Loops

Outline

- Updating variables
- While loop
  - Updating variables
  - Break & continue
  - Infinite Loops
- For loop
  - Iterable objects – string, list, range, (dictionary, tuple, set)
  - Updating variables
  - Break & continue
  - Applications – adding up, counter, maximum number
  - Nested loops

Loop statements

- Computers are often used to automate repetitive tasks. Repeated execution of a set of statements is called iteration.

- A loop statement repeats an action over and over.
  - `While loop`
  - `For loop`

Updating variables

- A common pattern in loops statements is an assignment statement that updates a variable, where the new value of the variable depends on the old.

```
x = 10
x = x + 1
print (x)     x = 10 +1 = 11
x = x + 1
print (x)     x = 11 +1 = 12
x = x + 1
print (x)     x = 12 +1 = 13
```

```
11
12
13
```

```
x = 10
x = x - 1
print (x)     x = 10 - 1 = 9
x = x - 1
print (x)     x = 9 - 1 = 8
x = x - 1
print (x)     x = 8 - 1 = 7
```

```
9
8
7
```

Updating variables

- Assignment operators are used to assign values to variables.

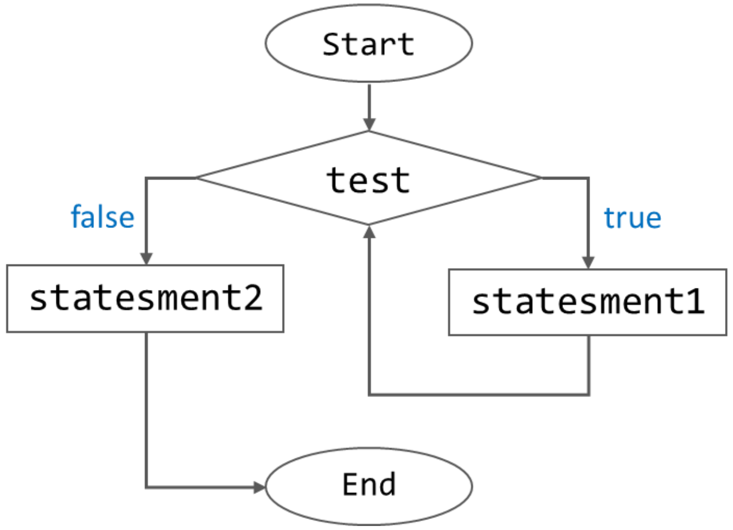
Assignment operator	Example	Same as
=	x = 1	x = 1
+=	x += 1	x= x+1
-=	x -= 1	x= x-1
*=	x *= 2	x= x*2
/=	x /= 2	x= x/2

While loop

- A while statement repeatedly executes a block of statements as long as a test at the top keeps evaluating to a true value.
- It is called a “loop” because control keeps looping back to the start of the statement until the test becomes false.

General format

```
While <test>:              # loop test
      <statements1>  # loop body
else:                      # optional else
      <statements2>  # run if the test is false
```



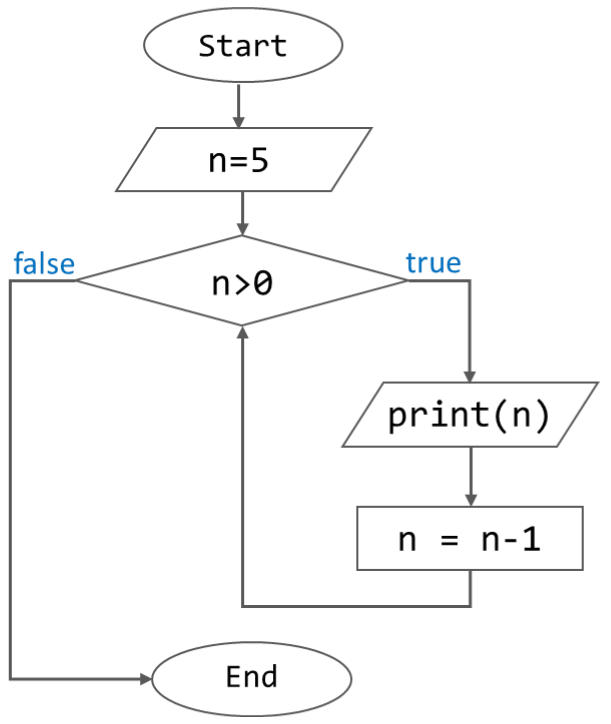
While loop

- While n is greater than 0, display the value of n and then reduce the value of n by 1.

```
n = 5
while n > 0:        # loop test
    print(n)        # loop body
    n = n - 1
```

```
5
4
3
2
1
```

Iteration	n	n>0	n - 1
1	5	True	5 - 1 = 4
2	4	True	4 - 1 = 3
3	3	True	3 - 1 = 2
4	2	True	2 - 1 = 1
5	1	True	1 - 1 = 0
6	0	False	

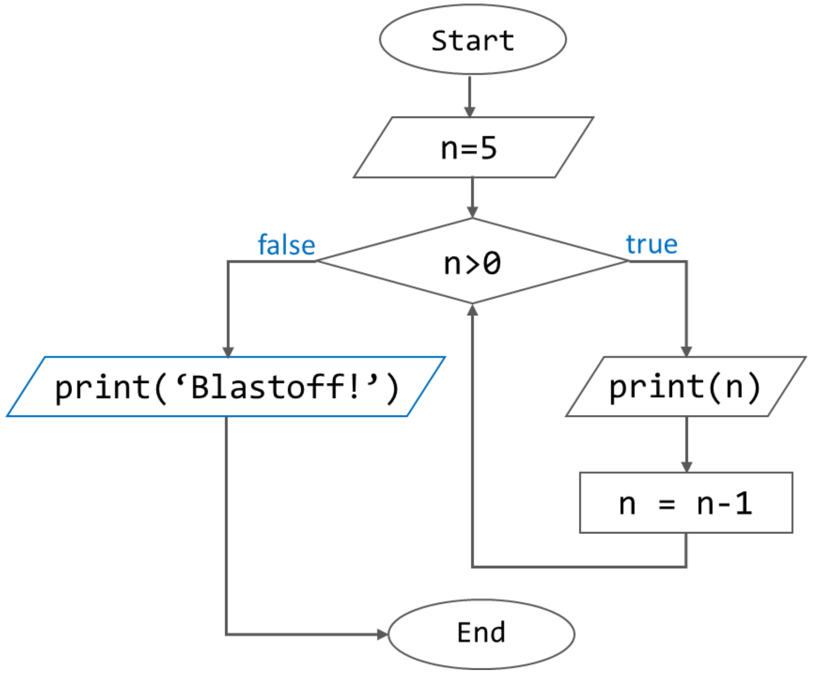


While loop - else

- The statements inside the `else` clause are executed only when the test is false.

```
n = 5
while n > 0:
    print(n)
    n = n - 1
else:
    print("Blastoff!")  # run if the test is false
```

```
5
4
3
2
1
Blastoff!
```



Exercise

(A.1) Define a variable `n=1`. Write a program to print out the following results.

```
1
2
3
4
5
```

(A.2) Define a variable `n=2`. Write a program to print out the following results.

```
2
4
6
8
10
done
```

While loop – break and continue

- `Break` and `continue` statements can appear anywhere inside the `while` (or `for`) loop's body, but they are usually coded further nested in an `if` test to take action in response to some condition.

General format

```
While <test1>:
        <statements1>
        if <test2>:
                break       # Exit  loop now, skip else
        if <test3>:
                continue    # Go to top of loop now, to test1
        <statements2>
else:
        <statements3>       # run if we didn't hit a 'break'
```

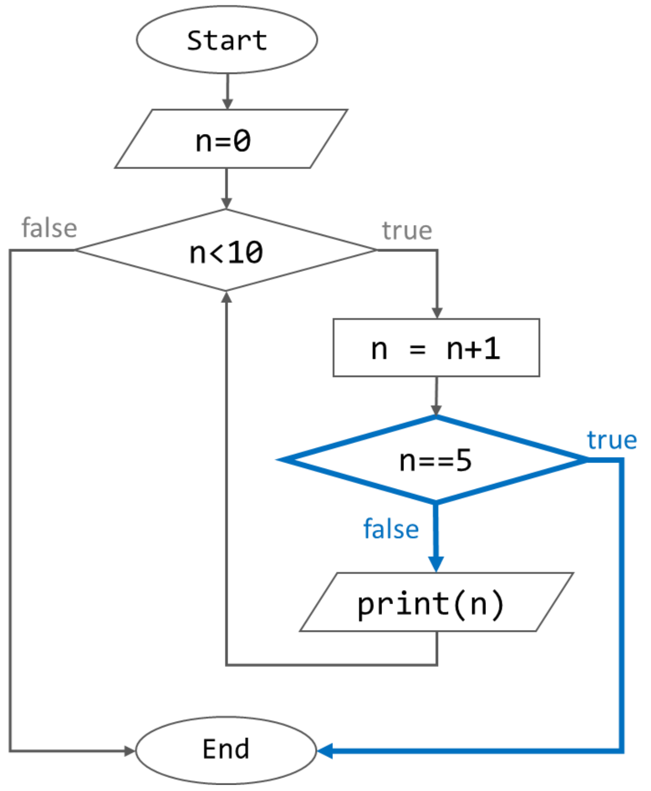
While loop - break

- The break statement causes an immediate exit from a loop.

```
n = 0
while n<10:
    n = n + 1
    if n==5:
        break
    print(n)
```

```
1
2
3
4
```

Iteration	n	n<10	n + 1	n == 5
1	0	True	0 + 1 = 1	False
2	1	True	1 + 1 = 2	False
3	2	True	2 + 1 = 3	False
4	3	True	3 + 1 = 4	False
5	4	True	4 + 1 = 5	True

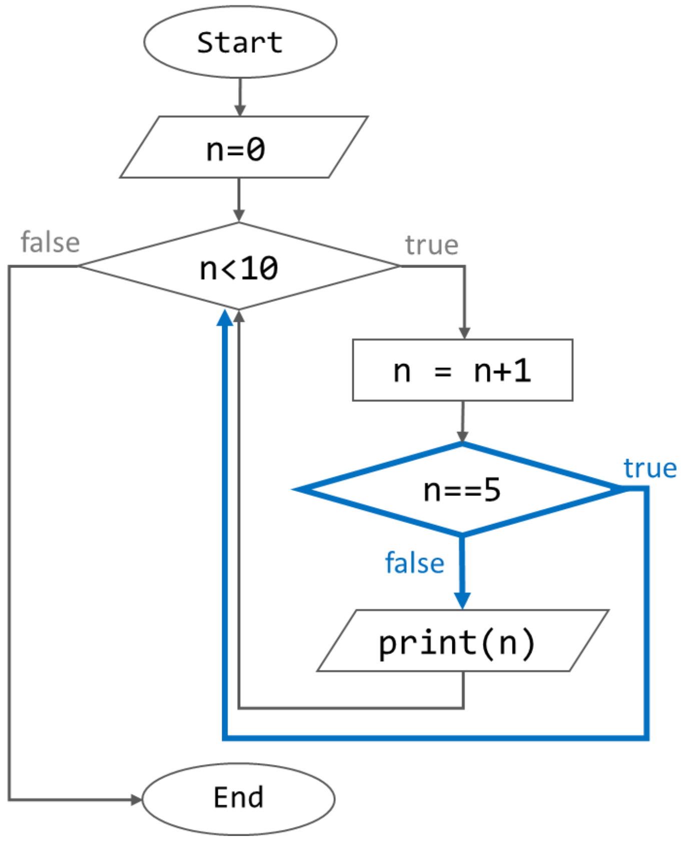


While loop - continue

- The continue statement causes an immediate jump to the top of a loop.

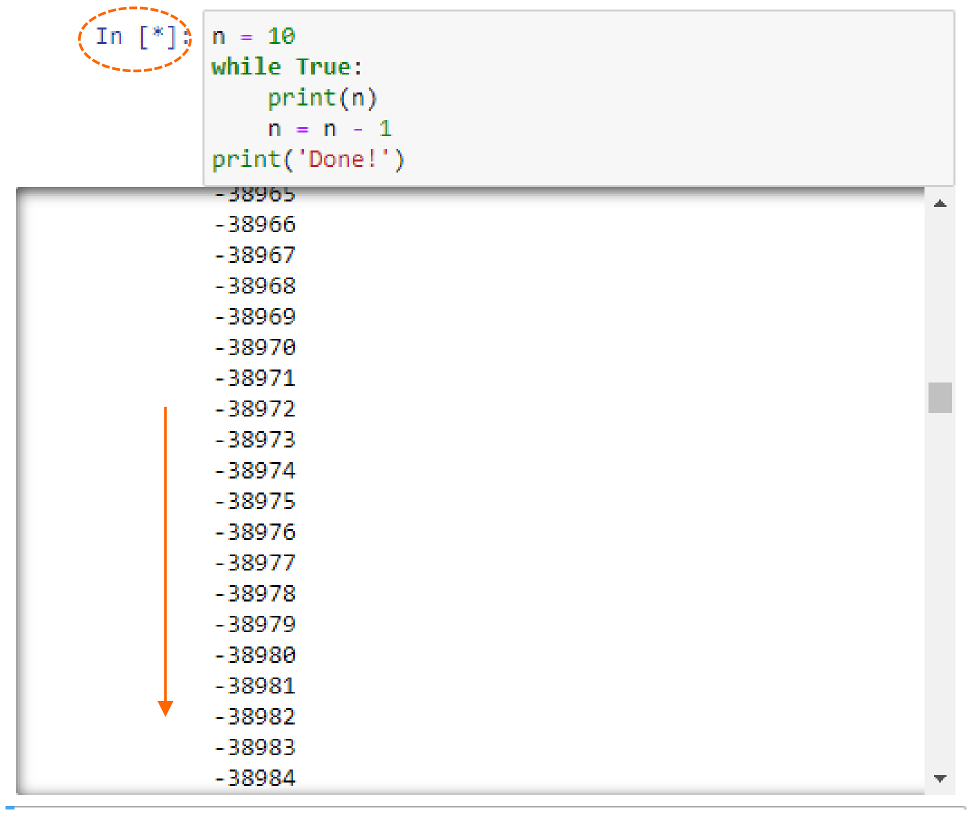
```
n = 0
while n<10:
    n = n + 1
    if n==5:
        continue
    print(n)
```

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



Infinite loops

- What happens if the loop always is true?



```
In [*]:
n = 10
while True:
    print(n)
    n = n - 1
print('Done!')
```

```
-38965
-38966
-38967
-38968
-38969
-38970
-38971
-38972
-38973
-38974
-38975
-38976
-38977
-38978
-38979
-38980
-38981
-38982
-38983
-38984
```

- Terminate the kernel.
- Rewrite you code (e.g., add a break statement).

Exercise

(B.1) Define an input box named `x` . Write a program to print numbers from 0 to 20. But stop the process if the number is equal to `x` .

(B.2) Define an input box named `y` .Write a program to print numbers from 0 and 20, except for y.

For loop

WHILE loops & FOR loops

- Using `while` statement to construct an indefinite loop
  - **`While`** statement loops until some condition becomes False.
- Using `for` statement to construct a definite loop
  - **`for`** statements loops through a known set of items so it runs through as many iterations as there are items in the set.

- While statement and for statement are similar in structure:
  - There is a condition to be evaluated, and then there is a loop body.

```
While <test>:                  # loop test
      <statements>             # loop body
```

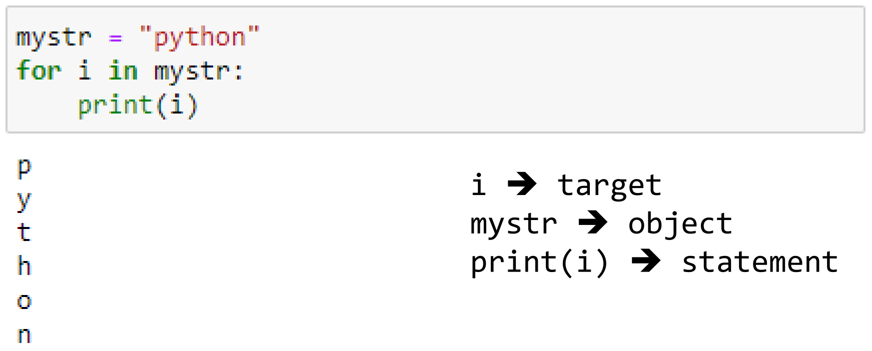
```
for <target> in <object>:  # Assign object items to target
      <statements>          # Repeat loop body use target
```

For loop

- A `for` loop can iterate through an iterable object.
  - string, list, range, dictionary, tuple, set

General format

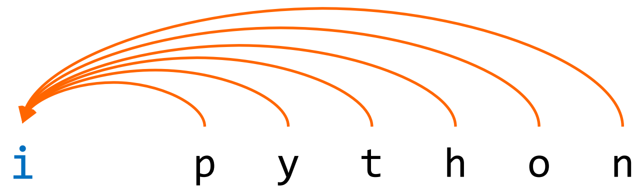
```
for <target> in <object>:      # Assign object items to target
        <statements>           # Repeat loop body use target
```



```
mystr = "python"
for i in mystr:
    print(i)
```

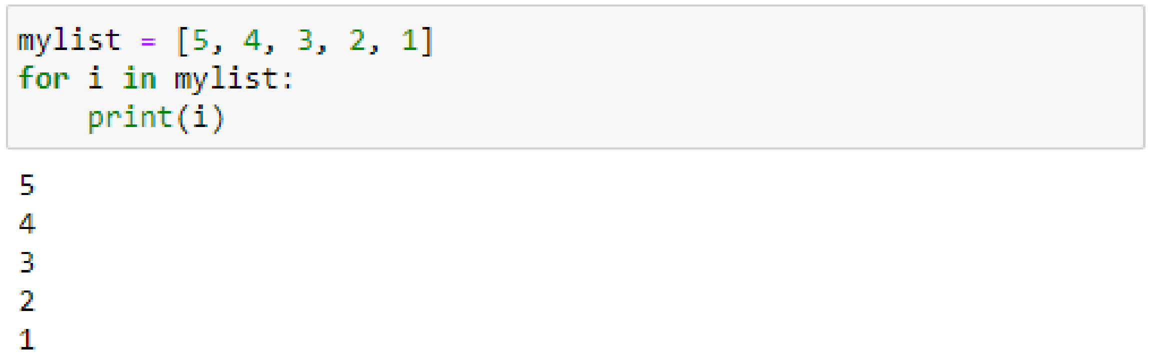
```
p
y
t
h
o
n
```

```
i ➔ target
mystr ➔ object
print(i) ➔ statement
```



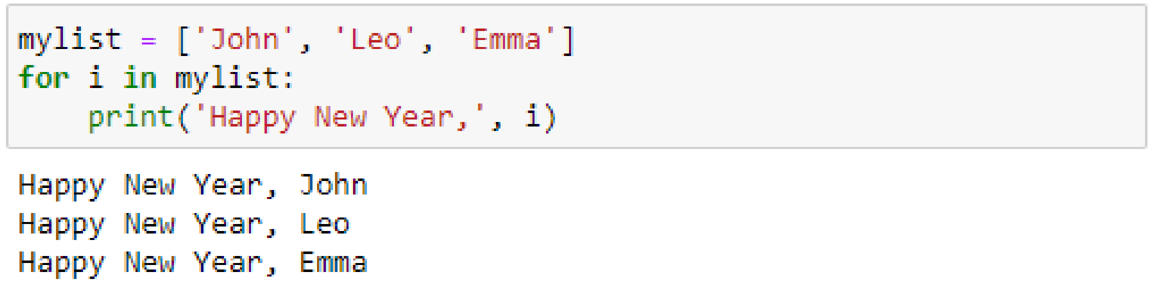
For loop – list

- Assign the target to each of the items in a list in turn.



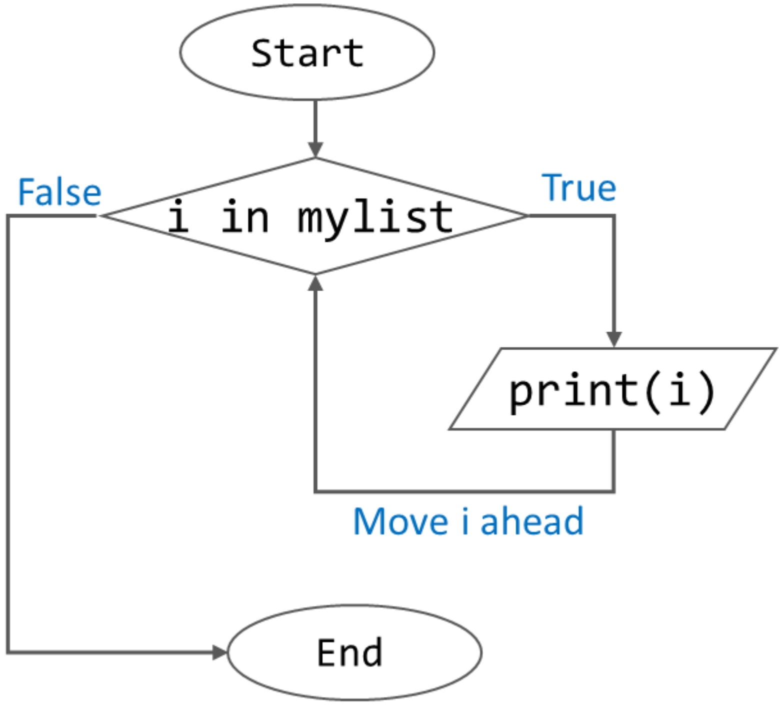
```python
mylist = [5, 4, 3, 2, 1]
for i in mylist:
    print(i)
```

```
5
4
3
2
1
```



```python
mylist = ['John', 'Leo', 'Emma']
for i in mylist:
    print('Happy New Year,', i)
```

```
Happy New Year, John
Happy New Year, Leo
Happy New Year, Emma
```



For loop - range

- The range() function is used to generate a sequence of numbers, starting from 0 by default, and increments by 1 (by default), and stops before a specified number.

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

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

```
for i in range(1,5):
    print(i)
```

Not including 5

```
1
2
3
4
```

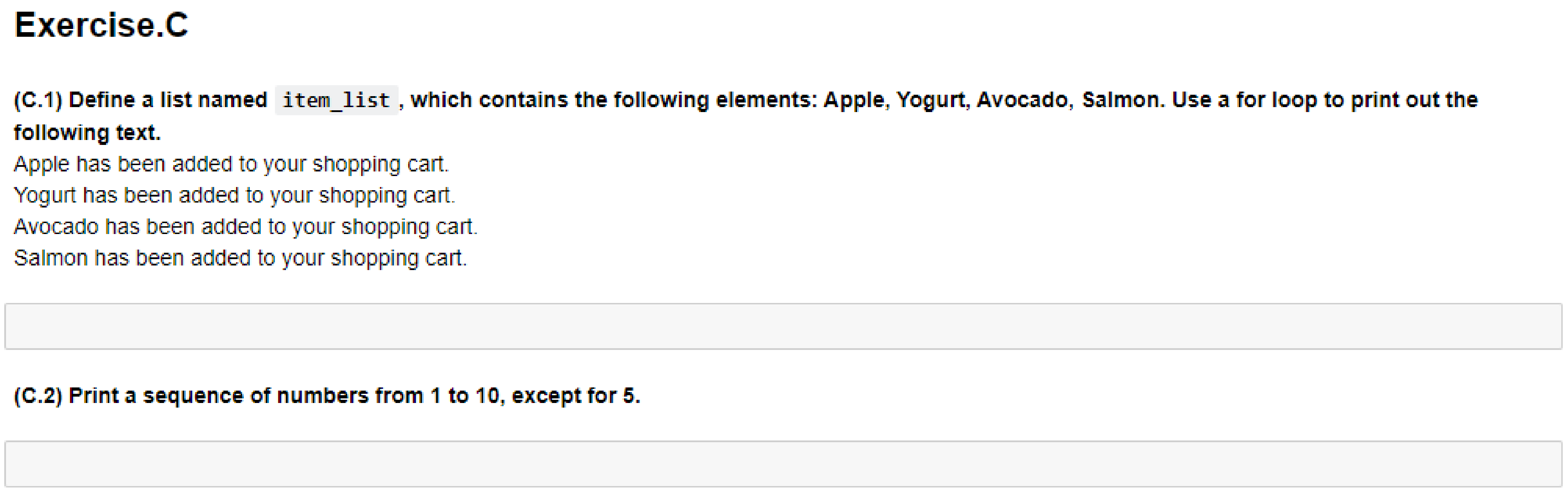
```
for i in range(5):
    print("hello world")
```

```
hello world
hello world
hello world
hello world
hello world
```

BI

AACSB ACCREDITED EFMD EQUIS ACCREDITED ASSOCIATION OF MBAS AMBA ACCREDITED

Exercise



Exercise.C

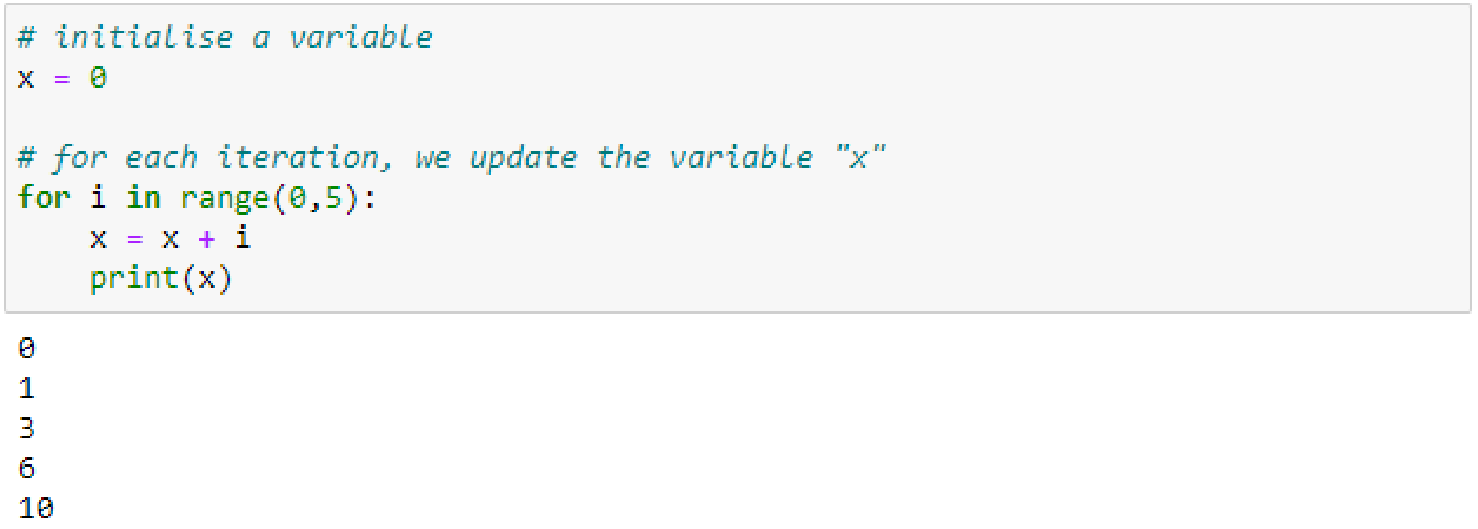
(C.1) Define a list named `item_list` , which contains the following elements: Apple, Yogurt, Avocado, Salmon. Use a for loop to print out the following text.

Apple has been added to your shopping cart.
Yogurt has been added to your shopping cart.
Avocado has been added to your shopping cart.
Salmon has been added to your shopping cart.

(C.2) Print a sequence of numbers from 1 to 10, except for 5.

For loop – updating variables

- A common pattern in loops statements is to update a variable, where the new value of the variable depends on the old.



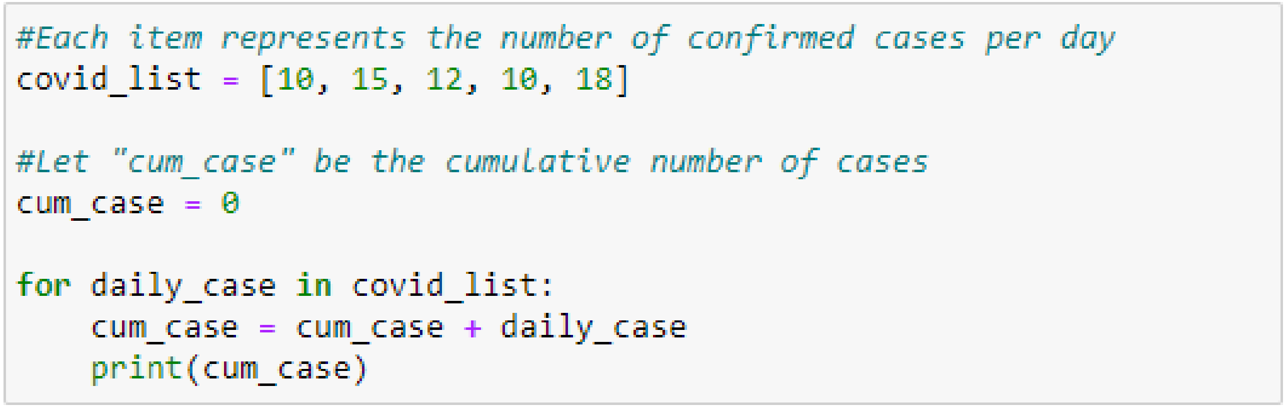
```
# initialise a variable
x = 0

# for each iteration, we update the variable "x"
for i in range(0,5):
    x = x + i
    print(x)
```

```
0
1
3
6
10
```

Iteration	i	x
1	0	0 + 0 = 0
2	1	0 + 1 = 1
3	2	1 + 2 = 3
4	3	3 + 3 = 6
5	4	6 + 4 = 10

For loop – updating variables

- Example: Calculate the cumulative number of cases per day



```python
#Each item represents the number of confirmed cases per day
covid_list = [10, 15, 12, 10, 18]

#Let "cum_case" be the cumulative number of cases
cum_case = 0

for daily_case in covid_list:
    cum_case = cum_case + daily_case
    print(cum_case)
```

10	day1
25	Day1 + Day2
37	Day1 + Day2 + Day3
47	Day1 + Day2 + Day3 + Day4
65	Day1 + Day2 + Day3 + Day4 + Day5

Iteration	daily_case	cum_case
1	10	0 + 10 = 10
2	15	10 + 15 = 25
3	12	25 + 12 = 37
4	10	37 + 10 = 47
5	18	47 + 18 = 65

Exercise

(D.1) Define a list named `mylist`, which contains the following elements: 1,3,5,7,9. Use a for loop to print out the cumulative number.

Hint: Defined a variable `total_number = 0`, update `total_number` in each iteration. The first line is 1; the second line is 1+3 = 4; the third line is 4+5 =9, and so on.

Expected result:

```
1
4
9
16
25
```

(D.2) Create a list with the following elements: A, B, C, D, E. Use a for loop to print out the following result.

Hint: Assign `""` to a variable to initialize an empty string.

Expected result:

```
A
AB
ABC
ABCD
ABCDE
```

For loop – break & continue

- Use `break` and `continue` in a for loop’s body.

```python
for i in range(0,10):
    if i == 5:
        break
    print(i)
```

```
0
1
2
3
4
```

```python
for i in range(0,10):
    if i == 5:
        continue
    print(i)
```

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

For loop – adding up

- Updating a variable in a for loop’s body.

```
# initialise a variable
total_number = 0

# for each iteration, we update the variable "total_number"
for number in range(1,11):
    total_number = total_number + number
    print(total_number)
```

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

For loop - counter

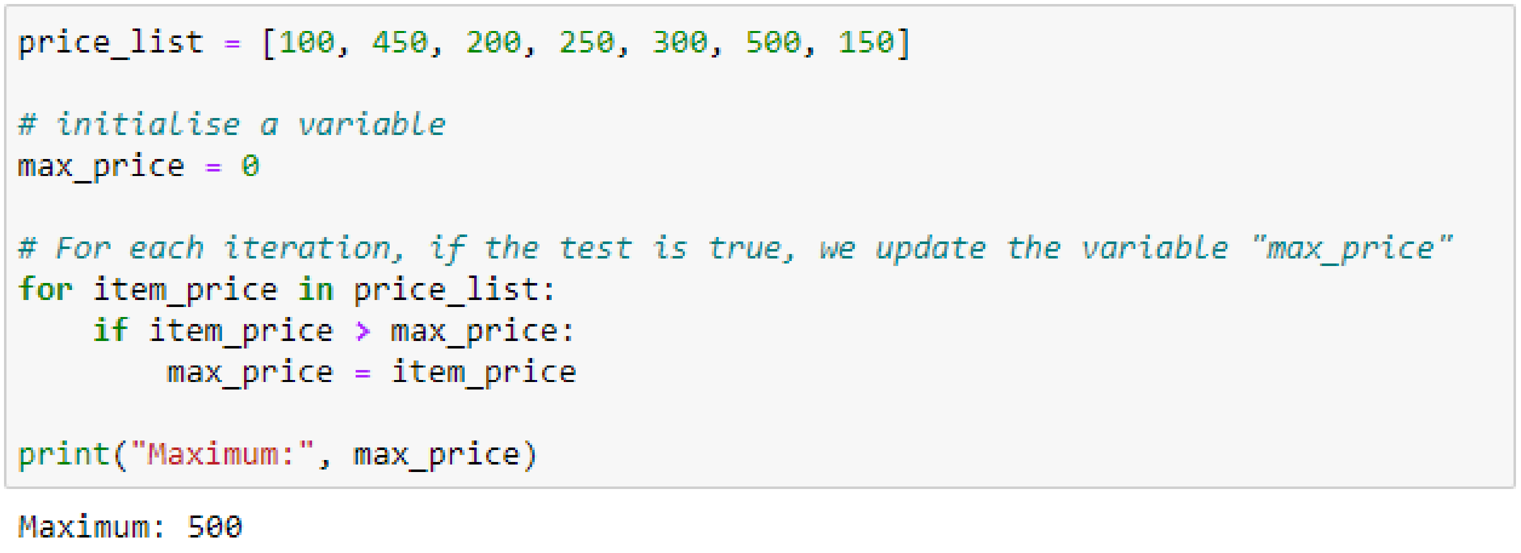
- Use a variable as a counter in a for loop’s body.

```python
animal_list = ['dog', 'dog', 'cat', 'dog', 'cat', 'cat', 'dog', 'cat', 'dog', 'cat']

# initialise a variable
counter = 0

# for each iteration, we update the variable "counter"
for animal in animal_list:
    if animal == 'dog':
        counter = counter +1

print('# of dogs:', counter)
```

```
# of dogs: 5
```

For loop – find the maximum number

- Find the maximum number in a list.



```
price_list = [100, 450, 200, 250, 300, 500, 150]

# initialise a variable
max_price = 0

# For each iteration, if the test is true, we update the variable "max_price"
for item_price in price_list:
    if item_price > max_price:
        max_price = item_price

print("Maximum:", max_price)
```

```
Maximum: 500
```

Iteration	Item_price	max_price
1	100	100
2	450	450
3	200	450
4	250	450
5	300	450
6	500	500
7	150	500

Exercise

(E.1) Print all numbers until you find a number that is divisible by 7.

```
num_list = [88,36,139,12,24,158,29,16,152,98,45,184,191,92,117]
```

(E.2) Count the frequency of the letter "A" in the list without using `list.count()` function.

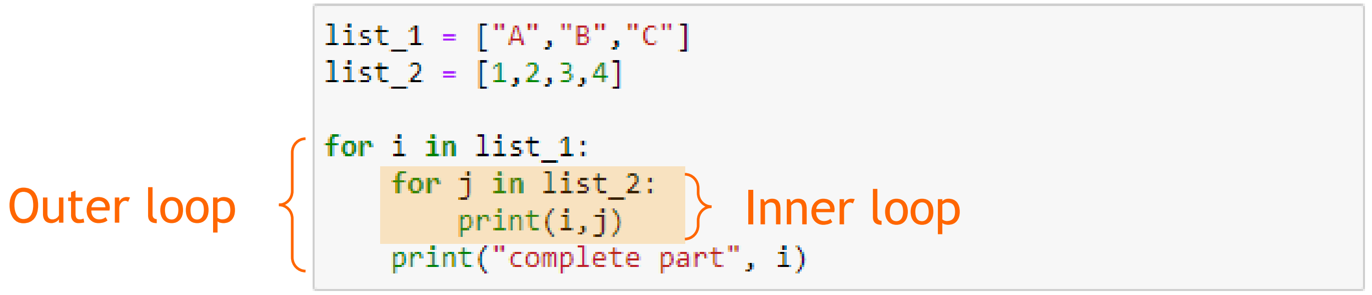
```
ABC_list = ['A','B','B','A','C','C','A','B','C','B','A','C','A','B','A','B','A','C','A','C']
```

(E.3) Find the longest company name in the list.

```
mylist = ["AirGarage","Airtable", "DoorDash","Dave","Bloomscape","Robinhood"]
```

For loop – nested loop

- A nested loop is a loop inside a loop.



```python
list_1 = ["A","B","C"]
list_2 = [1,2,3,4]

for i in list_1:
    for j in list_2:
        print(i,j)
    print("complete part", i)
```

```
A 1
A 2
A 3
A 4
complete part A
B 1
B 2
B 3
B 4
complete part B
C 1
C 2
C 3
C 4
complete part C
```

For loop – nested loop

- Example:
  - In each iteration of outer for loop, the inner for loop execute 3 times to print the current `i` and `j`.

```python
for i in range(1,4):
    for j in range(1,4):
        print ("{}-{}".format(i,j))
```

```
1-1
1-2
1-3
2-1
2-2
2-3
3-1
3-2
3-3
```

```python
for i in range(1,4):
    for j in range(1,4):
        print ("{}-{}".format(i,j), end = " ")
    print("\n")
```

```
1-1 1-2 1-3

2-1 2-2 2-3

3-1 3-2 3-3
```

Start a new line after each iteration of the outer for loop.

Exercise

(F.1) Suppose you want to order a drink and a dessert, print out all possible combinations.

Example:

Tea & Carrot cake

```
menu_beverages = ["Tea","Coffee","Juice","Water"]
menu_dessert = ["Carrot cake", "Lemon tart","Tiramisu", "Mousse"]
```

(F.2) Write a program to print out the following results.

```
1+1=2
1+2=3
1+3=4
2+1=3
2+2=4
2+3=5
3+1=4
3+2=5
3+3=6
```