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.
- Iteration is the repetition of a process in a computer program that can be done with the help of loops.
- 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.

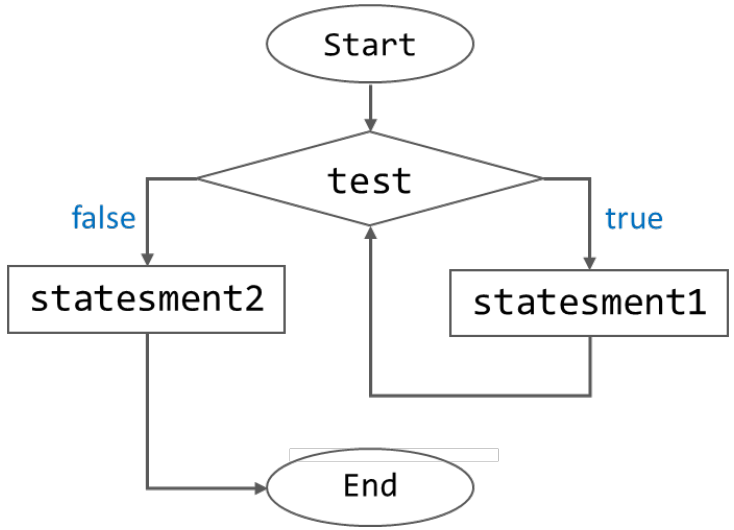
Assignment operator	Example	Same as
=	x = 1	x = 1
+=	x += 1	x= x+1
-=	x -= 1	x= x-1
*=	x *= 2	x= x*2
/=	x /= 2	x= x/2

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
      <statement1>     # loop body
else:                  # optional else
      <statement2>     # run if the test is false
```



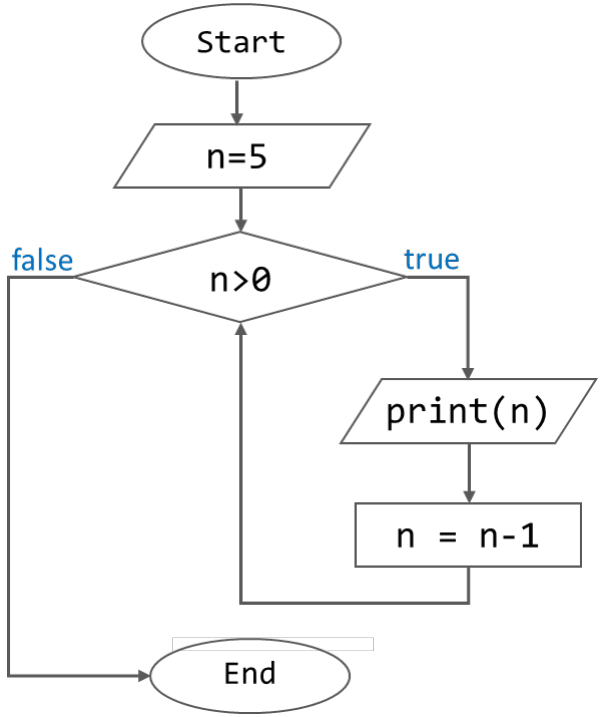
While loop

- While n is greater than 0, display the value of n and then decrement 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	

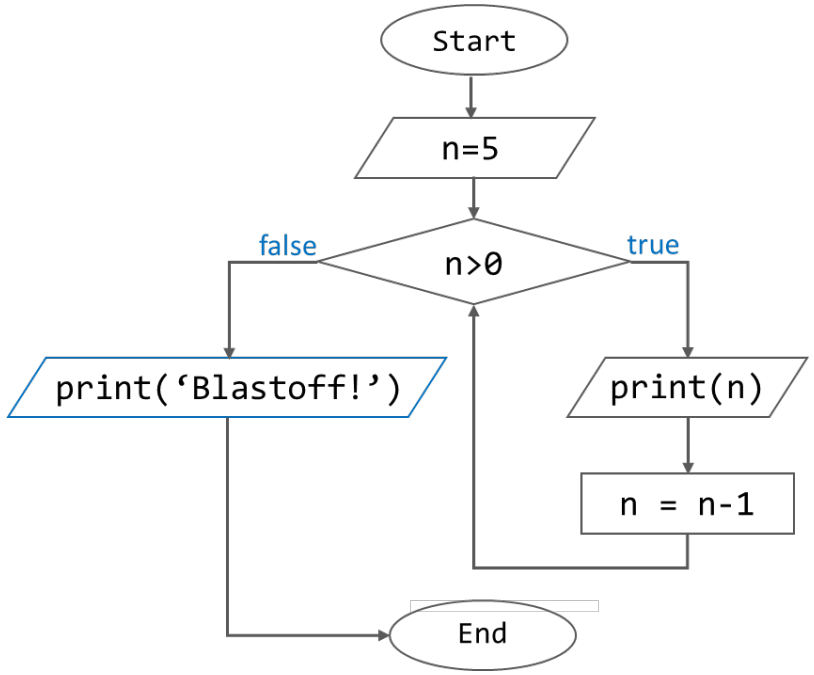


While loop - else

- 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 this line 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

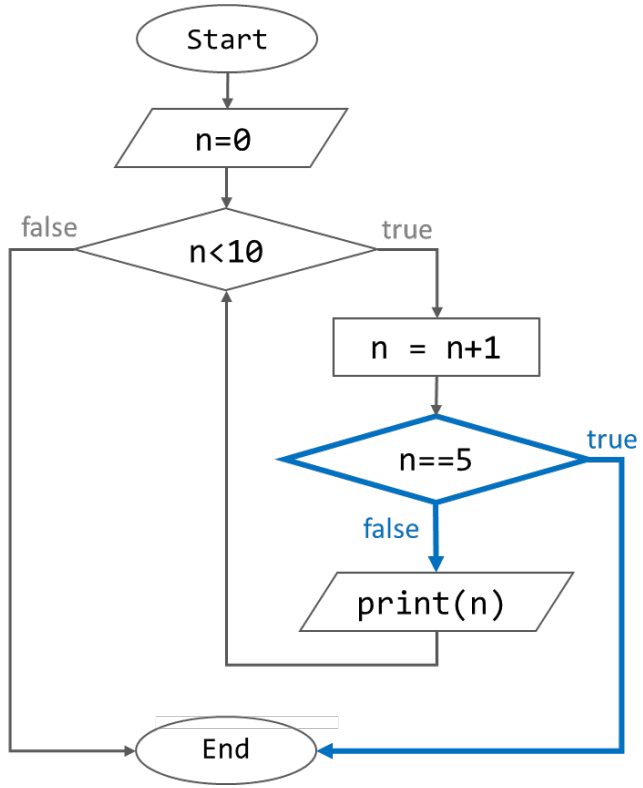
```
While <test1>:
        <statement1>
        if <test2>:
                break       # Exit  loop now, skip else
        if <test3>:
                continue    # Go to top of loop now, to test1
        <statement2>
else:
        <statement3>        # run if we didn't hit a 'break'
```

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
```

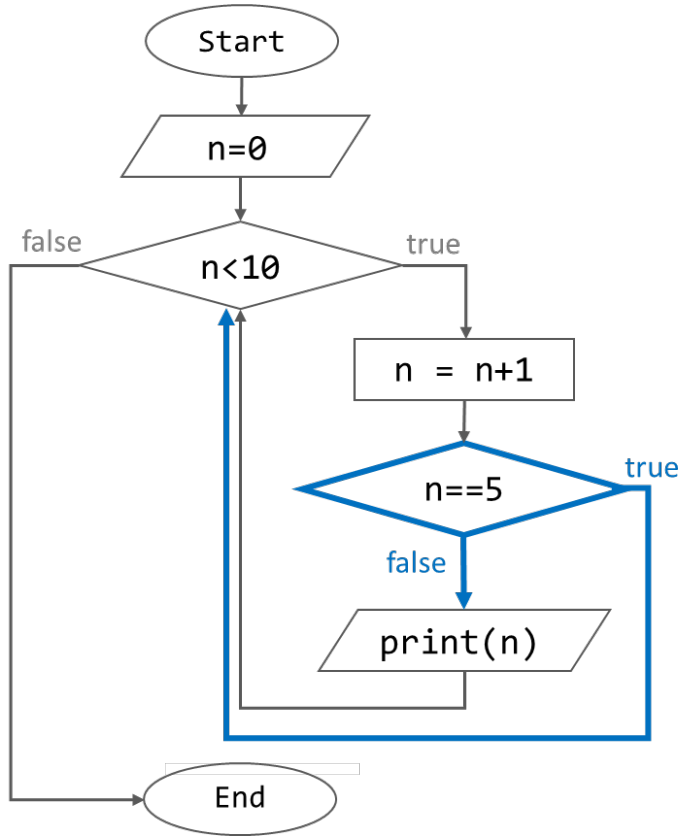


While loop - continue

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

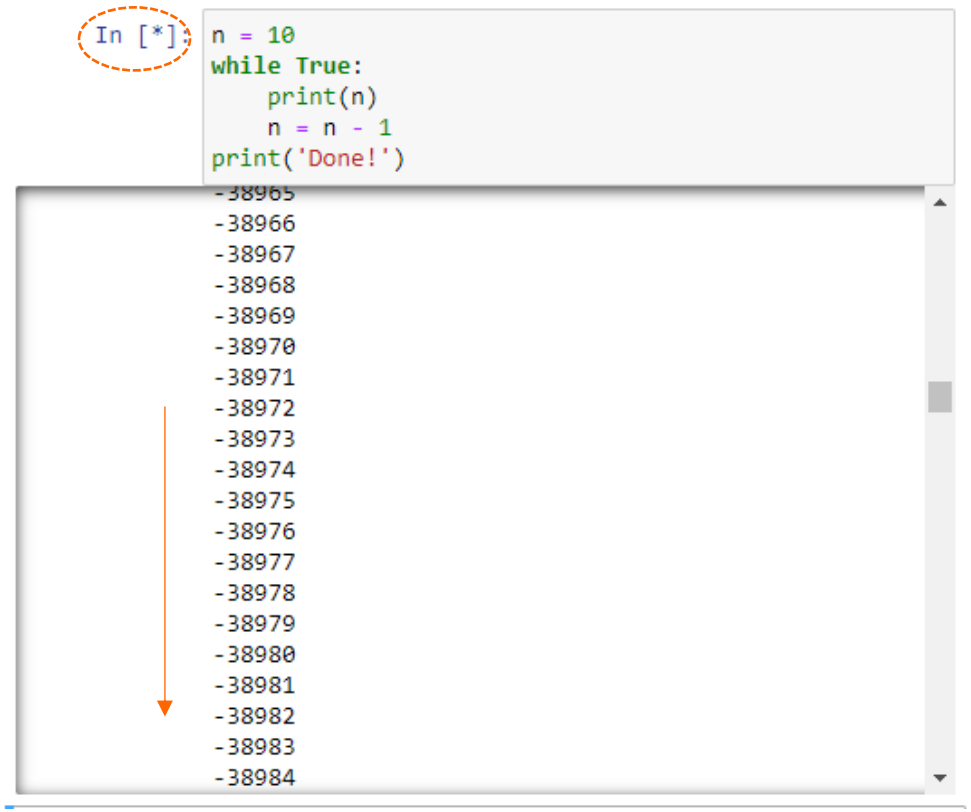
```
n = 0
while n<10:
    n = n + 1

    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?



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

Exercise

Exercise.B

(B.1) Write a program that asks the user to enter a number between 1 and 20, and prints all numbers from 1 to 20, except for the entered number.
(Use `end = " "` to print all numbers on one line.)

Example-1:
Enter a number: `10`
1 2 3 4 5 6 7 8 **9** **11** 12 13 14 15 16 17 18 19 20

Example-2:
Enter a number: `18`
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 **17** **19** 20

```
```

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>:
        <statements>
```

Assign the next item in the object to the target

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

Iteration	i	Print (i)
1	i =“p”	p
2	i =“y”	y
3	i =“t”	t
4	i =“h”	h
5	i =“o”	o
6	i =“n”	n

For loop – list

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

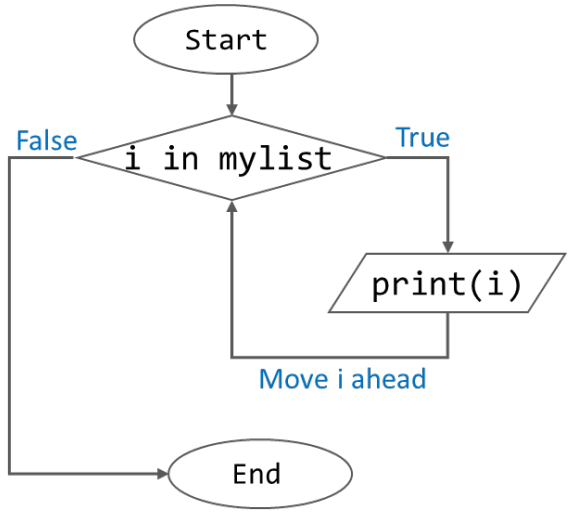
```
mylist = [5, 4, 3, 2, 1]
for i in mylist:
    print(i)
```

```
5
4
3
2
1
```

Iteration	i	Print (i)
1	i = 5	5
2	i = 4	4
3	i = 3	3
4	i = 2	2
5	i = 1	1

```
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 incrementing by 1 (by default), and stopping before the 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
```

- https://www.geeksforgeeks.org/python-range-function/

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.

Expected result:
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 15 to 25 on one line.
Expected result:
15 16 17 18 19 20 21 22 23 24 25

```
```

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

```
#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

Exercise.D

(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: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
```

```

```

For loop – break & continue

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

```
for i in range(0,10):
    if i == 5:
        break # Exit  loop now
    print(i)
```

```
0
1
2
3
4
```

```
for i in range(0,10):
    if i == 5:
        continue  # Go to top of loop now
    print(i)
```

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

For loop – Application (1) 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 - Application (2) counter

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

```
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 – Application (3) find maximum number

- Find the maximum number in a list.

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

# Initially take the first element as the largest number
max_price = price_list[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

Exercise.E

(E.1) Print all numbers in the list `num_list` until a number divisible by 7 is found.

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

```

```

(E.2) Given the following list. How many names start with "S"?

Hint: Initialize a variable as a counter.

```
name_list = ["Henry", "Victoria","Isaac","Sara", "Maya", "Zoe","Isabelle", "Nora", "Madelyn",
             "Sophia","Charlotte", "Michael","Sebastian","Leah", "Ryan", 'Matthew', "Mila"]
```

```

```

For loop – nested loop

- A nested loop is a loop inside a loop.

Outer loop

```
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)
```

Inner loop

```
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
```

i	j	print(i, j)
A	1	A 1
	2	A 2
	3	A 3
	4	A 4
B	1	B 1
	2	B 2
	3	B 3
	4	B 4
C	1	C 1
	2	C 2
	3	C 3
	4	C 4

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`.

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

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

```
for i in range(1,4):
    for j in range(1,4):
        print (f"{i}-{j}", end = " ")
    print()
```

```
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

Exercise.F

(F.1) Suppose you want to order a drink and a dessert, print out all possible combinations.
Example:
Tea & Carrot cake
Tea & Lemon tart
...
Water & Mousse

```
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
```

```

```