One Solution ")
```

```
    x = -b / (2 * a)
    print(" x= " + str(x))
if delta > 0:
    print(" Two Solutions ")
    x1 = (-b - math.sqrt(delta)) / (2 * a)
    x2 = (-b + math.sqrt(delta)) / (2 * a)
    print(" x1= " , x1 , " x2 " ,x2)
```

تحسين الكود المستخدم لحساب قيمة الجذور العقدية الخاصة بهذه المعادلة:

```
print("Quadratic Equation Solver")

a = float(input("Input a: "))
b = float(input("Input b: "))
c = float(input("Input c: "))

delta = b**2 - 4*a*c

if delta < 0:
    print("Imaginary Roots")
    real = -b / (2 * a)
    imag = math.sqrt(-delta) / (2 * a)
    print("x1 = ", real, "+", imag, "i")
    print("x2 = ", real, "-", imag, "i")

elif delta == 0:
    print("One Solution")
    x = -b / (2 * a)
    print("x =", x)

else:
    print("Two Solutions")
    x1 = (-b - math.sqrt(delta)) / (2 * a)
    x2 = (-b + math.sqrt(delta)) / (2 * a)
    print("x1 =", x1)
    print("x2 =", x2)
```

◆ التمرين الثالث: تتمة التمرين السابق لكن السماح لكل القيم

أي انه وفي حال كانت a=0 ستكون عندها المعادلة من الدرجة الثانية

ويكون حليها بسيطا

الكود المستخدم:

```
# Read a, b, c
Sa = input(" Input a = "); a = float(Sa)
Sb = input(" Input b = "); b = float(Sb)
Sc = input(" Input c = "); c = float(Sc)
if a == 0:
    if b == 0:
        if c == 0:
            print("Each Real is a solution")
        else: #c!= 0
            print("Incorrect Equation")
    else:# b!=0
        x = -c / b
        print("The Solution is : " + str(x))
else:# a != 0
    delta = b * b - 4 * a * c
    if delta < 0:
        print(" No Real Solution ")
    if delta == 0:
        print(" One Solution ")
        x = -b / (2 * a)
        print(" x= " + x)
    if delta > 0:
        print(" Two Solutions ")
        x1 = (-b - math.sqrt(delta)) / (2 * a)
        x2 = (-b + math.sqrt(delta)) / (2 * a)
        print(" x1= " + str(x1) + " x2 " + str(x2))
```

يمكن إضافة القطعة الخاصة بحساب الجذر العقدي في التمرين السابق

◆ التمرين الرابع: الاعداد الكاملة

نقول عن عدد كامل بانه كامل في حال كان هذا العدد مساو لقيمة مجموع قواسمه بما فيه الواحد

مثال: 1+2+3=6 ويكون عندها العدد 6 هو عدد كامل

المطلوب من هذا التمرين هو إيجاد اول n عدد من الاعداد الكاملة ضمن مجموعة الاعداد الطبيعية

1- تعريف متغير compt بالقيمة 0 ليكون عداد لعدد الأعداد الكاملة التي تم إيجادها

2- طباعة رسالة لطلب عدد الأعداد الكاملة المراد إيجادها من المستخدم

3- قراءة عدد الأعداد الكاملة المراد إيجادها من المستخدم وتخزينه في المتغير n

4- تعريف متغير nbr بالقيمة 2 ليكون العدد الذي سيتم فحصه في البداية

5- طباعة العدد 1 كأول عدد كامل

6- بدء حلقة while التي ستستمر حتى يصل عدد الأعداد الكاملة الموجودة إلى n

7- تعريف متغير sumdiv = 1 لحفظ مجموع قواسم العدد

8- حلقة for للتكرار من 2 إلى nbr/2 لفحص القواسم

9- إذا كان الباقي من القسمة يساوي 0، إضافة القاسم لمتغير sumdiv

10- إذا أصبح مجموع القواسم = العدد، طباعة العدد كعدد كامل وزيادة العداد

11- زيادة متغير nbr لفحص العدد التالي

12- تكرار الحلقة حتى يصل عدد الأعداد الكاملة إلى العدد المطلوب n

الكود المستخدم:

```
compt = 0
print("Number of perfect numbers you wish to find : ")
n = int(input())
nbr = 2
print( str(1) + " is a Perfect number")
```

```
while compt != n:
    sumdiv = 1
    k = 2
    while k<= nbr/2:
        if nbr % k == 0:
            sumdiv += k
        k+=1
    if sumdiv == nbr:
        print( str(nbr) + " is a Perfect number")
        compt += 1
    nbr += 1
```

❖ تمارين إضافية:

◆ أولا: القيمة المطلقة

بفرض تم إعطائك كدخل للبرنامج عدد صحيح xنريد ارجاع القيمة المطلقة الخاصة بهذا العدد

مثال: القيمة المطلقة ل 5-هي 5

لاحظ انه لا يمكنك استخدام التابع () abs

◆ ثانيا: اكتب برنامج يقوم بحساب العملية الحسابية المدخلة له من خلال استخدام التابع ()eval

مثال: الدخل كان على الشكل الاتي: (2*15)+34+2+1 يكون الخرج على الشكل الاتي: result is = 67

يتم ادخال الكلمة quitللخروج من البرنامج

عليك استخدام break

◆ ثالثا: اكتب برنامجا يقوم بحساب محيط ومساحة

دائرة يقوم المستخدم بإدخال قطر هذه الدائرة بعلم ان قيمة π=3,1415927

يقوم البرنامج بطباعة قيمة قطر الدائرة ونصف قطرها ومحيطها ومساحتها

◆ رابعا: هل يمكنك نمزجه مسألة خارجية؟!

حساب سرعة الموجة

موجة صوتية في جسمٍ مُعيَّن تردُّدها 260 Hz ،: وطولها الموجي 2.5 m. :بأيِّ سرعة تنتشر هذه الموجة الصوتية في ذلك الجسم، لأقرب متر لكل ثانية؟

الحل:

في هذا المثال، سنتناول موجة صوتية. نعلم من معطيات السؤال كل من التردد f=260Hz، والطول الموجي 2.5 m، وعلينا أن نحسب سرعة الموجة. تذكر أن سرعة الموجة، s، ترتبط بالتردد والطول الموجي من خلال المعادلة $s=f\lambda$. بالتعويض بالأعداد لدينا، نحصل على $s=f\lambda$=260×2.5=650/.Hzms: ومن ثَمَّ، نستنتج أن الموجة الصوتية تنتشر بسرعة 650 m/s. : نريد كتابة برنامج يقوم بمعرفة سرعة الموجة من خلال معرفة

تردد الموجة والطول الموجي الخاص بها بقوم البرنامج بإعادة سرعة انتشار الموجة لا تنسى انه عليك التأكد من انه لا يمكن ان يكون التردد او الطول الموجي اصفارا

الكود المستخدم:

```
f = float ( input(" enter f : "))
lamda = float ( input ("enter lambda :"))
s = f * lamda
if ( f !=0 and lamda !=0 ):
    pritn (s)
```

انتهى الدرس الأول