BÀI SỐ 3. MODULE TRONG PYTHON

- **Kiến thức:**

- Module trong trong python
- Bản chất của câu lệnh import – tạo và sử dụng module
- Module random trong python

- **Kỹ năng:**

- Làm việc với ngôn ngữ python, kĩ năng làm việc nhóm.
- Tạo ra module và sử dụng bằng câu lệnh import
- Luyện tập kỹ năng thuyết trình qua việc ôn tập cuối buổi học.

- **Sản phẩm: Trò chơi chiếc nón kỳ diệu**

- **Phân bổ thời gian:**

Nội dung	Thời gian
1. Module là gì? **2.** Tạo module như thế nào? **3.** Giới thiệu module random	**1h**
4. Bài tập: + Hướng dẫn + Làm bài + Thuyết trình **5. Tổng kết đánh giá**	**1h**

1. Module là gì?

Bài trước, các em đã được tìm hiểu về hàm và cách sử dụng hàm trong chương trình của python. Tuy nhiên, các em nghĩ sao nếu 1 chương trình mà chúng ta viết quá nhiều hàm như ví dụ sau:

Hàm 1

Hàm 2

Hàm n…

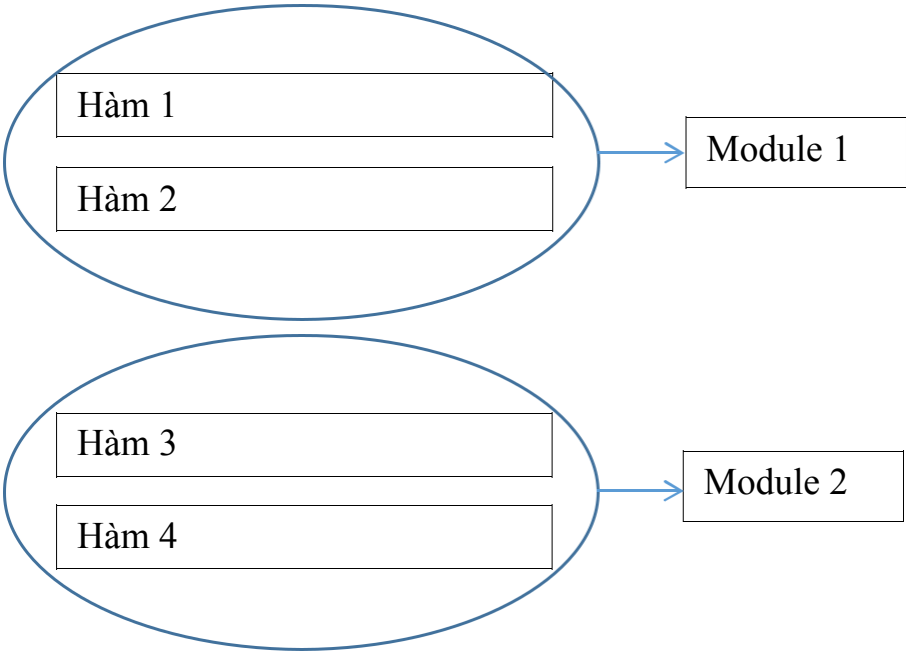
 # code khác….

Ví dụ 1.

```
1   def add(a, b):
2       print "ADDING %d + %d" % (a, b)
3       return a + b
4
5   def subtract(a, b):
6       print "SUBTRACTING %d - %d" % (a, b)
7       return a - b
8
9   def multiply(a, b):
10      print "MULTIPLYING %d * %d" % (a, b)
11      return a * b
12
13  def divide(a, b):
14      print "DIVIDING %d / %d" % (a, b)
15      return a / b
16
17
18  print "Let's do some math with just functions!"
19
20  age = add(30, 5)
21  height = subtract(78, 4)
22  weight = multiply(90, 2)
23  iq = divide(100, 2)
24
25  print "Age: %d, Height: %d, Weight: %d, IQ: %d" % (age, height, weight, iq)
26
27
28  # A puzzle for the extra credit, type it in anyway.
29  print "Here is a puzzle."
30
```

Thưa Cô, nếu chương trình nhỏ, thì nhìn còn gọn **nhưng** nếu chương trình lớn, viết nhiều hàm trông lộn xộn và khó nhìn lắm Cô ạ!

Chính xác, việc sinh ra hàm đã giúp ta giải quyết nhiều vấn đề rồi. Nhưng nếu chương trình viết quá nhiều hàm thì sẽ làm cho chương trình chúng ta khó nhìn. Để quản lý các hàm, trong python đưa ra giải pháp đó là chúng ta có thể gom tất cả **các hàm có cùng tính chất** thành 1 file, và có thể sử dụng lại file này khi cần. Cách giải quyết này gọi là **Module.**



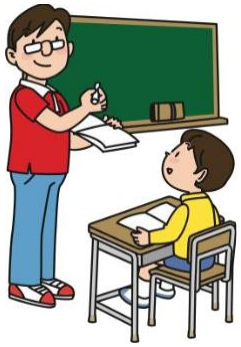
Hiểu 1 cách đơn giản, Module là cách mà chúng ta phân hóa chương trình ra các **nhánh nhỏ** cho **dễ quản lý** và gọi lại chúng khi nào cần, như thế chương trình của chúng ta sẽ có tính tái sử dụng và dễ chỉnh sửa khi cần.

2. Tạo module như thế nào?

2.1. Tạo Module

Cô có ví dụ sau nhé:

Ví dụ 2. Sử dụng hàm	Kết quả
#..... Khai báo các hàm---- **def add(a, b):** print ("ADDING %d + %d" % (a, b)) return a + b **def subtract(a, b):** print ("SUBTRACTING %d - %d" % (a, b)) return a - b **def multiply(a, b):** print ("MULTIPLYING %d * %d" % (a, b)) return a * b **def divide(a, b):** print ("DIVIDING %d / %d" % (a, b)) return a / b *#..... chương trình chính----* age = add(30, 5) height = subtract(78, 4) weight = multiply(90, 2) iq = divide(100, 2) print ("Age: %d, Height: %d, Weight: %d, IQ: %d" % (age, height, weight, iq))	**ADDING** 30 + 5 **SUBTRACTING** 78 - 4 **MULTIPLYING** 90 * 2 **DIVIDING** 100 / 2 **Age:** 35, **Height**: 74, **Weight:** 180, **IQ:** 50



Trong ví dụ 2 trên, đây là cách viết thông thường mà các em đã được học. Bây giờ chúng ta sẽ tìm hiểu cách tạo **module** nhé!

Bước 1. Các em tạo 1 thư mục bất kỳ lưu tại ổ đĩa C (tùy ý)

Bước 2. Các em mở chương trình python IDL (hoặc Wing IDE,…) và tạo 1 file python mới với tên là: ***my_math.py*** (lưu file vào thư mục tạo ra tại bước 1), trong file có chứa các dòng lệnh sau:

```
#..... Khai báo các hàm----
def add(a, b):
    print ("ADDING %d + %d" % (a, b))
    return a + b
def subtract(a, b):
    print ("SUBTRACTING %d - %d" % (a, b))
    return a - b
def multiply(a, b):
    print ("MULTIPLYING %d * %d" % (a, b))
    return a * b
def divide(a, b):
    print ("DIVIDING %d / %d" % (a, b))
    return a / b
```

Bước 3, tạo 1 file khác trong python, lưu lại file vào thư mục tạo ra ở bước 1 với tên file là: *my_main.py*:

```
import  my_math
age = my_math.add(30, 5)
height = my_math.subtract(78, 4)
weight = my_math.multiply(90, 2)
iq = my_math.divide(100, 2)
print ("Age: %d, Height: %d, Weight: %d, IQ: %d" % (age, height, weight, iq))
```

Bước 4, các em mở file: my_main.py lên và chay. Kết quả sẽ là:

```
1>>> [evaluate my_main.py]
     ADDING 30 + 5
     SUBTRACTING 78 - 4
     MULTIPLYING 90 * 2
     DIVIDING 100 / 2
     Age: 35, Height: 74, Weight: 180, IQ: 50
```

Phân tích ví dụ trên:

Trong cách làm trên, Cô tách làm 2 file, file 1: *my_math.py* là file chứa các hàm liên quan đến toán học; file 2: *my_main.py* là file để gọi file *my_math.py* và thực hiện các công việc. Cả hai file này phải được đặt cùng trong 1 thư mục.

Tại file 2: *my_main.py* muốn sử dụng các hàm trong file 1 ta phải sử dụng câu lệnh khai báo, theo cú pháp sau:

import module1, module2,..

Trong đó: **module1, module2**,... là các modules mà các em muốn import vào file hiện tại.

Ví dụ: **import my_math**

Sau khi khai báo, muốn sử dụng các hàm của module: *my_math*, chung ta phải gọi theo cú pháp sau:

Tên_module.Tên_hàm

Ví dụ:

```
age = my_math.add(30, 5)
height = my_math.subtract(78, 4)
weight = my_math.multiply(90, 2)
iq = my_math.divide(100, 2)
```

Em đã hiểu cách sử dụng Modules rồi! đơn giản chỉ cần:

- Tạo ra các file khác nhau, đặt cùng 1 thư mục.
- Muốn sử dụng file nào thì phải: **import** module vào
- Muốn sử dụng hàm nào thì chỉ cần gọi tên **module.ten_ham**

Cô ơi! Bây giờ nếu em muốn sử dụng 1 hàm trong module chèn vào mà không cho phép dùng hàm khác được không ạ? Ví dụ, ở ví dụ 2 trên em chỉ muốn cho phép dùng hàm **add().**

Tất nhiên là được rồi!... chúng ta chuyển sang phần 2.2. nhé!

2.2. Sử dụng from – import

Giả sử trong một trường hợp nào đó khi các em không muốn sử dụng hết toàn bộ module mà chỉ muốn sử dụng một số thứ trong đó mà thôi thì sẽ phải làm sao? Trong trường hợp này chúng ta sử dụng từ khóa **from...import** theo cú pháp như sau:

from modules import something, something2,...

Trong đó: something1,somthing2,... là những thứ mà các em muốn sử dụng ở trong modules. Nếu như các em muốn import tất cả những gì trong modules có và cho phép thì sử dụng keyword *.

Ví dụ 3. Cô chỉ muốn chèn hàm **add** trong module: *my_math*, thì Cô khai báo như sau:

Ví dụ 3. Dùng 1 hàm	Kết quả
from my_math **import** add age = **add(30, 5)** ← Chú ý: chỉ nhập **add(30,5)** không nhập my_math.add(30,5) print (" Tuoi: ", age)	ADDING 30 + 5 Tuoi: 35

Ví dụ 4. Dùng nhiều hàm sử dụng: *	Kết quả
from my_math **import** * print(**add(30, 5)**) print(**subtract(30, 5)**) print(**multiply(30, 5)**) 	ADDING 30 + 5 35 SUBTRACTING 30 - 5 25 MULTIPLYING 30 * 5 150

2.3. Định danh cho Module

Nếu như trong trường hợp tên module của chúng ta rất khó nhớ hay dài hay vì một lý do nào khác mà các em không muốn gọi module như thế thì với Python cũng có thể gán định danh mới cho module khi import chúng bằng cách sử dụng từ khóa: **as** với cú pháp như sau:

import *tên_module* **as** *tên _ định _danh*

 # hoặc đối với from import

from *tên_module* **import** *hàm* **as** *tên _ định _danh*

Ví dụ 4. Định danh cho module đối với import	Kết quả
import *my_math* **as toan** age =**toan.add(30, 5)** height = **toan.subtract(78, 4)** weight = **toan.multiply(90, 2)** iq = **toan.divide(100, 2)** print ("Age: %d, Height: %d, Weight: %d, IQ: %d" % (age, height, weight, iq))	ADDING 30 + 5 SUBTRACTING 78 - 4 MULTIPLYING 90 * 2 DIVIDING 100 / 2 Age: 35, Height: 74, Weight: 180, IQ: 50

Ví dụ 5. Định danh cho module đối với from .. import..	Kết quả
from my_math **import** add **as cong** age = **cong(30, 5)** print (" Tuoi: ", age)	ADDING 30 + 5 Tuoi: 35

Thưa Cô! Em đã hiểu cách tạo Module của riêng mình. Nhưng có cách nào dùng các Module có sẵn của Python không Cô?

Trong Python hỗ trợ nhất nhiều modules. Trong phần tiếp theo Cô sẽ giới thiệu module **random** trong python, các module khác chúng ta sẽ tìm hiểu trong các bài sau nhé.

3. Giới thiệu module random

Sự ngẫu nhiên luôn xảy ra ở xung quanh chúng ta. Khi các em tung một đồng xu hoặc một con súc sắc, các em không bao giờ có thể chắc chắn về kết quả cuối cùng.

Trong Python cung cấp cho chúng ta module: *random*. Các em có thể sử dụng để tạo ra trong trường hợp muốn tạo ra một giá trị ngẫu nhiên nào đó.

- **Tạo số ngẫu nhiên nhỏ hơn 1 số cho trước với hàm:** `randrange(a)`

Ví dụ 6.	Kết quả
`import random` `random.randrange(100)`	sinh ra một số ngẫu nhiên nhỏ hơn 100

- **Tạo số ngẫu nhiên với:** `randrange(a, b[,step])`

Ví dụ 7.	Kết quả
`import random` `random.randrange(0, 100, 3)`	sinh ra một số ngẫu nhiên trong khoảng từ 0 đến 100 và chia hết cho 3

- **Tạo số nguyên (int) ngẫu nhiên với:** `randint(a, b)`

Ví dụ 8.	Kết quả
`import random` `random.randint(1,6)`	sinh ra một số nguyên ngẫu nhiên trong khoảng từ 1 đến 6

- **Tạo số thực (float) ngẫu nhiên với:** `uniform(a, b)`

Ví dụ 8.	Kết quả
`import random` `random.uniform(1, 20)`	Sinh ra một số thực ngẫunhiêntrong khoảng từ 1 đến 20

- **Sử dụng module random với kiểu dữ liệu List**

Stt	Các hàm	Tác dụng
1	`choice(list)`	Trả về 1 giá trị ngẫu nhiên trong list
2	`shuffle(list)`	Trả về 1 danh sách đã xáo trộn ngẫu nhiên
3	`sample(list, n)`	Trả về n phần tử duy nhất trong list đã cho

Vi dụ 10:

```
01  import random
02
03  ids = [1, 8, 10, 12, 15, 17, 25]
04
05  random.choice(ids)           # returns 8
06  random.choice(ids)           # returns 15
07
08  names = ['Tom', 'Harry', 'Andrew', 'Robert']
09
10  random.choice(names)         # returns Tom
11  random.choice(names)         # returns Robert
12
13  random.shuffle(names)
14  names
15  # returns ['Robert', 'Andrew', 'Tom', 'Harry']
16
17  random.sample(names, 2)
18  # returns ['Andrew', 'Robert']
19
20  random.sample(names, 2)
21  # returns ['Tom', 'Robert']
22
23  names
24  # returns ['Robert', 'Andrew', 'Tom', 'Harry']
```

TỔNG KẾT BÀI HỌC

Xin chúc mừng các em đã hoàn thành bài học số 3, trong bài này, các em cần chú ý những nội dung sau:

1. Module là cách mà chúng ta phân hóa chương trình ra các **nhánh nhỏ** cho **dễ quản lý** và gọi lại chúng khi nào cần, như thế chương trình của chúng ta sẽ có tính tái sử dụng và dễ chỉnh sửa khi cần.
2. Câu lệnh **import:** Để sử dụng 1 module, chúng ta sử dụng câu lệnh **import.**
3. Câu lệnh **from … import:** Câu lệnh "*from … import*" cho phép ta import thuộc tính, hàm từ một module.
4. Câu lệnh **from … import *:** Câu lệnh *from … import* * cho phép ta import tất cả các thuộc tính, hàm, class của một module
5. Để định danh cho module sử dụng từ khóa

   as: **import** *tên_module* **as** *tên _ định _danh*

   # hoặc đối với from import

   from *tên_module* **import** *hàm* **as** *tên _ định _danh*

6. Modun random: Là module chứa các hàm giúp ta tạo các số ngẫu nhiên hoặc 1 dãy số ngẫu nhiên với các hàm:
`randrange()`,**`randint()`**,`uniform()`,…

Nội dung bài học tiếp theo: Turtle Graphics

Turtle là một thư viện đồ họa trong Python, giúp chúng ta tạo các ứng dụng, game với giao diện đồ họa đẹp mắt.

.

BÀI TẬP

Bài 1. Dòng nào **SAI** khi mô tả về module?

A. Khi tạo ra một module, ta phải để file đó cùng với thư mục với file chương trình muốn sử dụng module đó.

B. Các hàm được định nghĩa trong module đều phải có return.

C. Muốn sử dụng module phải viết chính xác tên module đó sau từ khóa import.

D. Gọi hàm một hàm trong module cần phải có tên module, dấu chấm và sau đó mới đến tên hàm.

Lựa chọn đáp án: ……

Bài 2. Câu lệnh nào sẽ tạo ra một số ngẫu nhiên từ 1 đến 10 :

A. Num = random.randint(1,100)%10

B. Num = randint(1,10)

C. Num = random.int(1,100)

D. Num = random.int(1,10)

Lựa chọn đáp án: …….

Bài 3. Các em hãy cho biết kết quả của chương trình sau:

```
1   import random
2   num = random.randint(1,100)
3   i = 0
4   while True:
5       guessNum = int (input ("Enter your guessing number: (1-->100)"))
6       i += 1
7       if guessNum == num:
8           print ("You guessed right")
9           print ("Try time number: ",i)
10          break
11      elif guessNum < num:
12          print("Try higher")
13      else:
14          print("Try lower")
```

GV:Quỳnh Vi **Bài 4. TRÒ CHƠI CHIẾC NÓN KỲ DIỆU**

Em hãy viết chương trình chiếc nón kỳ diệu, yêu cầu tạo 2 file: diem.py và my_main.py.

1. Module: *diem.py*, trong module này chứa các hàm điểm mà người chơi quay được, ví dụ:

```
def one():
  print("""
   1111111
  1:::::::1
 1:::::::1
 111:::::1
    1::::1
    1::::1
    1::::1
    1::::1
    1::::1
    1::::1
    1::::1
    1::::1
 111::::::111
 1::::::::::1
 1::::::::::1
 111111111111
   """)
```

```
def two():
  print("""
  222222222222222
 2:::::::::::::::22
 2::::::222222:::::2
 2222222      2:::::2
              2:::::2
              2:::::2
           2222::::2
      22222::::::22
    22::::::::222
   2:::::22222
  2:::::2
  2:::::2
  2:::::2        222222
  2::::::2222222:::::2
  2:::::::::::::::::::2
  22222222222222222222
   """)
```

• • •

```
def ten():
  print("""
   1111111       000000000
  1::::::1     00:::::::::00
 1:::::::1   00:::::::::::::00
 111:::::1  0:::::::000:::::::0
    1::::1  0::::::0   0::::::0
    1::::1  0:::::0     0:::::0
    1::::1  0:::::0     0:::::0
    1::::1  0:::::0     0:::::0
    1::::1  0:::::0     0:::::0
    1::::1  0:::::0     0:::::0
    1::::1  0:::::0     0:::::0
    1::::1  0::::::0   0::::::0
 111::::::1110:::::::000:::::::0
 1::::::::::1 00:::::::::::::00
 1::::::::::1   00:::::::::00
 111111111111     000000000
   """)
```

2. Trong file chương trình: *my_main.py* sẽ **import** *diem.py*; trước khi đưa câu hỏi trắc nghiệm, người chơi phải quay điểm; sau đó, cho người chơi sẽ phải trả lời câu hỏi trắc nghiệm. Nếu trả lời đúng tính điểm, trả lời sai không tính điểm. Mỗi người chơi sẽ phải trả lời 3 câu trắc nghiệm. kết thúc đưa ra tổng điểm mà người chơi đạt được.

Gợi ý: Sử dụng module random để tạo điểm ngẫu nhiên từ 1-10. Để tạo sự hiệu ứng đẹp mắt, tương ứng với điểm quay được sẽ gọi hàm tương ứng trong module: *diem.py*

BÀI TẬP VỀ NHÀ

Các em hãy phát triển bài tập 4 ở trên, để tăng hiệu ứng trò chơi các em hãy thêm module âm thanh khi quay điểm và trả lời câu hỏi. Nếu trả lời đúng 1 âm thanh xuất hiện, nếu trả lời sai 1 âm thanh báo sai xuất hiện.

Gợi ý: Sử dụng module âm thanh winsound

```
# chèn module âm thanh trong python
import winsound
# chơi 1 bản nhạc được copy vào cùng thư mục file chương trình
winsound.PlaySound('Home_Run.wav', winsound.SND_FILENAME)
```

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....