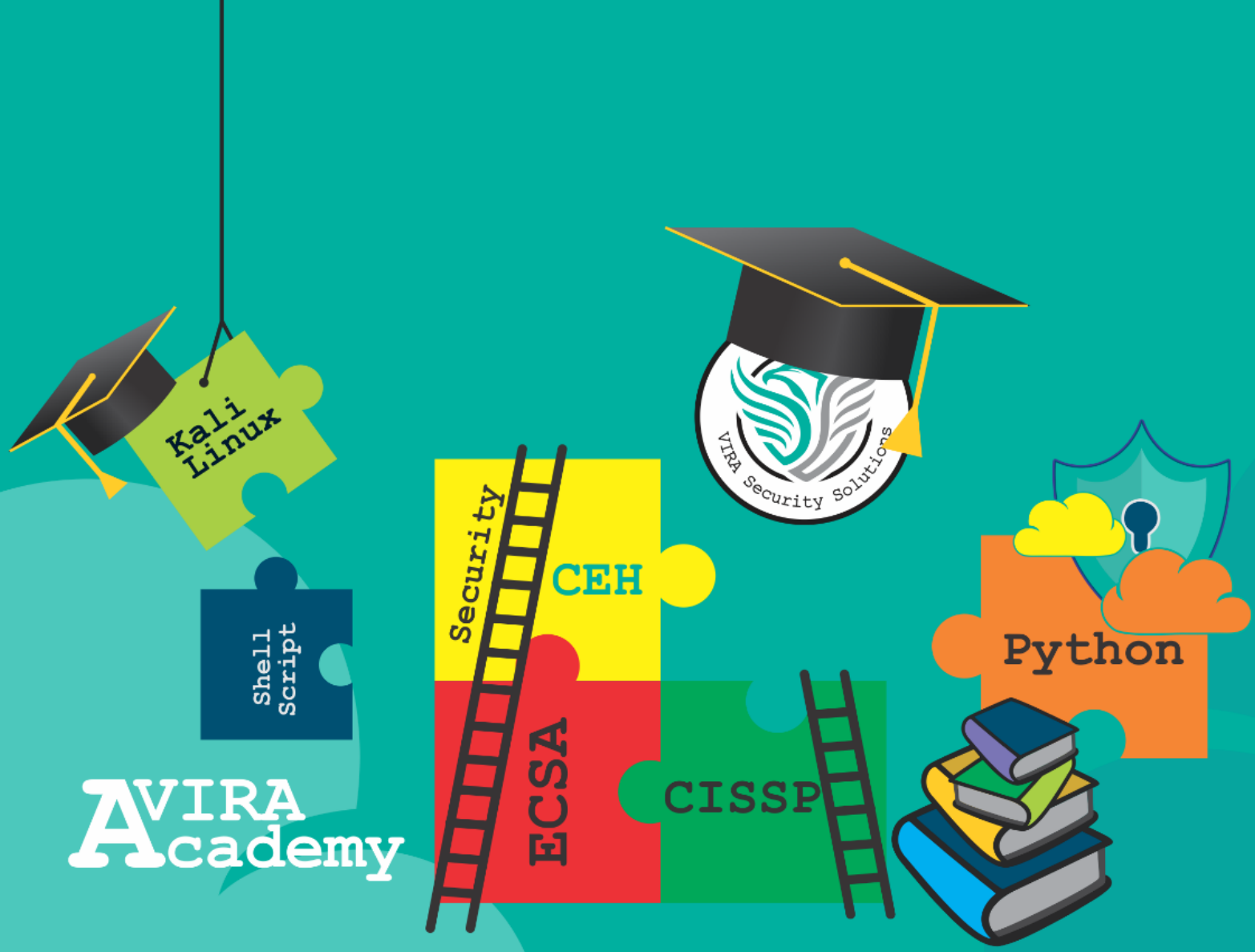

VIRA
Security
Solutions


Kali Linux
Shell Script
VIRA Academy
Security
CEH
ECSA
CISSP
VIRA Security Solutions
Python

Python

Part4

Mohammad Reza Gerami
mrgerami@aut.ac.ir
gerami@virasec.ir

Python
Conditional Logic
&
LOOPS

Python Conditions and If statements

Python supports the usual logical conditions from mathematics:

- Equals: a == b
- Not Equals: a != b
- Less than: a < b
- Less than or equal to: a <= b
- Greater than: a > b
- Greater than or equal to: a >= b

These conditions can be used in several ways, most commonly in "if statements" and loops.

An "if statement" is written by using the if keyword.

Python Conditions and If statements

If statement:

```
a = 33
b = 200
if b > a:
  print("b is greater than a")
```

Python Conditions and If statements

Elif

The `elif` keyword is pythons way of saying "if the previous conditions were not true, then try this condition".

```
a = 33
b = 33
if b > a:
  print("b is greater than a")
elif a == b:
  print("a and b are equal")
```

Python Conditions and If statements

Eles

The `else` keyword catches anything which isn't caught by the preceding conditions.

```
a = 200
b = 33
if b > a:
  print("b is greater than a")
elif a == b:
  print("a and b are equal")
else:
  print("a is greater than b")
```

Ternary Operator

Conditional Expression

Condition_if_true if condition else condition_if_false

If true do this

If false do this

```
is_friend = True
can_message = "message allowed" if is_friend else "not allowed to message"
print(can_message)
```

```
is_friend = False
can_message = "message allowed" if is_friend else "not allowed to message"
print(can_message)
```

Python Conditions and If statements

Short Hand If

If you have only one statement to execute, you can put it on the same line as the if statement.

```
if a > b: print("a is greater than b")
```

Python Conditions and If statements

Short Hand If ... Else

If you have only one statement to execute, one for if, and one for else, you can put it all on the same line:

```
a = 2
b = 330
print("A") if a > b else print("B")
```

Python Conditions and If statements

Short Hand If ... Else

One line if else statement, with 3 conditions:

```
a = 330
b = 330
print("A") if a > b else print("=") if a == b else print("B")
```

Truthy and Falsy

All values are considered "truthy" except for the following, which are "falsy":

None
False
0
0.0
0j
Decimal(0)
Fraction(0, 1)
[] - an empty list
{} - an empty dict
() - an empty tuple
'' - an empty str
b'' - an empty bytes
set() - an empty set
an empty range, like range(0)
objects for which
    obj.__bool__() returns False
    obj.__len__() returns 0

A "truthy" value will satisfy the check performed by `if` or `while` statements. We use "truthy" and "falsy" to differentiate from the `bool` values `True` and `False`.

Short Circuiting


```
is_friend = True
is_user = True
print(is_friend and is_user)
```

```
is_friend = True
is_user = True
if is_friend and is_user:
    print('best friends forever')
```

```
is_friend = True
is_user = True
if is_friend or is_user:
    print('best friends forever')
```

You can have if statements inside if statements, this is called nested if statements.

```
x = 41

if x > 10:
  print("Above ten,")
  if x > 20:
    print("and also above 20!")
  else:
    print("but not above 20.")
```

LOOPS

A for loop is used for iterating over a sequence (that is either a list, a tuple, a dictionary, a set, or a string).

This is less like the for keyword in other programming languages, and works more like an iterator method as found in other object-orientated programming languages.

With the for loop we can execute a set of statements, once for each item in a list, tuple, set etc.

```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
  print(x)
```

Looping Through a String

Even strings are iterable objects, they contain a sequence of characters:

```
for x in "banana":
  print(x)
```

The break Statement

With the break statement we can stop the loop before it has looped through all the items:

Exit the loop when x is "banana":

```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
  print(x)
  if x == "banana":
    break
```

Exit the loop when x is "banana":Exit the loop when x is "banana", but this time the break comes before the print:

```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
  if x == "banana":
    break
  print(x)
```

The range() Function

To loop through a set of code a specified number of times, we can use the range() function,
The range() function returns a sequence of numbers, starting from 0 by default, and increments by 1 (by default), and ends at a specified number.

Using the range() function:

```
for x in range(6):
  print(x)
```

Note that `range(6)` is not the values of 0 to 6, but the values 0 to 5.

range

The `range()` function defaults to 0 as a starting value, however it is possible to specify the starting value by adding a parameter: `range(2, 6)`, which means values from 2 to 6 (but not including 6):

```
for x in range(2, 6):
  print(x)
```

The `range()` function defaults to increment the sequence by 1, however it is possible to specify the increment value by adding a third parameter: `range(2, 30, 3)`:

Increment the sequence with 3 (default is 1):

```
for x in range(2, 30, 3):
  print(x)
```

Else in For Loop

The else keyword in a for loop specifies a block of code to be executed when the loop is finished:

Print all numbers from 0 to 5, and print a message when the loop has ended:

```
for x in range(6):
  print(x)
else:
  print("Finally finished!")
```

Nested Loops

A nested loop is a loop inside a loop.

The "inner loop" will be executed one time for each iteration of the "outer loop":

Print each adjective for every fruit:

```
adj = ["red", "big", "tasty"]
fruits = ["apple", "banana", "cherry"]

for x in adj:
  for y in fruits:
    print(x, y)
```

```
red apple
red banana
red cherry
big apple
big banana
big cherry
tasty apple
tasty banana
tasty cherry
```

Nested Loops

A nested loop is a loop inside a loop.

The "inner loop" will be executed one time for each iteration of the "outer loop":

Print each adjective for every fruit:

```
for x in  ["red", "big", "tasty"] :
  for y in  ["apple", "banana", "cherry"] :
    print(x, y)
```

```
red apple
red banana
red cherry
big apple
big banana
big cherry
tasty apple
tasty banana
tasty cherry
```

The pass Statement

for loops cannot be empty, but if you for some reason have a for loop with no content, put in the pass statement to avoid getting an error.

```
for x in [0, 1, 2]:
  pass
```

Exercise:

Complete the syntax and run

Loop through the items in the fruits list.

```
fruits = ["apple", "banana", "cherry"]
 …… x …..   fruits …..
            print(x)
```

```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
  if x == "banana":
      …..
  print(x)
```

Exercise:

Use the range function to loop through a code set 6 times.

```
for x in …….. :
  print(x)
```

Exit the loop when x is "banana".

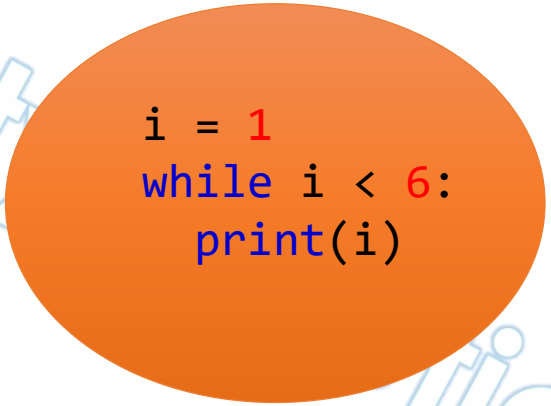
```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
  if x == "banana":
     ……..
  print(x)
```

While Loops

With the while loop we can execute a set of statements as long as a condition is true.

Print i as long as i is less than 6:

```
i = 1
while i < 6:
  print(i)
  i += 1
```



Note: remember to increment i, or else the loop will continue forever.

While Loops

The while loop requires relevant variables to be ready, in this example we need to define an indexing variable, i, which we set to 1.

The break Statement

With the `break` statement we can stop the loop even if the while condition is true:

Exit the loop when i is 3:

```
i = 1
while i < 6:
  print(i)
  if i == 3:
    break
  i += 1
```

The continue Statement

With the continue statement we can stop the current iteration, and continue with the next:

Continue to the next iteration if i is 3:

```
i = 0
while i < 6:
  i += 1
  if i == 3:
    continue
  print(i)
```

The else Statement

With the else statement we can run a block of code once when the condition no longer is true:

Print a message once the condition is false:

```
i = 1
while i < 6:
  print(i)
  i += 1
else:
  print("i is no longer less than 6")
```

While Loops


```
my_list = [1,2,3]
for item in my_list:
    print(item)

i = 0
while i < len(my_list):
   print(my_list[i])
   i +=1
```

While Loops


```
my_list = [1,2,3]
for item in my_list:
    print(item)
    break

i = 0
while i < len(my_list):
    print(my_list[i])
    i +=1
    break
```

While Loops


```
my_list = [1,2,3]
for item in my_list:
    print(item)
    break

i = 0
while i < len(my_list):
    print(my_list[i])
    i +=1
    break
```

While Loops

Infinite loop:

```
while True:
    print(my_list[i])
    break
```

```
while True:
    input('say something:')
    break
```

```
while True:
    input('say something:')
```

PASS


```
my_list = [1,2,3]
for item in my_list:
    print(item)
    pass

i = 0
while i < len(my_list):
    print(my_list[i])
    i +=1
    pass
```

```
my_list = [1,2,3]
for item in my_list:
    #thinking

i = 0
while i < len(my_list):
    print(my_list[i])
    i +=1
    pass
```

Iterable

```
car = {
    'Brand' : 'BMW',
    'Model' : 'X5',
    'Year':2020
    }
for item in car:
    print(item)

for item in car.items():
    print(item)

for item in car.values():
    print(item)

for item in car.keys():
    print(item)
```

Iterable


```
car = {
    'Brand' : 'BMW',
    'Model' : 'X5',
    'Year':2020
    }

for key,value in car.items():
    print(key,value)

for item in 50:
    print(item)
```

Note: iterable – List, dictionary, tuple, set, string
One by one check each item in the collection

Counter


```
my_list = [1,2,3,4,5,6,7,8,9,10]

counter = 0
for item in my_list:
    counter = counter + item
    print(counter)
```

```
1
3
6
10
15
21
28
36
45
55
```

Counter


```
my_list = [1,2,3,4,5,6,7,8,9,10]

counter = 0
for item in my_list:
    counter = counter + item
print(counter)
```

```
55
```

Range


```
for number in range(0,5):
    print(number)
```

```
for _ in range(0,5):
    print(_)

for _ in range(0,10,2):
    print(_)
```

```
for _ in range(10,0):
    print(_)

for _ in range(10,0,-1):
    print(_)

for _ in range(10,0,-2):
    print(_)
```

Range

```
for _ in range(10,0,-2):
    print(list(range(10)))
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

```
for _ in range(2):
    print(list(range(10)))
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

Enumerate

```
for char in enumerate('Helloooo'):
    print(char)
```

Index	Char
(0,	'H')
(1,	'e')
(2,	'l')
(3,	'l')
(4,	'o')
(5,	'o')
(6,	'o')
(7,	'o')

```
for i,char in enumerate('Helloooo'):
    print(i,char)
```

Index	Char
0	H
1	e
2	l
3	l
4	o
5	o
6	o
7	o

Enumerate

```
for char in enumerate([1,2,3]):
    print(char)
```

```
(0, 1)
(1, 2)
(2, 3)
```

```
for char in enumerate((1,2,3)):
    print(char)
```

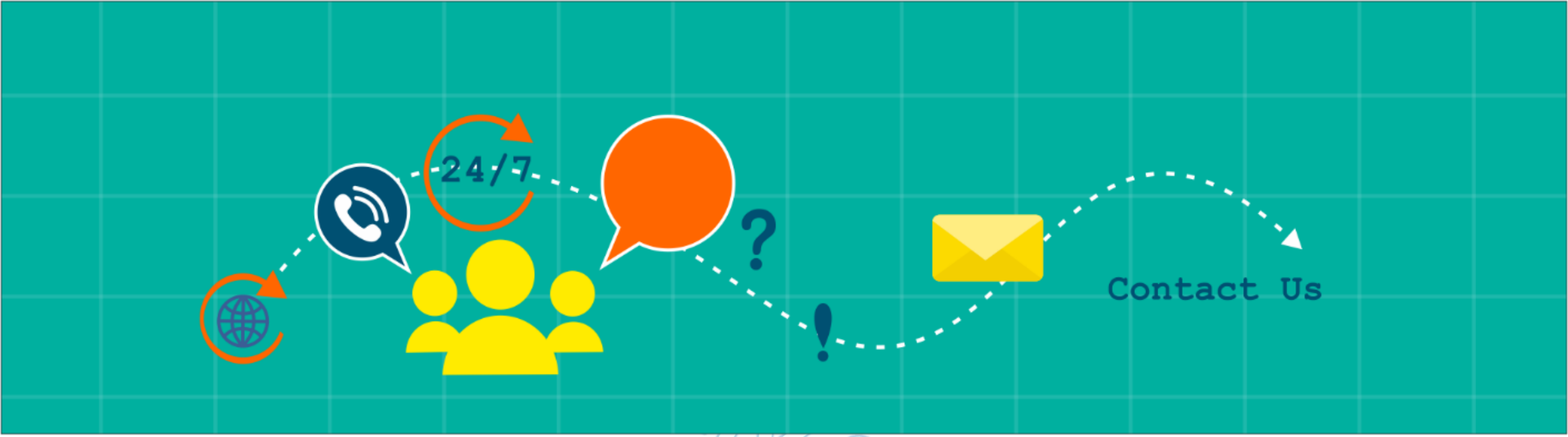
```
(0, 0)
(1, 1)
(2, 2)
(3, 3)
(4, 4)
(5, 5)
(6, 6)
(7, 7)
(8, 8)
(9, 9)
```

```
for char in enumerate(list(range(10))):
    print(char)
```

Enumerate

```
for i,char in enumerate(list(range(20))):
    print(i,char)
    if char == 15:
        print(f'index of 15 is : {i}')
```

```
0 0
1 1
2 2
3 3
4 4
5 5
6 6
7 7
8 8
9 9
10 10
11 11
12 12
13 13
14 14
15 15
index of 15 is : 15
16 16
17 17
18 18
19 19
```



Visiting Address: No 53 Vafa Manesh Ave

Heravi, Pasdaran Ave, TEHRAN-IRAN

Tel No: 0098 21 22196115-09125792641

Email: info@ virasec.ir

Website: www.virasec.ir

آدرس: تهران، پاسداران، هروی، خیابان وفامنش، پلاک ۵۳

شماره تماس: ۰۹۱۲۵۷۹۲۶۴۱-۰۲۱۲۲۱۹۶۱۱۵