CSIT881
Programming and Data Structures

For-Loop Statements

UNIVERSITY
OF WOLLONGONG
AUSTRALIA

Objectives

- For loop
- More on string data type

The first for-loop example

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

Program output:

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

```
0
1
2
3
4
5
6
7
8
9
```

`range(0,10)` → number 10 is excluded!!!

Times table example

```
for i in range(1,10):
  print("{0} x {1} = {2}".format(i, 5, 5*i))
```

Program output:

```
i = 1, print("{0} x {1} = {2}".format(i, 5, 5*i))
i = 2, print("{0} x {1} = {2}".format(i, 5, 5*i))
i = 3, print("{0} x {1} = {2}".format(i, 5, 5*i))
i = 4, print("{0} x {1} = {2}".format(i, 5, 5*i))
i = 5, print("{0} x {1} = {2}".format(i, 5, 5*i))
i = 6, print("{0} x {1} = {2}".format(i, 5, 5*i))
i = 7, print("{0} x {1} = {2}".format(i, 5, 5*i))
i = 8, print("{0} x {1} = {2}".format(i, 5, 5*i))
i = 9, print("{0} x {1} = {2}".format(i, 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("{0} x {1} = {2}".format(i, 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("{0} x {1} = {2}".format(i, 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("{0} + {1} = {2}".format(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("{0} + {1} = {2}".format(i, second, 10))
```

Friend of 10 table

second = 10 - i

```
print("{0} + {1} = {2}".format(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):
  second = 10 - i
  print("{0} + {1} = {2}".format(i, second, 10))
```

Friend of 10 table

second = 10 - i

```
print("{0} + {1} = {2}".format(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

or simply

```
for i in range(0,11):
  print("{0} + {1} = {2}".format(i, 10 - i, 10))
```

Friend of 10 table

```
print("{0:>2} + {1:>2} = {2:>2}".format(i, 10 - i, 10))
```

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

```
 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
```

better display

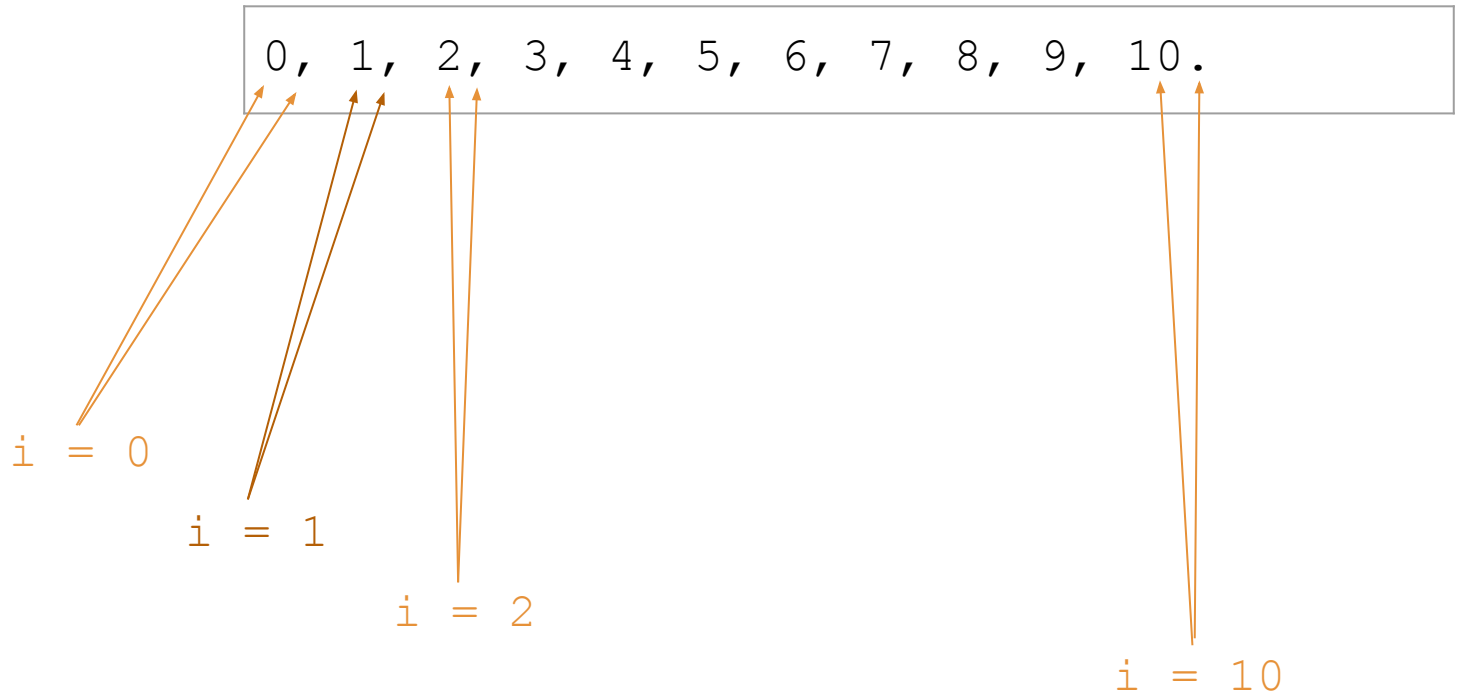
```
for i in range(0,11):
  print("{0:>2} + {1:>2} = {2:>2}".format(i, 10 - i, 10))
```

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



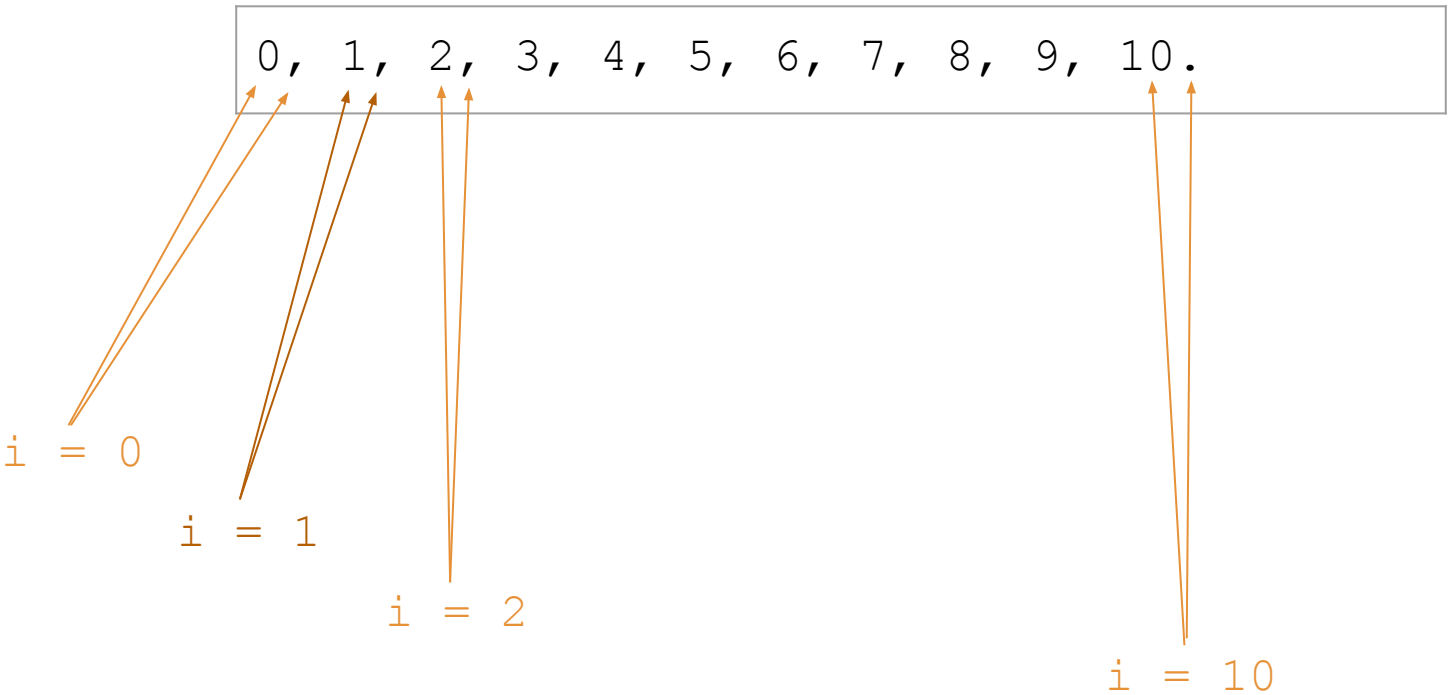

```
for i in range(0,11):
  # print the number
  print(i, end="")

  # print the trailing
  trailing = "frog"
  print(trailing, end="")
```

Output:

0frog1frog2frog3frog4frog5frog6frog7frog8frog9frog10frog

Consecutive numbers

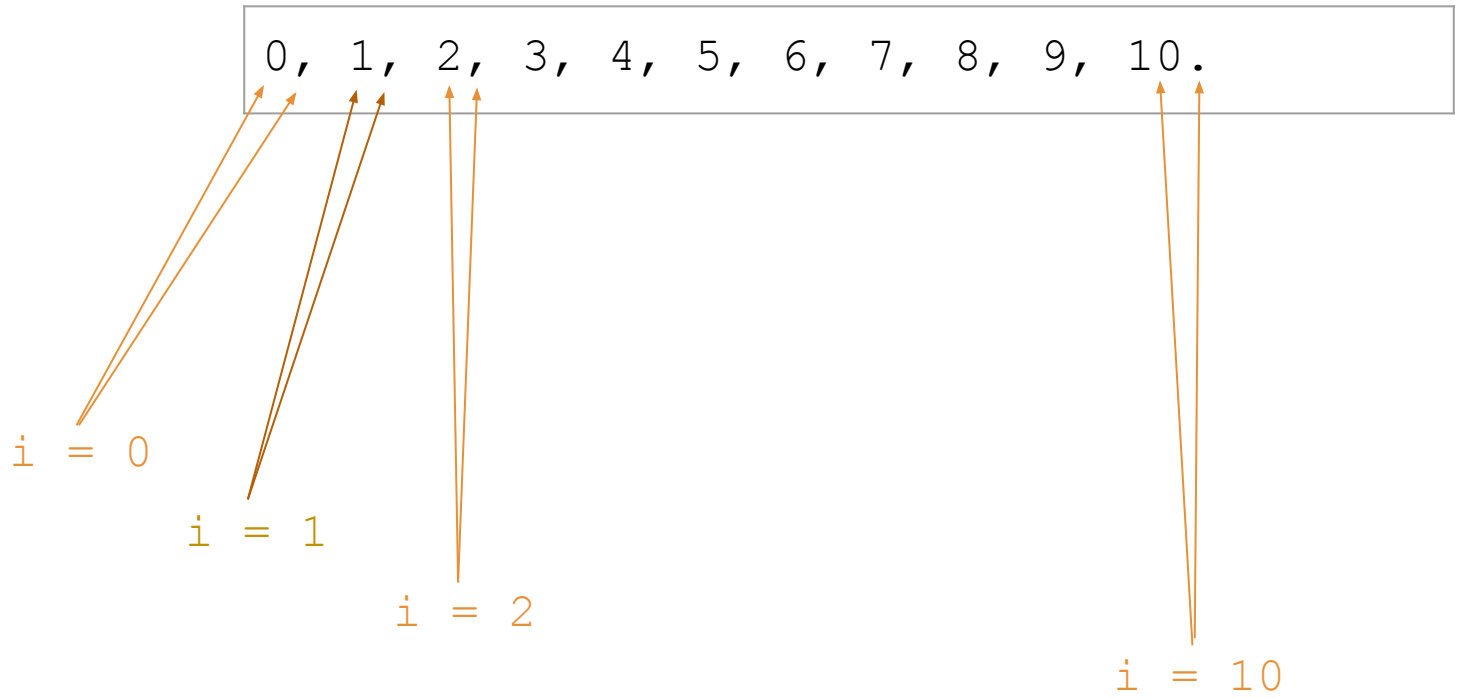



```
for i in range(0,11):
  // print the number
  // print the trailing
```

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

- `i = 0, 1,..., 9`: the trailing is the comma
- `i = 10`: the trailing is the full-stop

Consecutive numbers




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

  if (i < 10):
    trailing = ", "
  else:
    trailing = "."

  print(i, end="")          <- print the number
  print(trailing, end="")   <- print the trailing
```

Sum of numbers

```
1 + 2 + 3 + 4 + … + 10 = ?
```

Adding one number of a time:

result = 0

i = 1 → result = result + **1**	0 + **1** = 1
i = 2 → result = result + **2**	1 + **2** = 3
i = 3 → result = result + **3**	3 + **3** = 6
i = 4 → result = result + **4**	6 + **4** = 10
i = 5 → result = result + **5**	10 + **5** = 15
…	
i = 10 → result = result + **10** = ?	45 + **10** = 55

Sum of numbers

```
1 + 2 + 3 + 4 + … + 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("The sum of 1 to 10 is {0}".format(result))
```

Sum of numbers

```
1 + 2 + 3 + 4 + … + 10 = ?
```

```
result = 0

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

print(result)
```

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
          ...
```

```python
# display 5 lines of pattern
for i in range(1, 6):
  # display the ith line

  # the first number on line i is 2i
  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=" ")

  # 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** 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

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

use **break** *to stop the loop*

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!
```

String data type

Upper case:

```
name = "John Smith"
name_uppercase = name.upper()
print(name_uppercase)
```

→ JOHN SMITH

Lower case:

```
name = "John Smith"
name_lowercase = name.lower()
print(name_lowercase)
```

→ john smith

String data type

Searching for a substring:

```
name = "Alexandra"

index = name.find("exa")
print(index)

index = name.find("frog")
print(index)

index = name.find("Alex")
print(index)
```

2

-1

0

`find` returns the first index if found, otherwise, it return -1 if not found

Index 0 means the first character.

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`

String data type

Slicing a string:

```
sentence = "Python is cool!"

sub_sentence1 = sentence[1:4]
# "yth"
```

[i:j] gives substring from index i up to index (j-1), so altogether, there are (j-i) characters

```
sub_sentence2 = sentence[1:]
# "ython is cool!"
```

[i:] gives substring from index i up to the end

```
sub_sentence3 = sentence[:4]
# "Pyth"
```

[:j] is the same as [0:j] gives substring from index 0 up to index (j-1), so altogether, there are j characters

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
!
```

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 username
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

```
# 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
```