

CSIT110

Fundamental Programming with Python

Loop Statements (1)

Goh X. Y.



In this lecture

- For loop
- More on string data type
- `print(..., end=...)`
- The break keyword
- More about str

How does it look like?

```
for i in <iterator>:  
    # statements using i  
    print(i)
```

Definition

To iterate: *verb*

- To perform or utter repeatedly

Example of an iterator – range()

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

start – optional. stop – required. step - optional

Returns a sequence of numbers,
starting from 0 by default,
increments by 1 by default,
stops **BEFORE** a specified number.

```
range(4)          -> 0,1,2,3  
range(7, 10)      -> 7,8,9  
range(3, 8, 2)    -> 3,5,7
```

Taken from www.w3school.com

The first for-loop example

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

Program output:



range(0,10)

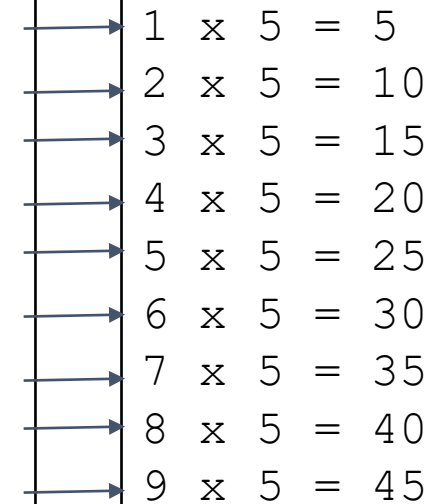
number 10 is excluded!!!

Times table example

```
for i in range(1,10):  
    print(f"{i} x 5 = {5*i}")
```

Program output:

```
i = 1, print(f"{i} x 5 = {5*i}")  
i = 2, print(f"{i} x 5 = {5*i}")  
i = 3, print(f"{i} x 5 = {5*i}")  
i = 4, print(f"{i} x 5 = {5*i}")  
i = 5, print(f"{i} x 5 = {5*i}")  
i = 6, print(f"{i} x 5 = {5*i}")  
i = 7, print(f"{i} x 5 = {5*i}")  
i = 8, print(f"{i} x 5 = {5*i}")  
i = 9, print(f"{i} x 5 = {5*i}")
```



1	x	5	=	5
2	x	5	=	10
3	x	5	=	15
4	x	5	=	20
5	x	5	=	25
6	x	5	=	30
7	x	5	=	35
8	x	5	=	40
9	x	5	=	45

Times table example 2

```
for i in range(1,10):  
    print(f"{i} x 5 = {5*i}")
```

We want to print times table based on user input

```
number_input = input("Enter a number: ")  
number = int(number_input)  
for i in range(1,10):  
    print(f"{i} x {number} = {number*i}")
```

```
Enter a number: 6  
1 x 6 = 6  
2 x 6 = 12  
3 x 6 = 18  
4 x 6 = 24  
5 x 6 = 30  
6 x 6 = 36  
7 x 6 = 42  
8 x 6 = 48  
9 x 6 = 54
```

Friend of 10 table

```
0 + 10 = 10
1 + 9 = 10
2 + 8 = 10
3 + 7 = 10
4 + 6 = 10
5 + 5 = 10
6 + 4 = 10
7 + 3 = 10
8 + 2 = 10
9 + 1 = 10
10 + 0 = 10
```


Friend of 10 table

i = 0	→	0 + 10 = 10
i = 1	→	1 + 9 = 10
i = 2	→	2 + 8 = 10
i = 3	→	3 + 7 = 10
i = 4	→	4 + 6 = 10
i = 5	→	5 + 5 = 10
i = 6	→	6 + 4 = 10
i = 7	→	7 + 3 = 10
i = 8	→	8 + 2 = 10
i = 9	→	9 + 1 = 10
i = 10	→	10 + 0 = 10

```
for i in range(0,11):
```

Friend of 10 table

What is this `second` number?

```
print(f"{i} + {second} = {10}")
```

i = 0	→	0 + 10 = 10
i = 1	→	1 + 9 = 10
i = 2	→	2 + 8 = 10
i = 3	→	3 + 7 = 10
i = 4	→	4 + 6 = 10
i = 5	→	5 + 5 = 10
i = 6	→	6 + 4 = 10
i = 7	→	7 + 3 = 10
i = 8	→	8 + 2 = 10
i = 9	→	9 + 1 = 10
i = 10	→	10 + 0 = 10

```
for i in range(0,11):  
    print(f"{i} + {second} = {10}")
```

Friend of 10 table

What is this `second` number?

```
print(f"{i} + {second} = {10}")
```

`second = 10 - i`

i = 0	→	0 + 10 = 10
i = 1	→	1 + 9 = 10
i = 2	→	2 + 8 = 10
i = 3	→	3 + 7 = 10
i = 4	→	4 + 6 = 10
i = 5	→	5 + 5 = 10
i = 6	→	6 + 4 = 10
i = 7	→	7 + 3 = 10
i = 8	→	8 + 2 = 10
i = 9	→	9 + 1 = 10
i = 10	→	10 + 0 = 10

```
for i in range(0,11):  
    second = 10 - i  
    print(f"{i} + {second} = {10}")
```

Friend of 10 table

What is this `second` number?

```
print(f"{i} + {second} = {10}")
```

`second = 10 - i`

i = 0	→	0 + 10 = 10
i = 1	→	1 + 9 = 10
i = 2	→	2 + 8 = 10
i = 3	→	3 + 7 = 10
i = 4	→	4 + 6 = 10
i = 5	→	5 + 5 = 10
i = 6	→	6 + 4 = 10
i = 7	→	7 + 3 = 10
i = 8	→	8 + 2 = 10
i = 9	→	9 + 1 = 10
i = 10	→	10 + 0 = 10

or
simply

```
for i in range(0,11):  
    print(f"{i} + {10 - i} = {10}")
```

Friend of 10 table

```
print(f"{i:>2} + {second:>2} = 10")
```

i = 0	→	0 + 10 = 10
i = 1	→	1 + 9 = 10
i = 2	→	2 + 8 = 10
i = 3	→	3 + 7 = 10
i = 4	→	4 + 6 = 10
i = 5	→	5 + 5 = 10
i = 6	→	6 + 4 = 10
i = 7	→	7 + 3 = 10
i = 8	→	8 + 2 = 10
i = 9	→	9 + 1 = 10
i = 10	→	10 + 0 = 10

Better
display

```
for i in range(0,11):  
    print(f"{i:>2} + {10 - i:>2} = 10")
```

print(...,end=...)

print(...,end=...)

Default:
end="\n"

```
for i in range(0,11):  
    # print the number  
    print(i, end="")  
    # print trailing word  
    trailing = "frog"  
    print(trailing, end="")
```

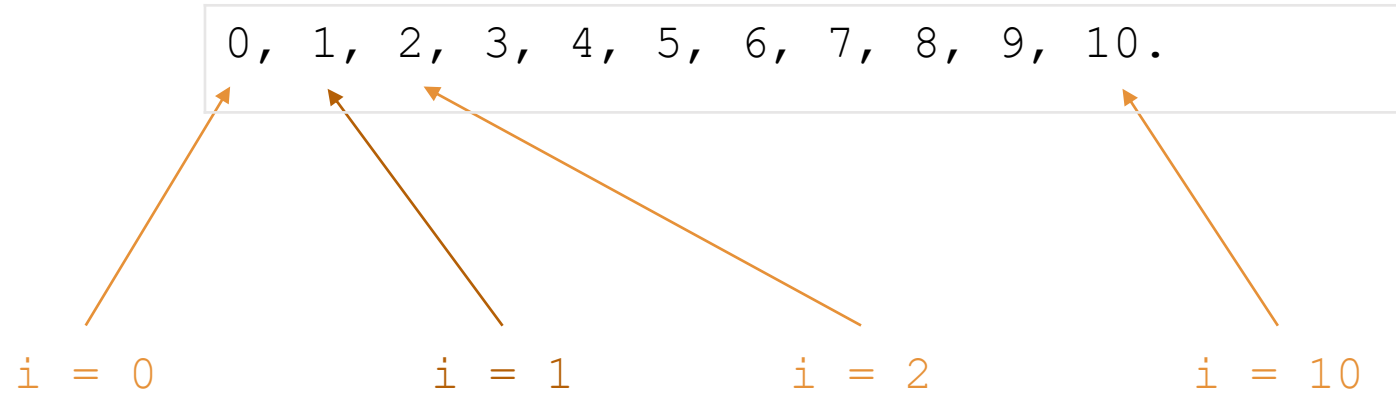
0frog1frog2frog3frog4frog5frog6frog7frog8frog9frog10frog

Consecutive Numbers

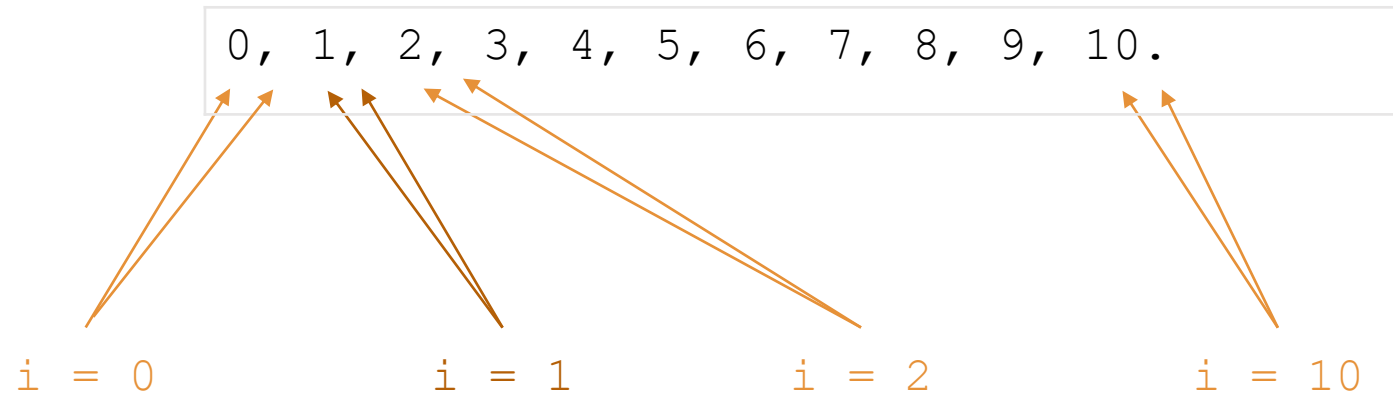
We want to write a program to print the following output

```
0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10.
```


Consecutive Numbers



Consecutive Numbers

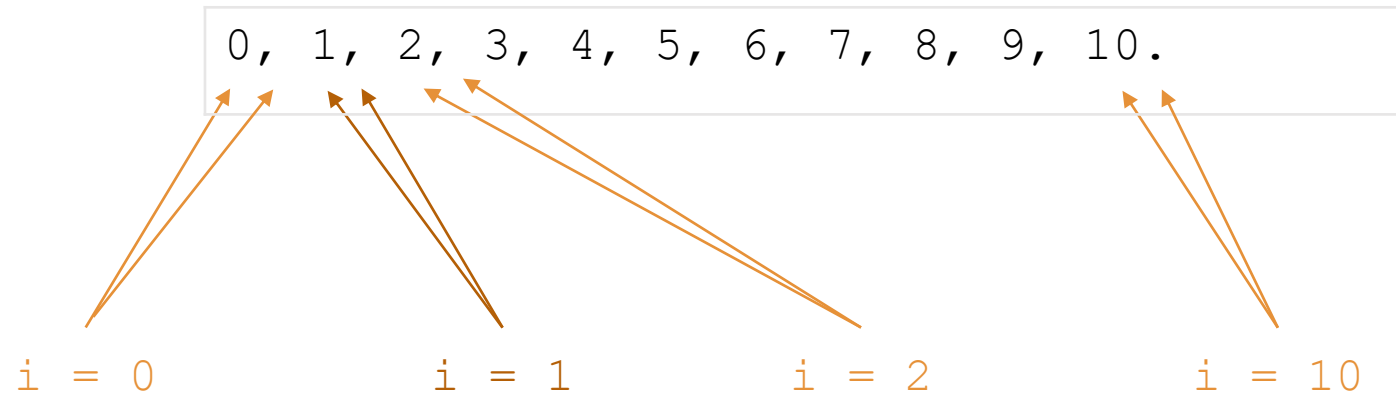


```
for i in range(0,11):  
    # print the number  
    # print the trailing
```

The **trailing** depends on the index i :

- $i = 0, 1, \dots, 9$: the trailing is a comma
- $i = 10$: the trailing is a full-stop

Consecutive Numbers



```
for i in range(0,11):  
    if i < 10:  
        trailing = ", "  
    else:  
        trailing = "."  
    print(i, end= trailing) # prints the number  
# while replacing the newline with the trailing variable
```

Sum of Numbers

$$1 + 2 + 3 + 4 + \dots + 10 = ?$$

Adding one number at a time:

`result = 0`

`i = 1 → result = result + 1`

`i = 2 → result = result + 2`

`i = 3 → result = result + 3`

`i = 4 → result = result + 4`

`i = 5 → result = result + 5`

...

`i = 10 → result = result + 10 = ?`

$$0 + \mathbf{1} = 1$$

$$1 + \mathbf{2} = 3$$

$$3 + \mathbf{3} = 6$$

$$6 + \mathbf{4} = 10$$

$$10 + \mathbf{5} = 15$$

$$45 + \mathbf{10} = 55$$

Sum of Numbers

$$1 + 2 + 3 + 4 + \dots + 10 = ?$$

```
# initialise the result to zero
result = 0

# keep adding the result with number from 1 to 10
for i in range(1,11):
    result = result + i
    # display the result
print(f"The sum of 1 to 10 is {result}")
```

The sum of 1 to 10 is 55

Sum of Numbers – one step at a time

$$1 + 2 + 3 + 4 + \dots + 10 = ?$$

```
result = 0
for i in range(1,11):
    result = result + i
    print(result)
```

1
3
6
10
15
21
28
36
45
55

Adding one number of a time:

result = 0

i = 1 → result = 0 + **1** = 1

i = 2 → result = 1 + **2** = 3

i = 3 → result = 3 + **3** = 6

i = 4 → result = 6 + **4** = 10

i = 5 → result = 10 + **5** = 15

...

i = 10 → result = result + **10** = ?

Number Pattern

```
2 1
4 3 2 1
6 5 4 3 2 1
8 7 6 5 4 3 2 1
10 9 8 7 6 5 4 3 2 1
```

```
i = 1 → 2 1
i = 2 → 4 3 2 1
i = 3 → 6 5 4 3 2 1
i = 4 → 8 7 6 5 4 3 2 1
i = 5 → 10 9 8 7 6 5 4 3 2 1
```

What is the pattern?

for each i from 1 to 5

```
start_number = 2 * i
```

print from the **start_number** down to **1**
that is:

```
start_number - 0
start_number - 1
start_number - 2
start_number - 3
...
```



```
# display 5 lines of pattern
for i in range(1, 6):
    # display the ith line
    # the first number on line i is 2*i
    start_number = 2 * i
    # print from start number down to 1
    for j in range(0, start_number):
        number = start_number - j
        print(number, end=" ") # no newline
    # print a new line to complete the line I
    print()
```

```
2 1
4 3 2 1
6 5 4 3 2 1
8 7 6 5 4 3 2 1
10 9 8 7 6 5 4 3 2 1
```

The **break** keyword

The **break** keyword

The **break** statement terminates the closest enclosing loop.

```
# a flag to indicate user has answered YES
user_say_yes = False

# patiently ask the user 10 times until they say YES
for i in range(0, 10):
    answer = input("Would you like green eggs and ham? (Y/N): ")
    if answer == "Y":
        user_say_yes = True
        print("That's a smart choice!")
        break  # ← use break to stop the loop

# if the user has not said yes
if (user_say_yes == False):
    print("Oh well, you don't know what you're missing!")
```

The **break** keyword

```
Would you like green eggs and ham? (Y/N) : N  
Would you like green eggs and ham? (Y/N) : N  
Would you like green eggs and ham? (Y/N) : N  
Would you like green eggs and ham? (Y/N) : N  
Would you like green eggs and ham? (Y/N) : N  
Would you like green eggs and ham? (Y/N) : N  
Would you like green eggs and ham? (Y/N) : N  
Would you like green eggs and ham? (Y/N) : N  
Would you like green eggs and ham? (Y/N) : N  
Would you like green eggs and ham? (Y/N) : N  
Oh well, you don't know what you're missing!
```

```
Would you like green eggs and ham? (Y/N) : N  
Would you like green eggs and ham? (Y/N) : N  
Would you like green eggs and ham? (Y/N) : Y  
That's a smart choice!
```

Str

Can be used as an iterator:

```
text = "hi there!"  
for i in text:  
    print(i)
```

h
i

t
h
e
r
e
!

String data type

Find the length of a string:

```
greeting = "Hi there!"  
greeting_length = len(greeting) → 9
```

Get one character at a time:

```
print(greeting[0]) → H  
print(greeting[1]) → i  
print(greeting[2]) → space  
print(greeting[3]) → t  
print(greeting[4]) → h  
print(greeting[5]) → e  
print(greeting[6]) → r  
print(greeting[7]) → e  
print(greeting[8]) → !
```

Question. What is the last index?

Answer. `len(greeting) - 1`

Display characters of string

```
greeting = "Hi there!"  
for i in range(0, len(greeting)):  
  
    # get the ith character  
    letter = greeting[i]  
  
    # display the ith character  
    print(letter)
```

Question. What is the last index?

Answer. `len(greeting) - 1`

Output:

```
H  
i  
  
t  
h  
e  
r  
e  
!
```

Function + for + if-else

Example: Generate password

Example: Generate password

In an online game, the initial password is generated from the username by replacing each letter i to 1, r to 7, s to 5, and z to 2.

Write a program to generate this initial password.

```
Enter username: Superman123  
Password is 5upe7man123
```

```
Enter username: zebra8  
Password is 2eb7a8
```

```
Initially set password = ""  
Username letter  Password letter  
    z                2          password = "2"  
    e                e          password = "2e"  
    b                b          password = "2eb"  
    r                7          password = "2eb7"  
    a                a          password = "2eb7a"  
    8                8          password = "2eb7a8"
```

Example: generate password

```
# ask user to enter usernameh
username = input("Enter username: ")

# construct the password
```

Initially set password = ""		
Username letter	Password letter	
z	2	password = "2"
e	e	password = "2e"
b	b	password = "2eb"
r	7	password = "2eb7"
a	a	password = "2eb7a"
8	8	password = "2eb7a8"

```
# display password result
print("Password is " + password)
```

Example: Generate password

```
# construct the password for username
def generate_password(username):
    password = "frog"
    return password
```

```
# ask user to enter username
username = input("Enter username: ")

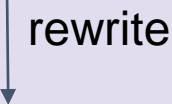
# construct the password
password = generate_password(username)

# display password result
print("Password is " + password)
```

```
Enter username: zebra8
Password is frog
```

Example: Generate password

```
# construct the password for username
def generate_password(username):
    password = "frog"
    return password
```



```
def generate_password(username):
    # initialize password as empty string
    password = ""

    for i in range(0, len(username)):
        # get the ith character from username
        username_letter = username[i]

        # construct corresponding character for password
        password_letter = transform_character(username_letter)

        # adding a character to password
        password = password + password_letter

    return password
```

CSIT110 - Fundamental Programming with Python

Example: Generate password

```
# construct password letter from username letter
```

```
def transform_character(letter):  
    password_letter = "p"  
    return password_letter
```

Enter username: zebra8
Password is pppppp

```
def generate_password(username):  
    # initialize password as empty string  
    password = ""  
  
    for i in range(0, len(username)):  
        # get the ith character from username  
        username_letter = username[i]  
  
        # construct corresponding character for password  
        password_letter = transform_character(username_letter)  
  
        # adding a character to password  
        password = password + password_letter  
  
    return password
```

Example: Generate password

```
# construct password letter from username letter
def transform_character(letter):
    password_letter = "p"
    return password_letter
```

```
def transform_character(letter):
    if (letter == "i") or (letter == "I"):
        password_letter = "1"
    elif (letter == "r") or (letter == "R"):
        password_letter = "7"
    elif (letter == "s") or (letter == "S"):
        password_letter = "5"
    elif (letter == "z") or (letter == "Z"):
        password_letter = "2"
    else:
        password_letter = letter

    return password_letter
```

rewrite



Enter username: zebra8
Password is 2eb7a8

Example: Generate password

```
# initialize password as empty string
password = ""
for i in range(0, len(username)):
    # get the ith character from username
    letter = username[i]
    # construct corresponding character for password
    if (letter == "i") or (letter == "I"):
        password_letter = "1"
    elif (letter == "r") or (letter == "R"):
        password_letter = "7"
    elif (letter == "s") or (letter == "S"):
        password_letter = "5"
    elif (letter == "z") or (letter == "Z"):
        password_letter = "2"
    else:
        password_letter = letter

    # adding a character to password
    password = password + password_letter
```

Example: Generate password

Or use string as an iterator

```
# initialize password as empty string
password = ""
for letter in username:
    # get the ith character from username
    letter = username[i]
    # construct corresponding character for password
    if (letter == "i") or (letter == "I"):
        password_letter = "1"
    elif (letter == "r") or (letter == "R"):
        password_letter = "7"
    elif (letter == "s") or (letter == "S"):
        password_letter = "5"
    elif (letter == "z") or (letter == "Z"):
        password_letter = "2"
    else:
        password_letter = letter

    # adding a character to password
    password = password + password_letter
```


Any questions?