SỞ GD & ĐT TP. HỒ CHÍ MINH
TRƯỜNG THPT LÝ THƯỜNG KIỆT

ĐỀ CƯƠNG KIỂM TRA GIỮA HỌC KÌ II
NĂM HỌC 2024 – 2025
Môn: Tin học, Lớp 10

CHỦ ĐỀ 5. GIẢI QUYẾT VẤN ĐỀ VỚI SỰ TRỢ GIÚP CỦA MÁY TÍNH

BÀI 18: CÁC CÂU LỆNH VÀO RA ĐƠN GIẢN

A. TÓM TẮT NỘI DUNG

1. CÁC LỆNH VÀO RA ĐƠN GIẢN

– Chức năng: Hàm input() là có chức năng **nhận dữ liệu nhập từ bàn phím** vào Python và trả về kết quả dưới dạng kiểu **xâu string(str).**

– Cú pháp:

<Tên biến> = input(<Dòng thông báo>)

– Trong đó:

- <Tên biến>: *Là biến lưu trữ dữ liệu người dùng nhập vào từ bàn phím.*
- [Dòng thông báo]: *Là một chuỗi thông báo cho người dùng biết nhập dữ liệu cần nhập, chuỗi này có thể là* ***rỗng****.*

– Ví dụ:

Chương trình	Kết quả biên dịch
`name = input("Nhập họ tên em: ")` `print("Xin chào ",name)`	`Nhập họ tên em: Nguyễn Văn Toán` `Xin chào  Nguyễn Văn Toán`

➲ **Lưu ý:**

– Khi người dùng nhấn phím Enter việc nhập dữ liệu từ bàn phím sẽ kết thúc.

– Giá trị trả về của hàm input() là một xâu, do đó, ta muốn chuyển kiểu từ kiểu xâu sang kiểu dữ liệu khác như (kiểu số nguyên, số thực …) thì ta phải kết hợp thêm hàm chuyển đổi kiểu dữ liệu

2. CÁC HÀM CHUYỂN ĐỔI KIỂU DỮ LIỆU

– Chúng ta có thể chuyển đổi qua lại giữa các kiểu dữ liệu số và chuỗi thông qua các hàm được Python tích hợp sẵn như **int(), float() và str()**. Điều này còn được gọi là ép kiểu.

Lệnh int ( )	có chức năng chuyển đổi số thực hoặc xâu chứa số nguyên thành số nguyên

Chuyển đổi	Chương trình	Kết quả thông dịch
1. Số thực	`>>> int(12.6)`	12
2. Xâu số nguyên	`>>> int("123")`	123 # Chuyển đổi được
3. Xâu số thực	`>>> int("10.35")`	**# LỖI:** Không chuyển đổi được

Traceback (most recent call last): File "<pyshell#21>", line 1, in <module> int("10.35")
ValueErrpr : invalid literal for int( ) with base 10: "10.35"

Lệnh float ( )	dùng để chuyển đổi số nguyên và xâu kí tự thành số thực	
Chuyển đổi	Chương trình	Kết quả thông dịch
1. Số nguyên	**>>> float(8)**	8.0
2. Xâu số	**>>> float("10.23")**	10.23

Lệnh str ( )	dùng để chuyển đổi các kiểu dữ liệu khác thành xâu kí tự	
Chuyển đổi	Chương trình	Kết quả thông dịch
1. Biểu thức	**>>> str(12+34)**	'46' # Tính tổng rồi chuyển thành xâu
	>>> str(2>3)	Flase
2. Số	**>>> str(12.567)**	'12.567'

Những trường hợp Không chuyển đổi được:	Chương trình
1. Hàm int() không chuyển đổi được xâu số thực	**>>> int("10.35")**
2. Hàm int(), float() không chuyển đổi được xâu là biểu thức toán	**>>> int("12+45")** **>>> float("12+45")**

– **Chuyển đổi kiểu cho giá trị trả về của hàm input( ) thành kiểu nguyên, thực**

1. Số nguyên	**>>> n = int(input("Nhập số tự nhiên: "))** Nhập số tự nhiên: 13
2. Số thực	**>>> x = float(input("Nhập số thực x: "))** Nhập số thực x: 7.5

– **Nhập nhiều số trên một dòng**

Bước 1: Nhập cả dòng bằng hàm input. >>> s = input('Nhập 3 số nguyên: ') Nhập 3 số nguyên: 10 20 30	Kết quả s là một xâu kí tự >>> print(s) '10 20 30'
Bước 2: Dùng hàm split để tách các số trong xâu input ra. >>> a = s.split()	Kết quả a là một danh sách (kiểu **list**) gồm các phần tử được tách ra từ xâu s >>> print(a) ['10', '20', '30']
Bước 3: Sử dụng map để ép các xâu được tách ra trong input sang số int hoặc float tùy theo đầu bài.	Kết quả x = 10, y = 20, z = 30 >>> print(x,y,z)

```
>>> x, y, z = map(int,a)                    10 20 30
                                        >>> print('Tổng 3 số: ', x+y+z)
                                            Tổng 3 số: 60
```

Gộp 3 bước trên ta có câu lệnh

```
>>> x, y, z = map(int,input('Nhập 3 số nguyên: ').split())
    Nhập 3 số : 10 20 30
>>> print('Tổng 3 số x, y, z: ', x + y + z)
    Tổng 3 số x, y, z:  60
```

B. CÂU HỎI TRẮC NGHIỆM

Câu 1: int là kiểu dữ liệu nào sau đây ?
A. Kiểu số nguyên **B.** Kiểu số thực **C.** Kiểu xâu ký tự **D.** Kiểu logic

Câu 2: float là kiểu dữ liệu nào sau đây?
A. Kiểu số nguyên **B.** Kiểu số thực **C.** Kiểu xâu ký tự **D.** Kiểu logic

Câu 3: Kiểu dữ liệu của biểu thức nào sau đây không thuộc kiểu bool?
A. 67>78 **B.** 17 !=12+3 **C.** 14 == 10+2 **D.** '12=10+2 '

Câu 4: str là kiểu dữ liệu nào sau đây?
A. Kiểu số nguyên **B.** Kiểu số thực **C.** Kiểu xâu ký tự **D.** Kiểu logic

Câu 5: Hàm nào sau đây là hàm kiểm tra kiểu dữ liệu ?
A. Style **B.** Tyle **C.** Type **D.** type

Câu 6: Cho biến s = 'Xin chào', Python sẽ cung cấp cho biến s kiểu dữ liệu nào?
A. string B. bool C. float D. str

Câu 7: Biết x có giá trị là 5.6, vậy x thuộc kiểu giá trị nào trong các kiểu sau?
A. bool B. int C. float D. str

Câu 8: Cho x = -49 và y = 4.9 Python cung cấp kiểu dữ liệu nào cho biến x và y?
A. x là int, y là float B. x là float, y là int C. x, y là int D. x, y là float

Câu 9: Hãy cho biết kiểu dữ liệu trả về của biến a sau câu lệnh? a = float(5)
A. Kiểu nguyên B. Kiểu thực C. Kiểu kí tự D. Kiểu logic

Câu 10: Cho x = 3.4 và y = int(x). Biến y nhận giá trị là:
A. 3 B. 4 C. 3.4 D. Báo lỗi

C. CÂU HỎI ĐÚNG/SAI

Câu 1: Trong Python, câu lệnh input() được sử dụng để nhận dữ liệu từ người dùng, còn câu lệnh print() dùng để hiển thị dữ liệu ra màn hình.

a) Câu lệnh input() luôn trả về dữ liệu dưới dạng chuỗi (str).

b) Câu lệnh print() có thể hiển thị nhiều giá trị cùng lúc bằng cách sử dụng dấu phẩy ",".

c) Hàm input() có thể nhận nhiều giá trị cùng lúc từ người dùng mà không cần xử lý thêm.

d) Có thể sử dụng tham số sep trong print() để tùy chỉnh ký tự phân cách giữa các giá trị được in ra.

Câu 2: Câu lệnh input() có thể được kết hợp với các kiểu dữ liệu khác nhau để xử lý dữ liệu đầu vào.

a) Dữ liệu từ input() luôn cần được chuyển đổi sang kiểu số nếu muốn thực hiện phép toán số học.

b) Có thể sử dụng input() để nhập một danh sách (list) trực tiếp từ người dùng mà không cần xử lý thêm.

c) Dữ liệu nhập vào từ input() có thể được ép kiểu sang int hoặc float bằng cách sử dụng int(input()) hoặc float(input()).

d) Câu lệnh input() có thể nhận tham số để hiển thị lời nhắc nhập dữ liệu.

Câu 3: Câu lệnh print() có nhiều tùy chọn để định dạng đầu ra.

a) Có thể sử dụng print(f"Xin chào, {name}!") để chèn giá trị của biến name vào chuỗi in ra.

b) Tham số end trong print() dùng để kiểm soát ký tự kết thúc dòng in ra, mặc định là dấu xuống dòng (\n).

c) print() không thể in một danh sách (list) hoặc từ điển (dictionary).

d) Có thể sử dụng toán tử + để nối nhiều chuỗi khi sử dụng print(), nhưng không thể dùng dấu phẩy ",".

BÀI 19: CÂU LỆNH RẼ NHÁNH IF

A. TÓM TẮT NỘI DUNG

1. RẼ NHÁNH

– Một việc được thực hiện hay không phụ thuộc vào một điều kiện thì ta thể hiện bằng câu mệnh đề có dạng điều kiện:

  - Dạng thiếu: **Nếu … thì …**

    Ví dụ 1: Nếu a chia hết cho 2 thì a là số chẵn

  - Dạng đủ: **Nếu … thì … ngược lại…**

    Ví dụ 2: Nếu a chia hết cho 2 thì a là số chẵn ngược lại a là số lẻ

– Trong lập trình, cấu trúc dùng để mô tả các mệnh đề có dạng như trên gọi là: **cấu trúc rẽ nhánh thiếu và đủ**

2. LỆNH IF

a) Câu lệnh rẽ nhánh dạng thiếu

– Cú pháp:

```
if <điều kiện>:
    <Khối lệnh>
```

Sau điều kiện cần có dấu hai chấm "**:**"

Nhóm, khối lệnh tiếp theo cần **viết lùi vào** (bằng **1 Tab** hay **4 dấu cách**) và thẳng hàng

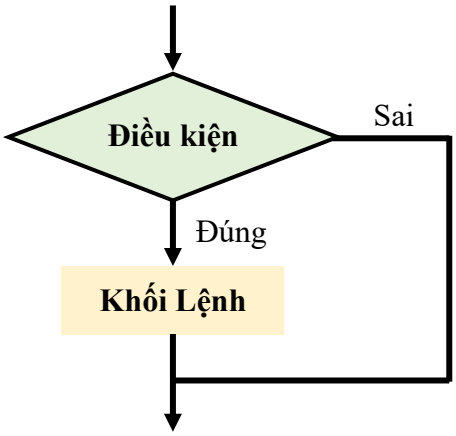
– Trong đó:

  - **<Khối lệnh>**: là một câu lệnh (không cần xuống hàng mà viết sau dấu "**:**") hoặc nhiều câu lệnh được viết lùi vào (bằng **1 Tab** hay **4 dấu cách**) và thẳng hàng.

– **<Điều kiện>:** là biểu thức quan hệ hoặc biểu thức logic.

– Thực hiện lệnh:

Python sẽ kiểm tra `<điều kiện>` **nếu** đúng **thì** thực hiện `<khối lệnh>`, **ngược lại** thì bỏ qua chuyển sang lệnh tiếp theo sau lệnh if.



Hình 19.2 Cấu trúc rẻ nhánh dạng thiếu

– Chạy chương trình cho biết kết quả xuất ra màn hình:

Câu 1:

```
x, y = 5, 10
if x > y:
    x+=1
y+=1
print('x =', x, '; y =', y)
```

A. x = 6 ; y = 10 **B.** x = 5 ; y = 10

C. x = 5 ; y = 11 **D.** x = 6 ; y = 11

Câu 2:

```
x, y = 5, 10
if x <= y:
    x+=1
y+=1
print('x =', x, '; y =', y)
```

A. x = 6 ; y = 10

B. x = 5 ; y = 10

C. x = 5 ; y = 11

D. x = 6 ; y = 11

– VD: Dùng cấu trúc rẻ nhánh dạng thiếu để mô tả các mệnh đề rẽ nhánh sau:

(1) Nhập diem Nếu diem >= 19.5 thì xuất ra thông báo “đậu THPT LTK”	…… …… ……
(2) Nhập DTB Nếu 6.5 <= DTB <8.0 thì xuất ra thông báo “học lực khá”	…… …… ……
(3) Nếu a chia hết cho 2 thì a là số chẵn	…… ……
(4) Nếu a không chia hết cho 2 thì a là số lẻ	…… ……

b) Câu lệnh rẽ nhánh dạng đủ

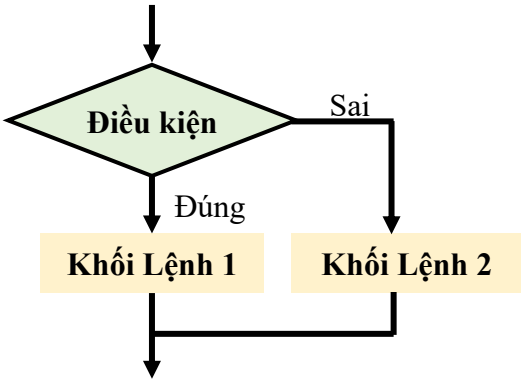
– Cú pháp:

```
if <điều kiện>:
     <Khối lệnh 1>
else:
     <Khối lệnh 2>
```

- Từ khóa if và else cần viết thẳng lề trái
- Các khối lệnh 1 và khối lệnh 2 cần **viết lùi vào** (bằng **1 Tab** hay **4 dấu cách**) và thẳng hàng

– Thực hiện lệnh:

Python sẽ kiểm tra `<điều kiện>` **nếu** đúng **thì** thực hiện `<khối lệnh 1>`, **ngược lại** thì thực hiện `<khối lệnh 2>`.



Hình 19.3 Cấu trúc rẻ nhánh dạng đủ

– Chạy chương trình cho biết kết quả xuất ra màn hình:

Câu 1:

```
x, y = 5, 10
if x >= y:
     x+=1
else:
     y+=1
x+=2
print('x =', x, '; y =', y)
```

A. x = 6 ; y = 10 **B.** x = 7 ; y = 11

C. x = 8 ; y = 10 **D.** x = 5 ; y = 11

Câu 2:

```
x, y = 5, 10
if x > y:
     x+=1
else:
     y+=1
x+=2
print('x =', x, '; y =', y)
```

A. x = 6 ; y = 10

B. x = 7 ; y = 11

C. x = 8 ; y = 10

D. x = 5 ; y = 11

– VD: Dùng cấu trúc rẻ nhánh dạng thiếu để mô tả các mệnh đề rẽ nhánh sau:

(1) Nhập a
Nếu a chia hết cho 2
thì a là số chẵn
Ngược lại a là số lẻ

(2) if Delta <0:
print(‘ PT vô nghiệm ‘)
else:(Delta>=0)
print(‘ PT có nghiệm.

.....
.....
.....
.....
.....
.....

c) Cấu trúc if lồng nhau

– Cú pháp:

```
if <điều kiện>:
      <khối lệnh 1>
elif <điều kiện 2>:
      <khối lệnh 2>
else:
      <khối lệnh 3>
```

Ví dụ: Nhập x. Nếu x lớn hơn 0 thì thông báo x là số dương, nếu x < 0 thì thông báo x là số âm, ngược lại thông báo x=0.

```
x = float(input('Hãy nhập số x: '))
if x > 0:
    print(x, 'là số dương')
elif x < 0:
    print(x, 'là số âm')
else:
    print('x = 0')
```

B. CÂU HỎI TRẮC NGHIỆM

Câu 1: Cho a=15, b=8. Hãy tính giá trị của a, b trong trường hợp sau?

```
if a > b:
    a=b-a
    b=a+2
else:
    a=b-a
```

A. a = 15, b = 8.
B. a = -7, b = -5
C. a = 5, b = -7.
D. a = 5, b = -7.

Câu 2: Phát biểu nào sau đây có thể lấy làm biểu thức điều kiện trong cấu trúc rẽ nhánh?

A. A+B **B.** A>B **C.** A MOD 100 **D.** "A nho hon B"

Câu 3: Cho đoạn chương trình sau:

```
if (a!=0):
    x=9 // a
else:
    x=-1
print (‘ x= ’, x + 1)
```

Khi cho a=0, đoạn chương trình trên sẽ in ra màn hình giá trị x=?

A. 0 B. -1
C. 1 D. Không xác định

Câu 4: Khối lệnh trong đoạn chương trình: if <điều kiện>: < khối lệnh>

A. Luôn thực hiện
B. Thực hiện khi điều kiện đúng
C. Thực hiện khi điều kiện sai
D. Điều kiện không tính được.

Câu 5: Đoạn chương trình:

```
if <điều kiện>:
    <khối lệnh 1>
else:
    <khối lệnh 2>
```

A. Thực hiện khối lệnh 1 khi điều kiện đúng
B. Thực hiện khối lệnh 1 khi điều kiện sai
C. Thực hiện khối lệnh 2 khi điều kiện đúng
D. Thực hiện khối lệnh 1 trước rồi đến khối lệnh 2

Câu 6: Trong ngôn ngữ lập trình Python, để điễn đạt rẽ nhánh ta dùng câu lệnh if - else, sau if là <điều kiện> vậy điều kiện là?

A. phép toán logic
B. Biểu thức số học
C. Câu lệnh đơn
D. Biến kiểu bool, biểu thức so sánh, biểu thức logic

Câu 7: Đoạn chương trình:

```
Max=a
```

Hãy cho biết đoạn chương trình bên dùng để?

```
if b>Max:
    Max=b
```

A. Tính giá trị a B. Tính giá trị b
C. Tìm GTLN trong 2 số a và b D. Tìm GTNN trong 2 số a và b

Câu 8: Câu lệnh ghép được sử dụng khi?

A. Cần nhiều lệnh đơn thực hiện một công việc B. Ghép nhiều câu lệnh thành một câu lệnh
C. Tác động của câu lệnh trước đó đến nhiều câu lệnh D. Cả ba trường hợp trên

Câu 9: Cho biết giá trị của T sau khi thực hiện đoạn chương trình sau:

```
T=0
A=5
B=7
if A<B:
    T=T+A
else:
    T=T-A
print(T)
```

A. 2 B. 7
C. 5 D. 12

Câu 10: Đoạn chương trình:

```
if b>a:
    Max=b
else:
    Max=a
```

Hãy cho biết đoạn chương trình bên dùng để?

A. Tính giá trị a B. Tính giá trị b
C. Tìm giá trị lớn nhất trong 2 số a và b D. Cả ba lựa chọn trên

C. CÂU HỎI ĐÚNG/ SAI

Câu 1: Trong Python, câu lệnh rẽ nhánh if được sử dụng để kiểm tra điều kiện và thực thi các khối lệnh khác nhau tùy thuộc vào điều kiện đó, giúp kiểm soát luồng thực thi của chương trình.

a) Câu lệnh if có thể được sử dụng để kiểm tra một điều kiện và thực thi một khối lệnh nếu điều kiện đó đúng.

b) Câu lệnh if luôn yêu cầu phải có else đi kèm.

c) Có thể sử dụng if...elif...else để kiểm tra nhiều điều kiện trong cùng một khối lệnh.

d) Câu lệnh if chỉ có thể kiểm tra các điều kiện là số nguyên, không thể kiểm tra chuỗi hoặc danh sách.

Câu 2: Câu lệnh if có thể sử dụng với các kiểu dữ liệu khác nhau để kiểm tra điều kiện.

a) Trong Python, các giá trị như 0, None, "" (chuỗi rỗng) và [] (danh sách rỗng) được coi là False trong điều kiện if.

b) Python cho phép sử dụng if để kiểm tra xem một danh sách có rỗng hay không.

c) Câu lệnh if không thể kiểm tra trực tiếp kiểu dữ liệu set.

d) Khi so sánh hai chuỗi bằng if, Python phân biệt chữ hoa và chữ thường.

Câu 3: Câu lệnh rẽ nhánh if có thể kết hợp với các toán tử logic (and, or, not) để kiểm tra nhiều điều kiện đồng thời.

a) Có thể sử dụng if để kiểm tra nhiều điều kiện cùng lúc bằng cách kết hợp toán tử and, or.

b) Toán tử not có thể được dùng để đảo ngược giá trị của một điều kiện trong if.

c) Khi sử dụng if A or B:, cả hai điều kiện A và B đều phải đúng thì khối lệnh mới được thực thi.

d) Python cho phép viết if mà không có điều kiện kiểm tra.

BÀI 20: CÂU LỆNH LẶP FOR

A. TÓM TẮT NỘI DUNG

1. LỆNH RANGE

- Hàm range() sẽ sinh ra một dãy số và bạn sẽ sử dụng vòng for để duyệt qua từng số trong dãy .
- Cú pháp:

```
range (start, stop, step)
```

- Các tham số:
  - **start**: Giá trị bắt đầu của dãy số (mặc định là 0).
  - **stop**: Giá trị cuối cùng của dãy số (cận này không được lấy – nghĩa là dãy sinh ra sẽ đi từ **start đến stop -1**)
  - **step**: Bước nhảy của dãy số (mặc định là 1).

```
range (stop) trả lại vùng giá trị từ 0 đến stop - 1
range (start, stop) trả lại vùng giá trị từ start đến stop - 1
```

- Ví dụ:
  - `range(100)` là dãy số nguyên liên tục từ **0 đến 99** (dãy số tăng)
  - `range(50)` là dãy số nguyên liên tục từ **0 đến 49**
  - `range(2,10)` là dãy số nguyên liên tục từ **2 đến 9**
  - `range(5,20)` là dãy số nguyên liên tục từ …… **đến** ……
  - `range(2,10,2)` là dãy các số nguyên **2, 4, 6, 8**
  - `range(5,20,5)` là dãy các số nguyên ……
  - `range(100,0,-1)` là dãy số nguyên **100, 99, 98, … , 3, 2, 1** (dãy số giảm)
  - `range(10, 1, -2)` là dãy các số nguyên ……
  - `range(100, 1) cho vùng rỗng.`

2. LỆNH LẶP FOR

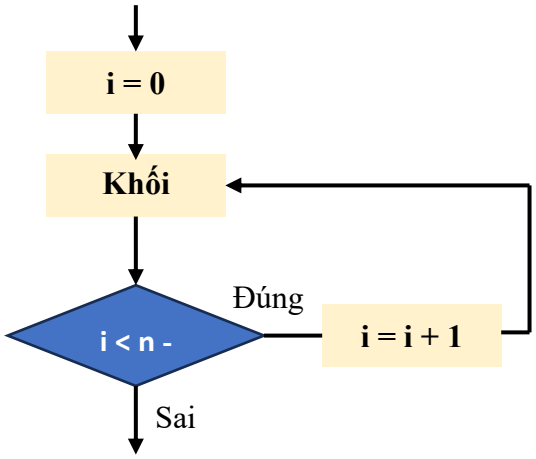
- Cú pháp:

```
for <biến lặp> in <chuỗi lặp>:
    <Khối lệnh>
```

– Trong đó:

- **<chuỗi lặp>** là chuỗi cần lặp có thể là **chuỗi, danh sách (kiểu list) hoặc dãy số.**
- **<biến lặp>** là **biến nhận giá trị của từng mục bên trong <chuỗi lặp>** trên mỗi lần lặp. Vòng lặp sẽ tiếp tục cho đến khi nó lặp tới mục cuối cùng trong <chuỗi lặp>
- **<Khối lệnh>** có thể là một câu lệnh (không cần xuống hàng mà viết sau dấu ":") hoặc nhiều câu lệnh được viết lùi vào (bằng **1 Tab** hay **4 dấu cách**) và thẳng hàng.

```
for i in range(n):
    <Khối lệnh>
```



Chương trình	Kết quả chạy chương trình
<chuỗi lặp> là **dãy số.**	

Vd 1: Xuất ra màn hình dãy số từ 0 đến 4
`for i in range(5):`
`  print(i, end=', ')`
	`0, 1, 2, 3, 4,`

Vd 2: Xuất ra màn hình 5 lần câu "Xin chào"
`for i in range(5):`
`  print('Xin chào')`
	`Xin chào`
	`Xin chào`
	`Xin chào`
	`Xin chào`
	`Xin chào`

Vd 3: Xuất ra màn hình dãy số từ 0 đến n-1. Với n nhập từ bàn phím
`n = int(input ("Nhập n: "))`
`for i in range(n):`
`  print(i, end=', ')`
	`Nhập n: 10`
	`0, 1, 2, 3, 4, 5, 6, 7, 8, 9,`

Vd 4: Tính tổng các số tự nhiên nhỏ hơn n (0+1+2+3+…+ n-1) *. Xuất tổng ra màn hình*
`n = int(input ("Nhập n: "))`
`tong = 0`
`for i in range(n):`
`  tong = tong + i`
`print('Tổng các số là', tong)`
	`Nhập n: 6`
	`Tổng các số là 15`

*Vd 5: Tính **tổng** các số tự nhiên **chẵn** nhỏ hơn n. Xuất tổng ra màn hình*	

n = int(input ("Nhập n: "))
tong = 0
for i in range(n):
    ……
        tong = tong + i
print('Tổng các số chẵn là', tong)	

*Vd 6: **Đếm** có bao nhiêu **số là bội của 3** trong dãy số tự nhiên nhỏ hơn n. Xuất đếm ra màn hình*
n = int(input ("Nhập n: "))
dem = 0
for i in range(n):
    ……
        ……
print(……)	

<chuỗi lặp> là **chuỗi**	
Vd 7: Lặp từng ký tự trong chuỗi for kytu in 'tin': print('Ký tự hiện tại:', kytu)	Ký tự hiện tại: t Ký tự hiện tại: i Ký tự hiện tại: n
<chuỗi lặp> là **danh sách (list)**	
Vd 8: Lặp theo danh sách (list) ds = ['bố','mẹ','em'] **for** i **in** ds: print('Tôi yêu', i)	Tôi yêu bố Tôi yêu mẹ Tôi yêu em
Vd 9: Tính tổng tất cả các số trong danh sách A A = [1, 7, 10] tong = 0 **for** i **in** A: tong = tong+i print("Tổng các số là", tong)	Tổng các số là: 18

B. CÂU HỎI TRẮC NGHIỆM

Câu 1: Vòng lặp nào trả về kết quả như cột bên trái dưới đây?

11111	A. for i in range(1, 6): print(i, i, i, i, i)
22222	B. for i in range(1, 5): print(str(i) * 5)
33333	C. for i in range(1, 6): print(str(i) * 5)
44444	D. for i in range(0, 5): print(str(i) * 5)
55555	

Câu 2: Điền phần còn thiếu trong đoạn code để được out như cột bên trái dưới đây:

55555	**for i in range(5, 0, ____): print(str(i) * 5)**	
44444	A. 0	B. None
33333	C. 1	D. -1

22222 11111	

Câu 4: Trong NNLT Python, câu lệnh nào sau đây đúng khi in ra màn hình các số tự nhiên từ 0 đến 10 trên 1 dòng?

A. for i in range(9): print(i,end='') B. for i in range(10): print(i,end='')
C. for i in range(11): print(i,end='') D. for i in range(12): print(i,end='')

Câu 5: Trong NNLT Python, câu lệnh nào sau đây sai khi in ra màn hình các số tự nhiên từ 0 đến 10 trên 1 dòng?

A. for i in range(0,11,1): print(i,end='') B. for i in range(11): print(i,end='')
C. for i in range(0,11): print(i,end='') D. for i in range(11,1): print(i,end='')

Câu 6: Trong NNLT Python, câu lệnh nào sau đây đúng khi in ra màn hình các số tự nhiên từ 10 đến 0 trên 1 dòng?

A. for i in range(10,0,-1): print(i,end='') B. for i in range(10,-1): print(i,end='')
C. for i in range(10,-1,-1): print(i,end='') D. for i in range(10,1,-1): print(i,end='')

Câu 7: Cho đoạn chương trình sau:

`s=0`
`for i in range(5):`
`s=s+i`
	Sau khi thực hiện, giá trị của s bằng bao nhiêu?
	A. 5 B. 10
	C. 0 D. 15

Câu 8: Đoạn chương trình `for i in range(5): print(i,end='')` cho kết quả là:

A. 1234 B. 12345 C. 01234 D. 012345

Câu 9: Đoạn chương trình sau cho kết quả là gì?

`for i in range(10):`
`  if i%2==0:`
`  print(i,end='')`
	A. 02468 B. 13579
	C. 2468 D. 246810

Câu 10: Đoạn chương trình `for i in range(0,10,2): print(i, end='')` cho kết quả gì?

A. 2468 B. 13579 C. 02468 D. 246810

C. CÂU HỎI ĐÚNG/ SAI

Câu 1: Trong Python, câu lệnh lặp for được sử dụng để lặp qua một dãy giá trị như danh sách, chuỗi, hoặc phạm vi số, giúp tự động hóa quá trình lặp lại một đoạn mã nhiều lần.

a) Câu lệnh for có thể được sử dụng để duyệt qua từng phần tử trong một danh sách, bộ, hoặc chuỗi.

b) Câu lệnh for yêu cầu bắt buộc phải có điều kiện dừng rõ ràng, giống như vòng lặp while.

c) Sử dụng for kết hợp với range() có thể giúp tạo vòng lặp chạy một số lần nhất định.

d) Câu lệnh for trong Python không thể lặp qua một từ điển (dictionary).

Câu 2: Trong Python, vòng lặp for có thể được sử dụng để duyệt qua nhiều kiểu dữ liệu khác nhau như danh sách, chuỗi, tập hợp và từ điển.

a) Vòng lặp for có thể duyệt qua từng ký tự trong một chuỗi.

b) Chỉ có thể sử dụng vòng lặp for với danh sách (list)

c) Có thể sử dụng for để duyệt qua các cặp khóa-giá trị trong từ điển (dictionary)

d) Vòng lặp for yêu cầu một biến đếm giống như trong các ngôn ngữ lập trình C hoặc Java.

Câu 3: Câu lệnh for có thể được kết hợp với range() để lặp lại một đoạn mã một số lần nhất định.

a) range(5) sẽ tạo ra một dãy gồm các số từ 0 đến 5.

b) range(2, 10, 2) tạo ra một dãy số chẵn từ 2 đến 8.

c) Có thể sử dụng range() trong vòng lặp for để lặp theo bước nhảy âm.

d) Vòng lặp for trong Python không thể lặp ngược một danh sách.

BÀI 21: CÂU LỆNH LẶP WHILE

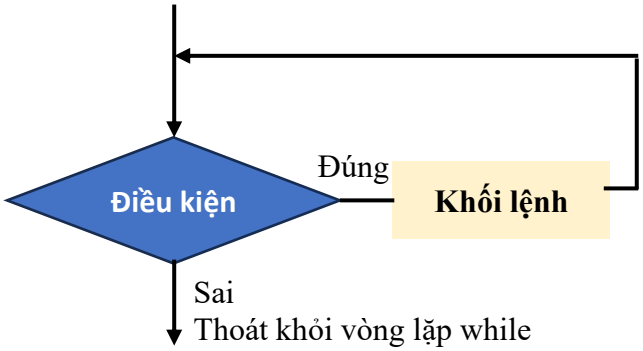
A. TÓM TẮT NỘI DUNG

1. LỆNH WHILE

– Lệnh lặp while thực hiện khối lệnh với **số lần lặp không biết trước**. Khối lệnh lặp được thực hiện cho đến khi **<điều kiện> = False** thì ngưng lặp

– Cú pháp:

```
while <điều kiện>:
    <Khối lệnh>
```



– Trong đó:

- **<điều kiện>** là biểu thức lôgic
- **<Khối lệnh>** có thể là một câu lệnh (không cần xuống hàng mà viết sau dấu ":") hoặc nhiều câu lệnh được viết lùi vào (bằng 1 Tab hay 4 dấu cách) và thẳng hàng.

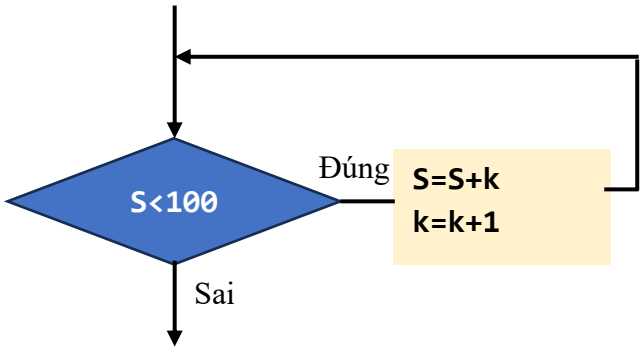
– Thực hiện lệnh: Python sẽ **kiểm tra <điều kiện>**, nếu **đúng** thì **thực hiện khối lệnh** lặp, nếu **sai** thì **kết thúc** lệnh while

– Số lần lặp của lệnh while phụ thuộc vào điều kiện của lệnh.

❖ ***Lưu ý:*** Để thoát khỏi một vòng lặp vô hạn. Khi thực thi chương trình Python ta sử dụng tổ hợp phím **Ctrl + C.**

– **Ví dụ 1:** Tính tổng các số tự nhiên đầu tiên sao cho tổng này nhỏ nhất và lớn hơn 100

```
S=0
k=0
while S<100:
    S=S+k
    k=k+1
print (S)
```



– **Ví dụ 2:** Thực hiện các lệnh sau. Kết quả sẽ in ra những số nào?

```
>>> k = 2
>>> while k < 50:
        print(k, end = " ")
        k = k + 3
```

Giải thích: Vòng lặp while sẽ dừng khi k vượt quá 50. Bắt đầu vòng lặp. k = 2. Sau mỗi bước lặp k tăng lên 3 đơn vị. Do vậy, kết quả sẽ phải in ra dãy sau:

```
2 5 8 11 14 17 20 23 26 29 32 35 38 41 44 47
```

❖ **Lưu ý:**

1. Vì lệnh while không biết trước số lần lặp, mà phụ thuộc vào điều kiện. Do đó, cần chú ý đến điều kiện của lệnh while để tránh bị lặp vô hạn.
2. Trong trường hợp nếu muốn dừng và thoát ngay khỏi vòng lặp while hoặc for có thể dùng lệnh **break**

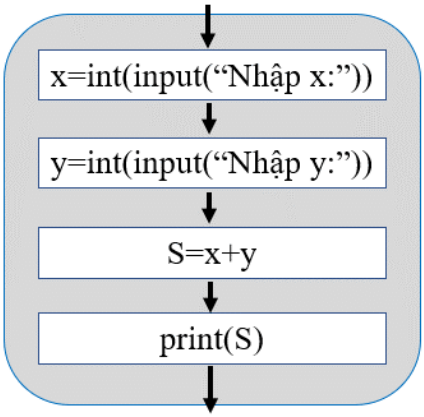
```
>>> for k in range(10):
        print(k, end = " ")
        if k == 5:  break
```

Kết quả chương trình:
0 1 2 3 4 5

2. CẤU TRÚC LẬP TRÌNH

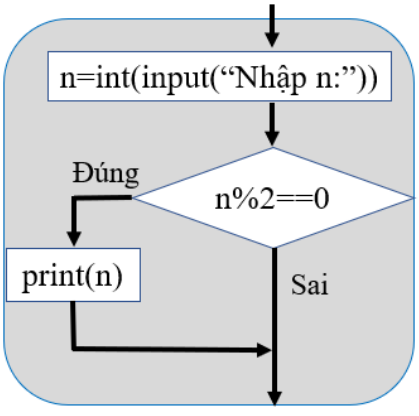
– Ba cấu trúc lập trình cơ bản của các ngôn ngữ lập trình bậc cao gồm: cấu trúc tuần tự, cấu trúc rẽ nhánh, cấu trúc lặp.

```
x=int(input("Nhập x:"))
y=int(input("Nhập y:"))
S=x+y
print(S)
```



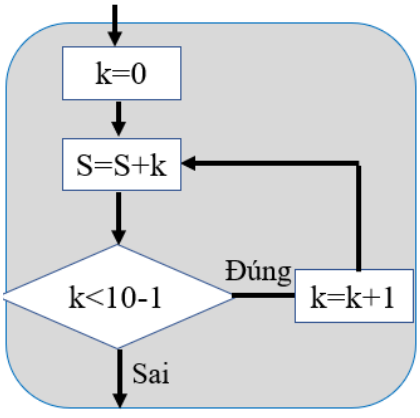
Cấu trúc tuần tự

```
n=int(input("Nhập n:"))
if n%2==0:
        print(n)
```



Cấu trúc rẽ nhánh

```
S=0
for k in range(10):
        S=S+k
```



Cấu trúc lặp

B. CÂU HỎI TRẮC NGHIỆM

Câu 1: Trong câu lệnh While thì dãy câu lệnh nào chỉ được thực hiện khi điều kiện còn?

A. True B. True hoặc False C. False D. Khác 0

Câu 2: Trong câu lệnh While, ngay sau điều kiện sẽ là dấu nào?

A. ; B. ? C. = D. :

Câu 3: Cú pháp của câu lệnh lặp với số lần chưa biết trước là?

A. while <điều kiện>: <Khối lệnh>	B. while <điều kiện> do < Khối lệnh>	C. while < Khối lệnh>: <điều kiện>	D. while < Khối lệnh> do <điều kiện>

Câu 4: Cho đoạn chương trình:

```
1  s=0
2  i=10
3  while  i >= 1:
4      s = s + 2*i
5      i = i - 2
6  print(s)
```

Kết quả của s sau khi thực hiện đoạn chương trình trên là:

A. 60 B. 15

C. 14 D. 28

Câu 5: Cho đoạn chương trình sau:

```
1  a = 12
2  b = 16
3  while a!=b:
4      if a>b:
5          a = a - b
6      else:
7          b = b - a
```

Sau khi thực hiện đoạn chương trình trên, giá trị của a và b lần lượt là:

A. 4 và 48 B. 4 và 4

C. 16 và 12 D. 12 và 16

Câu 6: Tính tổng S = 1 + 2 + 3 + … + n + … cho đến khi S>109. Điều kiện nào sau đây cho vòng lặp while là đúng:

A. While S>=109: B. While S =109: C. While S <=109: D. While S !=109:

Câu 7: Cho đoạn chương trình sau:

```
1  s=0
2  i=1
3  while i<=5:
4      s=s+1
5      i=i+1
```

Sau khi thực hiện đoạn chương trình trên giá trị của s là:

A. 9 B. 15

C. 5 D. 10

C. CÂU HỎI ĐÚNG/ SAI

Câu 1: Trong Python, câu lệnh lặp while được sử dụng để lặp lại một đoạn mã khi một điều kiện nhất định còn đúng, giúp kiểm soát luồng thực thi dựa trên điều kiện logic.

a) Vòng lặp while tiếp tục chạy khi điều kiện được xác định trong nó vẫn đúng.

b) Vòng lặp while luôn thực thi ít nhất một lần, ngay cả khi điều kiện ban đầu là False.

c) Có thể sử dụng từ khóa break để kết thúc vòng lặp while ngay lập tức.

d) Không thể sử dụng từ khóa else với vòng lặp while trong Python.

Câu 2: Trong Python, vòng lặp while có thể kết hợp với các cấu trúc dữ liệu khác để tạo ra các thuật toán linh hoạt.

a) Vòng lặp while có thể duyệt qua danh sách bằng cách sử dụng điều kiện dựa trên độ dài của danh sách.

b) Khi sử dụng while với danh sách, việc xóa phần tử khỏi danh sách trong vòng lặp có thể gây ra lỗi ngoài ý muốn.

c) Vòng lặp while không thể kết hợp với các cấu trúc dữ liệu như stack hoặc queue.

d) Sử dụng while với input() có thể giúp tạo ra các chương trình tương tác liên tục cho đến khi người dùng nhập một giá trị cụ thể.

Câu 3: Các vấn đề liên quan đến hiệu suất và tối ưu hóa khi sử dụng vòng lặp while.

a) Vòng lặp while thường chậm hơn for vì phải kiểm tra điều kiện mỗi lần lặp.

b) Sử dụng vòng lặp while thay vì for có thể làm mã nguồn khó đọc nếu không được kiểm soát tốt.

c) Vòng lặp while luôn là lựa chọn tốt nhất khi làm việc với danh sách có số lượng phần tử xác định trước.

d) Để tránh vòng lặp vô hạn, cần đảm bảo điều kiện while sẽ trở thành False tại một thời điểm nào đó.

--- Hết ---