تم تحميل هذا الملف من موقع المناهج البحرينية

ملخص تقن 106

تاريخ إضافة الملف على موقع المناهج: 2024-05-23 20:21:06

التواصل الاجتماعي بحسب الصف الأول الثانوي

اضغط هنا للحصول على جميع روابط "الصف الأول الثانوي"

روابط مواد الصف الأول الثانوي على تلغرام

التربية الاسلامية	اللغة العربية	اللغة الانجليزية	الرياضيات

المزيد من الملفات بحسب الصف الأول الثانوي والمادة علوم وتقانة في الفصل الثاني

1	المواضيع المطلوبة لمقرر تقن 106
2	الإجابة النموذجية لبنك أسئلة الامتحان النهائي مقرر تقن 106
3	شرح نظام الأنظمة العددية
4	الدروس المطلوبة من مقرر البرمجة بلغة البايثون مقرر تقن 106
5	بنك أسئلة الامتحان النهائي لمقرر تقن 106

الأنظمة العددية

➢ ماهي الأنظمة العددية Numbering systems

هي أنظمة عدّ أو ترقيم تعتمد على مجموعة محددة من الرموز لتمثيل الأعداد ، و تعتمد أيضاً على طريقة معينة لكتابتها و عرضها ، بحيث يحتل كل رمز من رموز النظام العددي موضعا يمثل قيمته .

➢ النظام العشري Decimal Numbering systems

يعتمد النظام العشري على الرموز من 0 إلى 10 لتمثيل الأعداد العشرية ، نسبة إلى العدد 10 ، الذي يمثل بدوره عدد الرموز من 0 إلى 9 . وهو النظام الأكثر شيوعا في حساباتنا اليومية و الأنظمة الاقتصادية و مختلف المجالات.

➢ النظام الثنائي Binary Numbering systems

يعتمد النظام الثنائي على الرموز من 0 إلى 1 لتمثيل الأعداد الثنائية نسبة إلى العدد 2 ، الذي يمثل بدوره عدد الرموز من 0 إلى 1 .

➢ التحويل من النظام العشري إلى النظام الثنائي :

للتحويل من النظام العشري إلى الثنائي يتم قسمة العدد على الرقم 2 و كتابة العدد الصحيح من النتيجة ، فإذا كانت النتيجة تحتوي على الباقي نضع في خانة الباقي العدد 1 و إذا كانت النتيجة لا تحتوي على باقي نضع العدد 0 ، كما في المثال التالي : تحويل العدد (47) $_{10}$ إلى ($\quad\quad$) $_2$:

القاسم	47	23	11	5	2
المقسوم عليه	2	2	2	2	2
النتيجة	23	11	5	2	1
الباقي	1	1	1	1	0

و يتم حساب العدد الثنائي بدءاً من آخر نتيجة نزولاً إلى خانة الباقي فيصبح الناتج : (101111) $_2$

➢ التحويل من النظام الثنائي إلى النظام العشري :

للتحويل من النظام الثنائي إلى النظام العشري ، نستخدم جدول الأعداد العشرية قوة 2 ، بحيث يتم وضع العدد الثنائي في خانات الجدول ، و نقوم بشطب الأعداد التي تحتها الرقم 0 ، و نجمع الأعداد التي تحت الرقم 1 فقط

كما في المثال التالي : لتحويل العدد الثنائي (101111) $_2$ إلى عدد عشري ( ) $_{10}$:

1.نرسم الجدول كالتالي و نبدأ بالعدد 2 قوة 0 ، و نكتب أسفل هذه الخانة ناتج العدد 2 قوة 0 .

2^{12}	2^{11}	2^{10}	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
4096	2048	1024	512	256	128	64	32	16	8	4	2	1

2.نكتب الرقم الثنائي المطلوب تحويله في السطر الأخير ، بحيث نضع كل عدد في خانة منفصله بدءاً من اليمين .

2^{12}	2^{11}	2^{10}	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
4096	2048	1024	512	256	128	64	32	16	8	4	2	1
							1	0	1	1	1	1

3.نقوم بشطب خانة الأعداد التي تحتوي على 0 ، و نقوم بجمع الأعداد في الخانة التي تحتوي على 1 .

2^{12}	2^{11}	2^{10}	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
4096	2048	1024	512	256	128	64	32	16	8	4	2	1
							1	0	1	1	1	1

$47 = 32 + 8 + 4 + 2 + 1$ ⟵ (47) $_{10}$

التعامل مع البيانات

➢ ماهي المتغيرات ؟

هي عبارة عن مواقع يتم حجزها في ذاكرة الجهاز (أجهزة الحواسيب أو الأجهزة اللوحية) بهدف تخزين البيانات فيها ، و يتم إنشاء متغير في اللحظة التي نقوم فيها بتعيين قيمة له لأول مره ، و يمكن تشبيهه بالصندوق الذي يحفظ بداخله الأشياء أو حاوية لتخزين القيم و استخدامها عند الحاجة .

➢ قواعد تسمية المتغيرات :

- يجب أن تتكون فقط من حروف و أرقام (A-Z , a - z , 0-9)
- لا يجب أن يبدأ برقم
- يمكن أن يكون مزيج من الحروف الكبيرة و الصغيرة بدون مسافة (مثال TotalToPay)
- لا تستخدم الكلمات المفتاحية للغة البرمجة python مثل : print , input , sum , max ...

مثال : حددي في الجدول التالي ، ما إذا كانت تسمية المتغيرات صحيحة أو لا مع ذكر السبب :

المتغير	صحيح	خاطئ	ذكر السبب
12Amount		✓	يبدأ برقم
T3b2	✓		مزيج من الحروف الكبيرة والصغيرة و الأرقام
Integer		✓	كلمة محجوزة في لغة البايثون
Size1	✓		مزيج من الحروف الكبيرة والصغيرة و الأرقام
Am%nt		✓	يحتوي على رمز %

➢ طريقة استخدام المتغيرات :

نص	x="Bahrain"	دالة	x=len("Bahrain")
رقم	x=123	جملة إدخال	x=input("الرسالة")
معادلة	x=(y+2)* 9	عملية حسابية	x=6 * 9

إعداد : أ. لطيفة عبدالسلام ، أ. منيرة عبدالسلام

مديرة المدرسة : أ. أمل يوسف مال الله

➢ ماهي البيانات ؟

البيانات هي أي قيمة يتعامل معها النظام أو يحفظها أو يُدخلها المستخدم ، فمثلاً الأعداد هي بيانات ، و النصوص هي بيانات ، و قد صنفت لغة البرمجة python البيانات إلى الأنواع التالية :

➢ أنواع البيانات :

نوع البيانات	الوصف	أمثلة
Integer	عدد صحيح	3 , –4 , 20
Float	عدد عشري	3.5 , –9.6 , 45.84
String	حروف و رموز	“A” , “a” , “+” , “12” , “$” , “welcome”
List	مصفوفة	x = [“mouse”,”keyboard”,”memory”]
Bool	قيمة منطقية	b = False , a = True

➢ العوامل الحسابية و المنطقية و عوامل المقارنة :

❍ العوامل الحسابية :

رمز العامل الحسابي	الوصف	مثال
+	للجمع في حال البيانات الرقميّة	If x=2 & y=3 ➔ x + y=5
+	للربط في حال البيانات النصيّة	If x=”ali” & y=”fahd”➔ x+y=alifahd
–	للطرح	x – y
*	للضرب	x * y
/	للقسمة	x / y
//	للحصول على الناتج الصحيح من عمليّة القسمة	5 // 2 = 2.5 the result is 2
%	للحصول على باقي عمليّة القسمة	5 % 2 = 1
**	للحصول على قوّة العدد	5**2 = 25

❍ عوامل منطقية :

رمز العامل المنطقي	الوصف	مثال
and	و	يجب أن يكون طرفي الجملة المنطقية صحيحان لتكون النتيجة true: 5>2 and 5<10 : the result is **true** 3>1 and 3>5: the result is **false** 4<>2 and 4<=4: the result is **true**
or	أو	يكفي أن يكون أحد طرفي الجملة المنطقية صحيحا لتكون نتيجة الجملة المنطقيّة **true**: 6>3 or 6>10 : the result is **true** 3<1 or 3>5: the result is **false**
not	لا	not(3>1): the result is **false** not(3>1) or 3>4: the result is **false**

❍ عوامل المقارنة :

رمز عامل المقارنة	الوصف	مثال
==	يساوي	x == y
!=	لا يساوي	x != y
<	أصغر من	x < y
<=	أصغر من أو يساوي	x <= y
>	أكبر من	x > y
>=	أكبر من أو يساوي	x >= y

مثال : أوجدي ناتج الجمل الحسابية الآتي في لغة python :

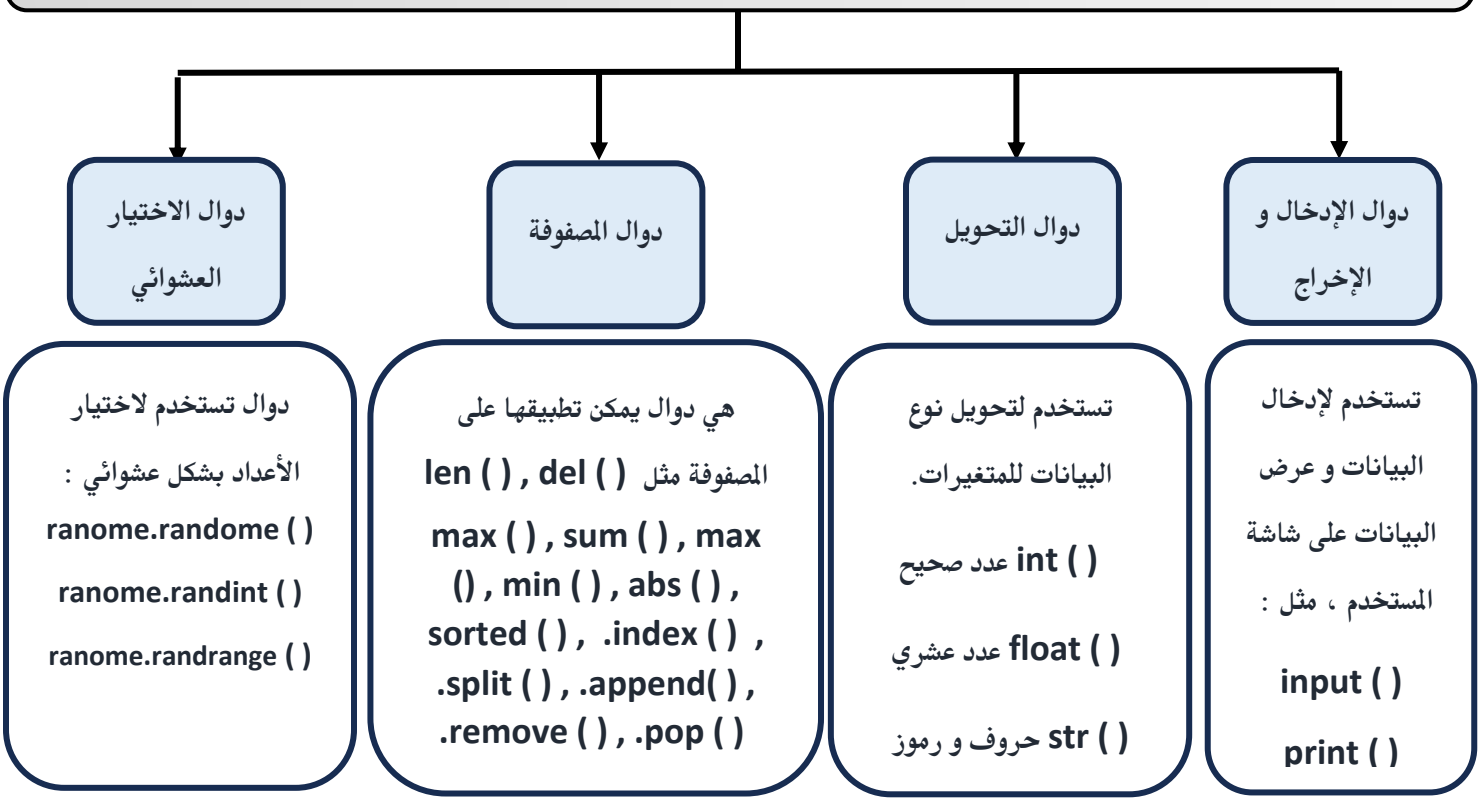
	python code	output
1	13/2	6.5
2	13%2	1
3	13//2	6
4	10**2	100
5	6*3	18
6	5+4	9
7	10–6	3

مثال : أوجدي ناتج الجمل المنطقية الآتية في لغة python مع العلم أن : z=6 y=2 x=4

	python code	output
1	x > y	True
2	z < y	False
3	x = 4	True
4	not (x = y) and z < y	False
5	y<>z and not(x=y)	True
6	x=y or x>(z–y)	False

الدوال البرمجية

الدالة هي كتلة من الجمل البرمجية الجاهزة مسبقا ، يتم استدعاؤها عند اللزوم و تنقسم الدوال إلى :



➢ أولاً : دوال الإدخال و الإخراج :

❍ دالة الطباعة ( ) **print** :

```
print("البيانات المراد طباعتها")
```

❍ طرق استخدام دالة الطباعة :

نص	`print("النص")`	نتيجة عملية حسابية	`print(5*9)`
عدد	`print(123)`	قيمة دالة	`print(len(c))`
قيمة متغير	`print(c)`	نص و متغير	`print("النص", x)`

❍ ملاحظة : – يجب أن يكتب النص بين علامتي التنصيص " " .

- يكتب المتغير أو الدالة بدون علامتي التنصيص " " .
- يمكن أن يكون النص متبوعاً بمتغير ، مع وجود علامة الفاصلة , بين النص و المتغير .

إعداد : أ. لطيفة عبدالسلام ، أ. منيرة عبدالسلام
مديرة المدرسة : أ. أمل يوسف مال الله

❍ **دالة الإدخال :** `varName=input("الرسالة")` ← اسم المتغير

❍ **ملاحظة :**

- **يجب أن نستخدم اسم متغير مناسب لتخزين القيمة التي سيتم إدخالها ، و أن يكون محتوى الرسالة مناسب لمحتوى المتغير ، فإذا كان المطلوب إدخال درجة الطالب ، يمكن أن تكون الرسالة : Enter your grade .**
- **دالة الإدخال input تعتبر أي قيمة يتم إدخالها قيمة نصية ، لذا عند القيام بعمليات حسابية فإن القيم المدخلة سيتم دمجها فقط .**

```
n1=input("Enter number1: ")
n2=input("Enter number2: ")
result=n1+n2
print("the result = ",result)
```

```
>>> %Run -c $EDITOR_CONTENT
 Enter number1: 5
 Enter number2: 4
 the result =  54
>>>
```

➢ **ثانياً : دوال التحويل :**

و هي تستخدم لتحويل القيم النصية إلى قيم عددية مثل :

❍ **تستخدم دالة ( ) int لتحويل قيمة المتغير إلى عدد صحيح .**

```
varName=int(input("الرسالة"))
```

❍ **تستخدم دالة ( ) float لتحويل قيمة المتغير إلى عدد عشري .**

```
varName=float(input("الرسالة"))
```

❍ **تستخدم دالة ( ) str لتحويل قيمة المتغير العددي إلى قيمة نصية**

❍

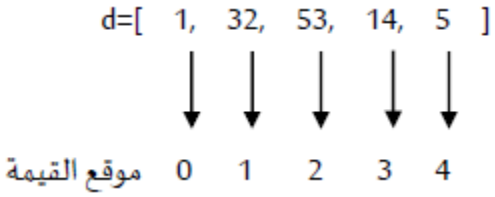
```
varname=26
varname1=str(varname)
```

➢ ثالثاً : دوال المصفوفة:

❍ ماهي المصفوفة ؟ هي أحد أنواع البيانات **list** ، والتي يمكن من خلالها تخزين أكثر من قيمة في متغير واحد ، و تسمح لهذه القيم أن تتكرر في القائمة .

مثال : `d=[1,32,53,14,5]`

ن ستطيع تمييز الم صفوفة من خلال وجود هذه الأقواس [ ] في المتغير ، بالإ ضافة إلى وجود أكثر من قيمة بداخله ، مع مراعاة الف صل بين القيمة بعلامة الفا صلة , فنلاحظ أن الم صفوفة ال سابقة تحتوي على 5 قيم رقمية ، و لكل عن صر في الم صفوفة موقع نطلق عليه **index** ، و يكون مثل الترقيم التسل سلي لقيم الم صفوفة والذي يبدأ دائما من 0 ، فإذا أردنا طباعة القيمة 53 ، يكون موقعها 2 و ليس 3 ، لأنه الترقيم يبدأ من 0



و يمكن أن تحتوي المصفوفة على قيم نصية : `city=["muharraq","manama","riffa"]`

أو قد تكون مزيج من القيم النصية و القيم الرقمية : `c=["muharraq",65,"riffa",23]`

❍ طباعة المصفوفة : `city=["muharraq","manama","riffa"]`

ذكرنا سابقاً بأنه لكل قيمة من قيم المصفوفة موقع تسلسلي يسمى **index** ، لذا يمكن استخدامه في الوصول إلى القيم لطباعتها أو للتعامل معها في مختلف العمليات و الأوامر البرمجية كما في المثال :

قيم المصفوفة	Muharraq	Manama	Riffa
index	0	1	2

```
city=["muharraq","manama","riffa"]
print(city[0])
print(city[1])
print(city[2])
```
(1)

```
>>> %Run -c $EDITOR_CONTENT
muharraq
manama
riffa
>>>
```
(2)

❍ لتسهيل التعامل مع المصفوفات نقوم بتطبيق هذه الدوال ، حيث يمكن تقسيمها إلى نوعين حسب طريقة استخدامها :

دوال المصفوفة

لاستخدام هذه الدوال ، يجب أن نبدأ بكتابة اسم الدالة ، وكتابة اسم المصفوفة داخل أقواس الدالة مع مراعاة وضع متغير لتخزين ناتج الدالة حسب نوعها .

(اسم المصفوفة) اسم الدالة

لاستخدام هذه الدوال ، يجب أن نبدأ بكتابة اسم المصفوفة متبوعة بنقطة ، ومن ثم كتابة اسم الدالة ، مع مراعاة وضع القيمة المطلوبة داخل الاقواس

اسم المصفوفة . اسم الدالة ( )

حذف برقم الموقع	del (اسم المصفوفة)
عدد القيم	x = len (اسم المصفوفة)
أقل قيمة	x = max (اسم المصفوفة)
أقل قيمة	x = min (اسم المصفوفة)
المجموع	x = sum (اسم المصفوفة)
المطلق	x = abs (اسم المصفوفة)
ترتيب تصاعدي	x = sorted (اسم المصفوفة)
x = sorted (اسم المصفوفة , reverse=True)	
ترتيب قيم المصفوفة تنازلياً	

إضافة قيمة	اسم المصفوفة.append (value)
حذف القيمة	اسم المصفوفة.remove (value)
حذف برقم الموقع	اسم المصفوفة.pop (value)
تحديد موقع القيم	اسم المصفوفة= x .index(value)
x =اسم المتغير.split( )	
تقسيم محتوى المتغير ووضعه في مصفوفة	

ملاحظة : جميع الدوال تقوم بإرجاع قيمة ، لذا يجب وضع متغير قبل استخدامها ، حتى يتم تخزين هذه القيمة في المتغير ، ما عدا دوال الحذف و الإضافة فإنها تقوم بعملية الحذف و الإضافة ولكن من دون إرجاع قيمة ، لذا لا نقوم بكتابة متغير يسبق استخدام دوال الحذف و الإضافة .

إعداد : أ. لطيفة عبدالسلام ، أ. منيرة عبدالسلام

مديرة المدرسة : أ. أمل يوسف مال الله

❍ أمثلة استخدام الدوال على المصفوفة : سيتم استخدام المصفوفات التالية لتوضيح طريقة استخدام الدوال:

```
city=["muharraq","manama","riffa"]
age=[25,18,22,23,17]
num=[2,-8,3,-9,-12]
```

➢ لحذف العنصر الأول من مصفوفة **city** باستخدام الدالة **del ( )**:

```
del(city[0])
print(city)
```

(1)

```
>>> %Run -c $EDITOR_CONTENT
 ['manama', 'riffa']
>>>
```

(2)

➢ لحساب أعلى قيمة في مصفوفة **age** باستخدام الدالة **max ( )** :

```
x=max(age)
print(x)
```

(1)

```
>>> %Run -c $EDITOR_CONTENT
 25
>>>
```

(2)

➢ لحساب أقل قيمة في مصفوفة **age** باستخدام الدالة **min ( )** :

```
x=min(age)
print(x)
```

(1)

```
>>> %Run -c $EDITOR_CONTENT
 17
>>>
```

(2)

➢ لحساب مجموع القيم في مصفوفة **age** باستخدام الدالة ( ) **sum** :

```
x=sum(age)
print(x)
```

(1)

```
>>> %Run -c $EDITOR_CONTENT
 105
>>>
```

(2)

➢ لحساب القيمة المطلقة للقيم السالبة في مصفوفة **num** باستخدام الدالة ( ) **abs** مع تحديد موقع القيمة المراد إيجاد القيمة المطلقة لها :

```
x=abs(num[1])
print(x)
```

(1)

```
>>> %Run -c $EDITOR_CONTENT
 8
>>>
```

(2)

➢ لترتيب القيم العددية ترتيباً تصاعدياً في مصفوفة **age** باستخدام الدالة ( ) **sorted** :

```
x=sorted(age)
print(x)
```

(1)

```
>>> %Run -c $EDITOR_CONTENT
 [17, 18, 22, 23, 25]
>>>
```

(2)

➢ لترتيب القيم العددية ترتيباً تنازلياً في مصفوفة **age** باستخدام الدالة ( ) **sorted** :

```
x=sorted(age,reverse=True)
print(x)
```

(1)

```
>>> %Run -c $EDITOR_CONTENT
 [25, 23, 22, 18, 17]
>>>
```

(2)

➢ التعرف على موقع عنصر محدد في المصفوفة city باستخدام الدالة ( ) index :

```
x=city.index("manama")
print(x)
```

(1)

```
>>> %Run -c $EDITOR_CONTENT
 1
>>>
```

(2)

➢ تقسيم النص إلى كلمات و تخزينها في كمصفوفة في متغير باستخدام الدالة ( ) split :

```
y="welcome to computer lab"
x=y.split()
print(x)
```

```
Shell
>>> %Run -c $EDITOR_CONTENT
 ['welcome', 'to', 'computer', 'lab']
>>>
```

➢ إضافة عنصر جديد في المصفوفة city باستخدام الدالة ( ) append :

```
city=["muharraq","manama","riffa"]
city.append("hidd")
print(city)
```

```
Shell
>>> %Run -c $EDITOR_CONTENT
 ['muharraq', 'manama', 'riffa', 'hidd']
>>>
```

➢ حذف عنصر من المصفوفة age حسب موقعه باستخدام دالة ( ) pop :

```
4 age=[25,18,22,23,17]
5 age.pop(3)
6 print(age)
```

Shell

```
>>> %Run -c $EDITOR_CONTENT
 [25, 18, 22, 17]
>>>
```

➢ حذف عنصر من المصفوفة age حسب قيمته باستخدام دالة ( ) remove :

```
4 age=[25,18,22,23,17]
5 age.remove(25)
6 print(age)
```

Shell

```
>>> %Run -c $EDITOR_CONTENT
 [18, 22, 23, 17]
>>>
```

➢ طباعة عناصر المصفوفة من الاتجاه العكسي :

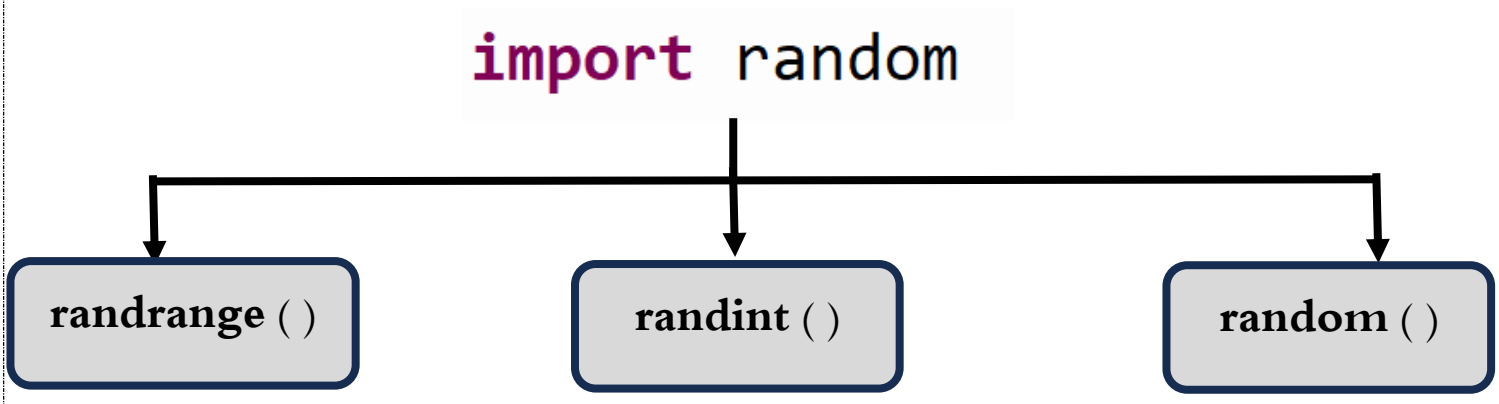
```
4 age=[25,18,22,23,17]
5 print(age[-1])
```

Shell

```
>>> %Run -c $EDITOR_CONTENT
 17
>>>
```

➢ رابعاً : دوال الاختيار العشوائي :

تستخدم دالة الاختيار العشوائي لاختيار عدد بصورة عشوائية و لها عدة دوال مصاحبة لذا يجب أن يتم استدعاء المكتبة الخاصة بها و تسمى **random** في كل مرة نريد أن نستخدم الدوال المصاحبة لها وهي :



➢ دالة **random ( )** :

تستخدم هذه الدالة للحصول على عدد عشوائي عشري بين **0.01** إلى **1.0** ، مع مراعاة وضع متغير ليتم تخزين نتيجة الاختيار العشوائي بداخله ، نلاحظ أن الرقم يختلف في كل مرة نقوم فيها بتنفيذ البرنامج .

```
1 import random
2 x=random.random()
3 print(x)
```

```
Shell
>>> %Run -c $EDITOR_CONTENT
 0.58447530138604
>>>
```

➢ دالة ( ) **randint** :

تستخدم هذه الدالة للحصول على عدد عشوائي صحيح مع تحديد بداية المجال و نهايته .

```
1 import random
2 x=random.randint(1,6)
3 print(x)
```

```
Shell
>>> %Run -c $EDITOR_CONTENT
 5

>>>
```

➢ دالة ( ) **randrange** :

تستخدم هذه الدالة لاختيار عدد عشوائي مع تحديد المجال بعدة طرق :

1. اختيار عدد عشوائي ضمن مجال محدد يبدأ من الصفر ، كما في المثال التالي : تم تحديد المجال 20 لذا سيكون الرقم العشوائي بين 0 و 19 (الرقم 20 لا يدخل ضمن المجال)

```
1 import random
2 x=random.randrange(20)
3 print(x)
```

```
Shell
>>> %Run -c $EDITOR_CONTENT
 19

>>>
```

2. اختيار عدد عشوائي ضمن مجال محدد يتم تحديد بدايته و نهايته كما في المثال التالي : تم تحديد المجال من 20 إلى 31 ، لذا سيكون الرقم العشوائي بين 20 و 30 (الرقم 31 لا يدخل ضمن المجال)

```
1 import random
2 x=random.randrange(20,31)
3 print(x)
```

Shell

```
>>> %Run -c $EDITOR_CONTENT
 23
>>>
```

3. اختيار عدد عشوائي ضمن مجال محدد مع تحديد بدايته و نهايته و خطوته ، كما في المثال التالي : تم تحديد المجال بين 3 و 18 و الخطوات 3 ، لذا سيكون الرقم العشوائي من الأرقام التالية

3 – 6 – 9 – 12 – 15

```
1 import random
2 x=random.randrange(3,18,3)
3 print(x)
```

Shell

```
>>> %Run -c $EDITOR_CONTENT
 12
>>>
```

الجملة الشرطية if

- الجملة الشرطية البسيطة
- الجملة الشرطية الكاملة
- الجملة الشرطية المتداخلة

➢ الجملة الشرطية البسيطة :

يتم استخدام الجملة الشرطية البسيطة لتنفيذ أوامر برمجية محددة بناءا على تحقق شرط معين و تكون كالتالي :

```
if الشرط المحدد :
    الأوامر البرمجية الناتجة عن تحقق الشرط
```

مراعاة كتابة : بعد كتابة الشرط

يجب ترك مسافة قبل جملة الأوامر البرمجية الناتجة عن تحقق الشرط والرجوع لبداية السطر تعني انتهاء الجملة الشرطية

مثال : اكتبي برنامجاً بلغة **python** ، يطلب من المستخدم إدخال العمر ، عرض رسالة **Adult** إذا كانت العمر أكبر من **18** .

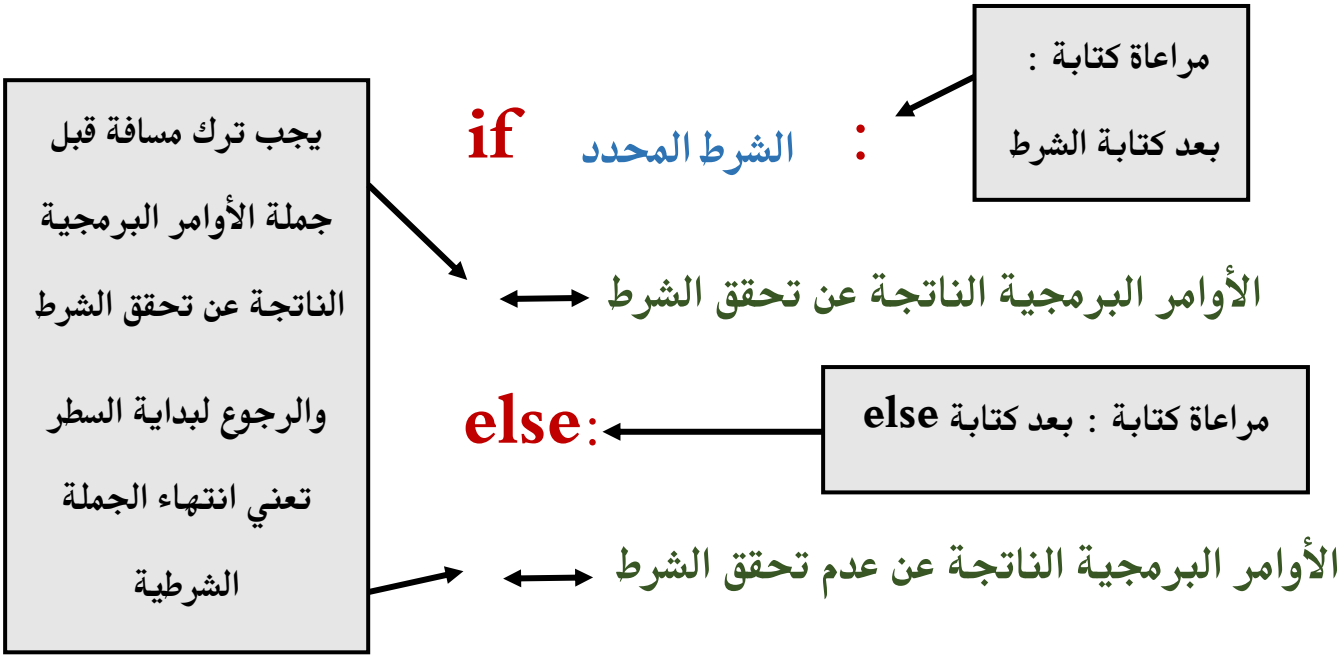
```
1  age=int(input("Enter youe age"))
2  if age>18:
3      print("Adult")
```

Shell

```
>>> %Run -c $EDITOR_CONTENT
 Enter youe age19
 Adult
>>>
```

➢ الجملة الشرطية الكاملة :

يتم استخدام الجملة الشرطية الكاملة لتنفيذ أوامر برمجية محددة بناءا على تحقق شرط معين أو في حالة عدم تحقق هذا الشرط ، يتم تنفيذ أوامر برمجية أخرى و تكون كالتالي :



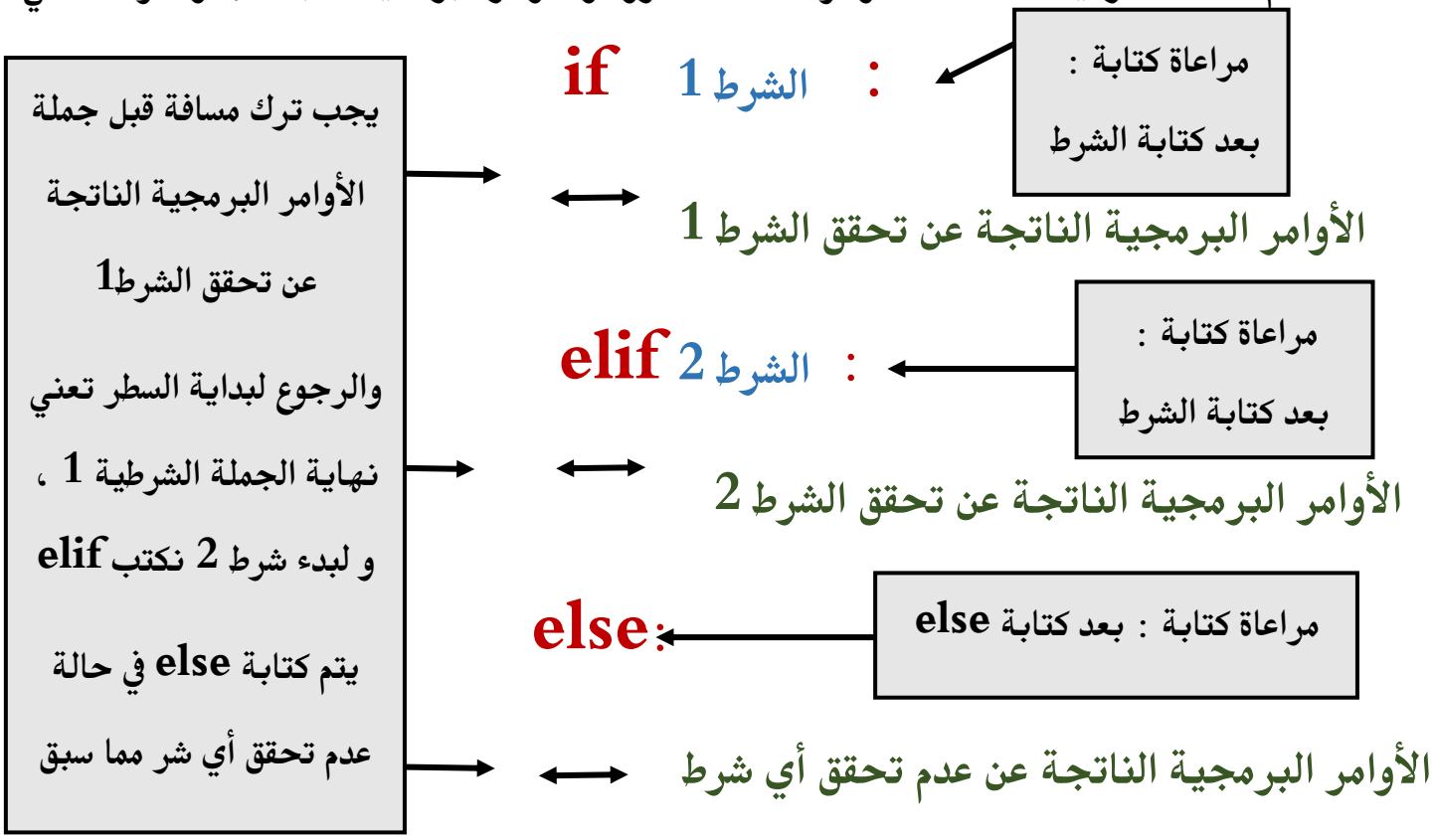
مثال : اكتبي برنامجاً بلغة **python** ، يطلب من المستخدم إدخال العمر ، عرض رسالة **Adult** إذا كانت العمر أكبر من **18** ، إذا كان العمر غير ذلك ، اعرض الرسالة **child** .

```
1  age=int(input("Enter youe age"))
2  if age>18:
3      print("Adult")
4  else:
5      print("Child")
```

```
Shell
>>> %Run -c $EDITOR_CONTENT
 Enter youe age15
 Child
>>>
```

➢ الجملة الشرطية المتداخلة :

نستخدام الجملة الشرطية المتداخلة عند وجود عدد من الشروط و الأوامر البرمجية التابعة لها و تكون كالتالي :



مثال : اكتبي برنامجاً بلغة python ، يطلب من المستخدم إدخال العمر ، عرض رسالة child إذا كانت العمر أصغر من 12 ، اعرض الرسالة child و إذا كان العمر أصغر من 18 اطبع الرسالة Teenager و إذا كان غير ذلك اطبع الرسالة Adult .

```
1  age=int(input("Enter youe age"))
2  if age<12:
3      print("child")
4  elif age<18:
5      print("Teenager")
6  else:
7      print("Adult")
```

```
Shell
>>> %Run -c $EDITOR_CONTENT
 Enter youe age17
 Teenager
>>>
```

الجملة التكرارية for

➢ الجملة التكرارية for في مجال محدد range :

تمكننا هذه الجملة البرمجية من تكرار تنفيذ أوامر برمجية ضمن مجال محدد من المرات ، و نتبع الطريقة التالية :

for متغير التكرار in range( ) :

⟷ جميع الأوامر البرمجية التي تقع أسفل الجملة التكرارية ، بعد وضع مسافة من السطر تكون تابعة لها ، والرجوع لبداية

```
1 for c in range(5,11):
2     print(c)
```

Shell

```
>>> %Run -c $EDITOR_CONTENT
5
6
7
8
9
10
>>>
```

➢ الجملة التكرارية for في المصفوفة :

يمكن أن نستخدم المجال range للجملة التكرارية في إدخال عدد محدد من العناصر للمصفوفة ، و نتبع الطريقة التالية :

اسم المصفوفة =[ ]

for متغير التكرار in range (عدد مرات الإدخال) :

⟷ جملة الإدخال

⟷ جملة إضافة العنصر للمصفوفة

مثال : اكتبي برنامج يقوم بإضافة 3 أعداد صحيحة إلى المصفوفة ls:

```
ls=[]                    # يجب تعريف المتغير قائمة
for i in range(3):  #تحديد المجال
    item=input("enter an integer:")  #إدخال عدد صحيح
    ls.append(int(item))             #إضافة العدد الصحيح إلى القائمة

print("The list entred is:",ls)      #عرض القائمة بعد إدخالها
```

```
enter an integer:1
enter an integer:2
enter an integer:3
The list entred is: [1, 2, 3]
```

➢ الجملة التكرارية **for** في المصفوفة :

تمكننا هذه الجملة البرمجية من قراءة محتوى القائمة و طباعتها أو لإجراء أوامر برمجية عليها ، و نتبع الطريقة التالية :

اسم المصفوفة = [, , , , ,]

for متغير التكرار in اسم المصفوفة ( ) :

طباعة متغير التكرار ⟷

مثال : اكتبي برنامج يقوم بطباعة عبارة **hello** قبل كل اسم من الأسماء المدرجة في المصفوفة التالية :

```
1 names=["amal","noor","sara","mona"]
```

الحل :

```
1 names=["amal","noor","sara","mona"]
2 for i in names:
3     print("hello",i)
```

Shell

```
>>> %Run -c $EDITOR_CONTENT
 hello amal
 hello noor
 hello sara
 hello mona
>>>
```

➢ الجملة التكرارية **for** في متغير نصي **string** :

يمكن استخدام متغير القيمة النصية ليكون هو المجال الخاص بالجملة التكرارية ، لنتمكن من طباعة كل حرف على حده أو التعامل معها في مختلف الأوامر البرمجية .

```
1  Sname="sameera"
2  for i in Sname:
3      print(i)
```

Shell

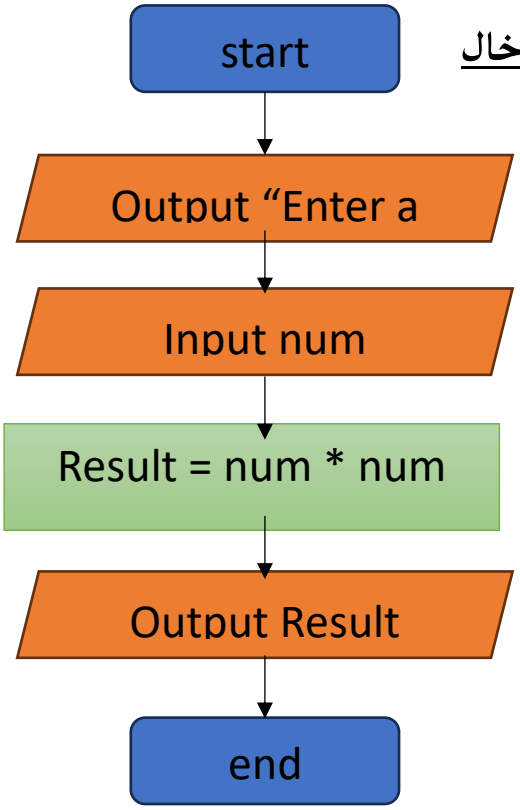
```
>>> %Run -c $EDITOR_CONTENT
s
a
m
e
e
r
a
>>>
```

الخرائط التدفقية Flow Charts

الخرائط التدفقية هي عبارة عن حل رسومي للمشكلة باستخدام مجموعة من الأشكال الهندسية ببعضها البعض في ترتيب منطقي لتسلسل الإجراءات البرمجية ، و يمكن تحويلها لكود برمجي أو العكس ، و يرمز كل شكل من الأشكال الهندسية إلى نوع الإجراء البرمجي .

الشكل	الاسم: عربي/English		الوصف
	البداية/النهاية	Start/End	يستخدم في بداية ونهاية الخريطة التدفّقيّة
	مدخلات/مخرجات	Input/output	يستخدم عند إدخال المعطيات و/أو عرض المخرجات
	معالجة	process	عمليات حسابية | منطقيّة، تعليمة برمجيّة
	اتخاذ القرار	Decision	عندما يكون هناك اجراء سيتّخذ بناء على شرط نتيجته (نعم/لا)
	الاتجاه	Flow arrow	يبيّن اتجاه الارتباط بين مختلف أشكال الخريطة التدفّقيّة

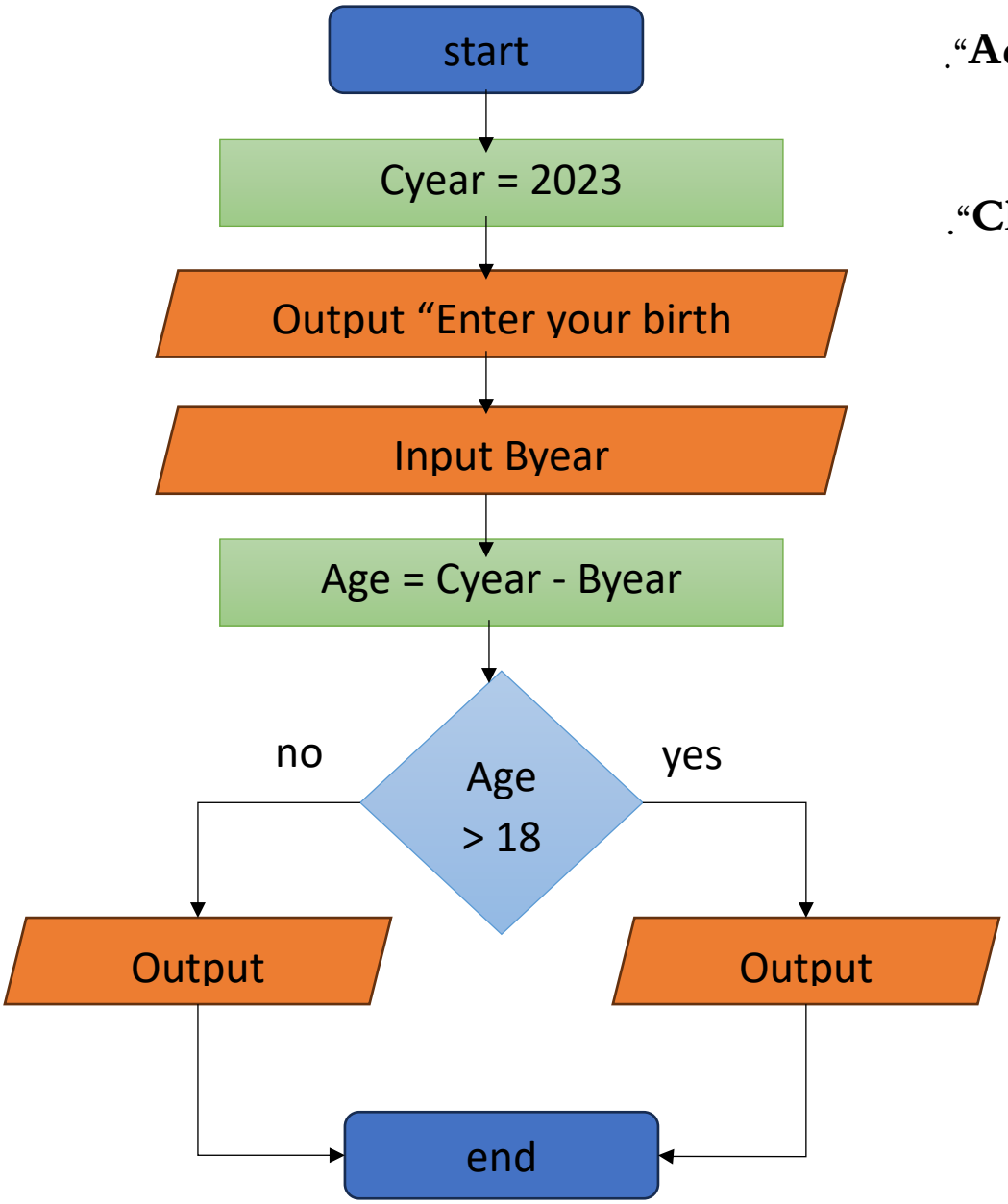
مثال1 : أرسمي خريطة تدفقية لإدخال عدد و إيجاد حاصل ضرب العدد في نفسه، ثم طباعة الناتج .



إعداد : أ. لطيفة عبدالسلام ، أ. منيرة عبدالسلام
مديرة المدرسة : أ. أمل يوسف مال الله

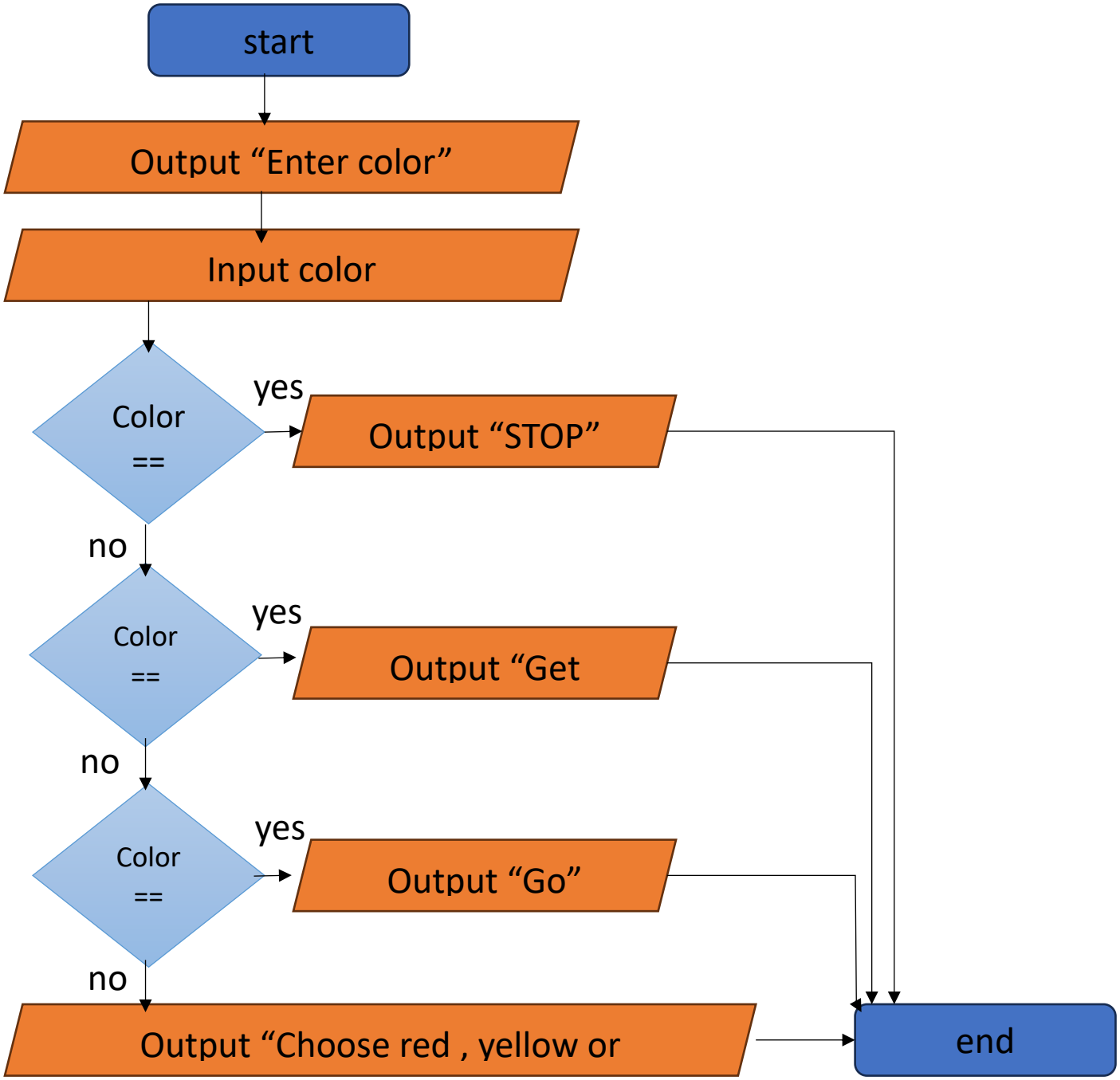
مثال2/ ارسمي خريطة تدفقية لحساب عمر الشخص وتحديد إذا كان الشخص بالغ أم طفل من خلال الخطوات التالية:

1. عرفي متغير باسم **Cyear** واجعلي قيمته **2023** ويمثل السنة الحالية.
2. أدخلي سنة ميلاد الشخص واحفظيها في متغير **Byear**.
3. أوجدي عمر الشخص من خلال العملية التالية: **age = Cyear – Byear**
4. إذا كان عمر الشخص أكبر من **18**:
5. اطبعي عبارة "**Adult**".
6. إذا كان غير ذلك:
7. اطبعي عبارة "**Child**".



مثال3/ حولي الكود البرمجي التالي إلى خريطة تدفقية:

```
color=input("Enter red, yellow or green color only!")
if color == "red":
    print("STOP")
elif color == "yellow":
    print("GET READY")
elif color == "green":
    print("GO")
else :
    print("chosse red, yellow or green color only!")
```



مثال4/ صممي خريطة تدفقية Flow Chart تمكنك من عرض مضاعفات الرقم 3 الآتية :

```
OUTPUT:
3
6
9
12
```

الحل البرمجي :

```
1  for i in range(3,13,3):
2      print(i)
```

الحل بالخريطة التدفقية :