Structure

NEE2106 Computer Programming for Electrical Engineers

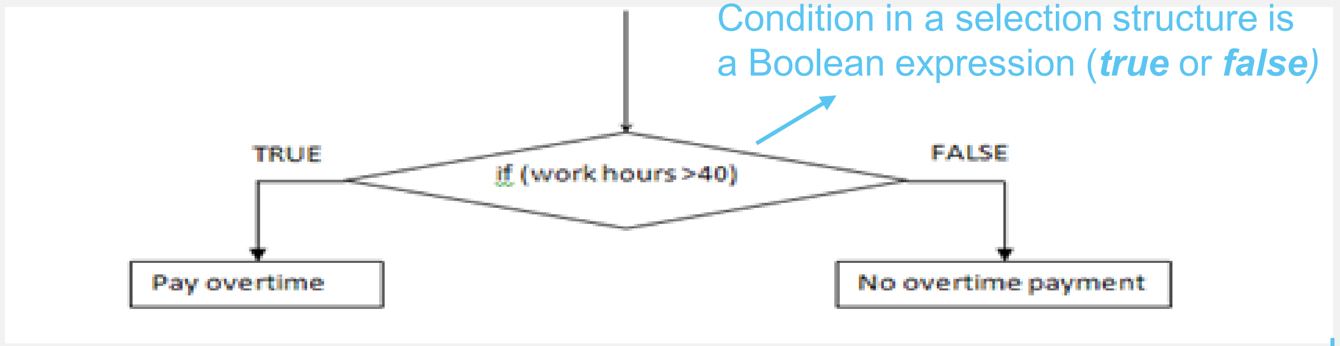
Prepared by: Dr. Rui Li (rui.li@vu.edu.au)

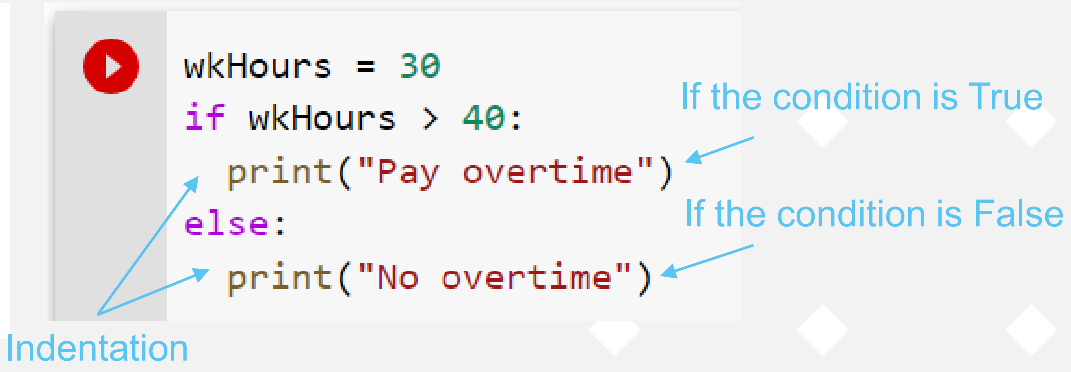
< Contents >

< Selection: if - else >

Selection structure selects which block of codes to be executed next

- **if….else….**
- **if….elif….else…**


Condition in a selection structure is a Boolean expression (***true*** or ***false***)
TRUE
FALSE
if (work hours >40)
Pay overtime
No overtime payment



```
wkHours = 30
if wkHours > 40:
  print("Pay overtime")
else:
  print("No overtime")
```

- Indentation is important in Python to mark a block of code to execute
- In if - else statement, indentation is used to indicate which code block will be executed depending on which condition
- Condition terminates with a colon “**:**”

< Boolean Data Type and Comparison >

In Python, we are using the following comparison operators to determine a Boolean value from a condition:

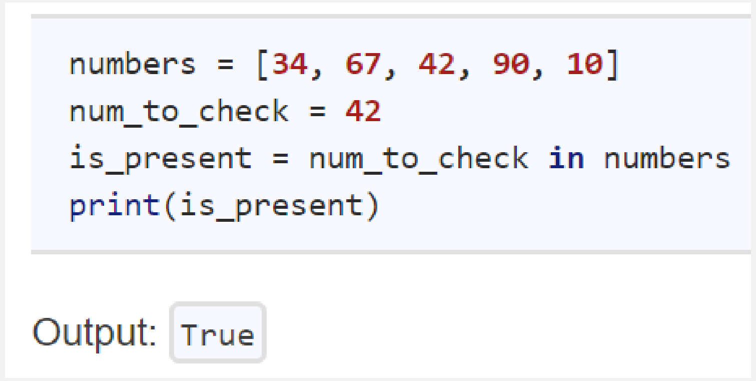
Operator Sign	Operator Name
<	Less than
>	Greater than
<=	Less than or equal to
>=	Greater than or equal to
==	Equal to
!=	Not equal to

Operator	Meaning	Example	Result
and	Logical and	(5<2) and (5>3)	False
or	Logical or	(5<2) or (5>3)	True
not	Logical not	not (5<2)	True

X	Y	X and Y	X or Y	not(X)	not(Y)
T	T	T	T	F	F
T	F	F	T	F	T
F	T	F	T	T	F
F	F	F	F	T	T

The '**in**' keyword in Python can check for the presence of an element in a list.

- True if the specified element is found
- False if not

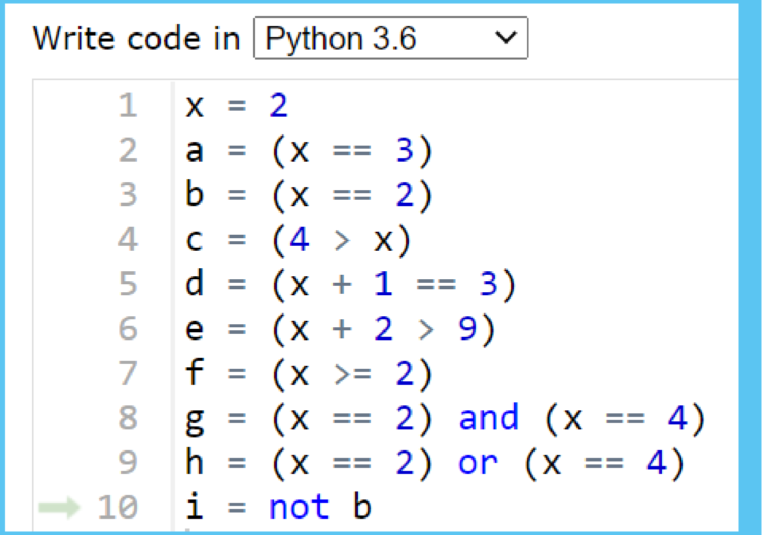


```python
numbers = [34, 67, 42, 90, 10]
num_to_check = 42
is_present = num_to_check in numbers
print(is_present)
```

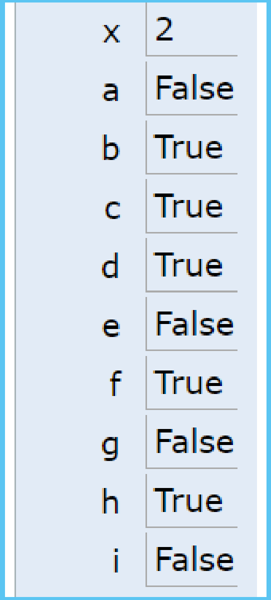
Output:

```
True
```

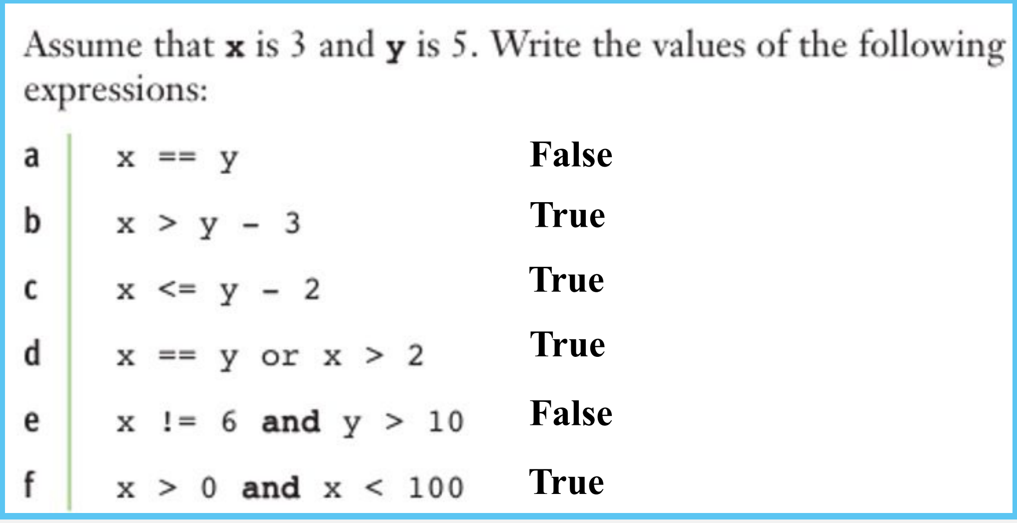
< Example of Comparison >



```python
x = 2
a = (x == 3)
b = (x == 2)
c = (4 > x)
d = (x + 1 == 3)
e = (x + 2 > 9)
f = (x >= 2)
g = (x == 2) and (x == 4)
h = (x == 2) or (x == 4)
i = not b
```



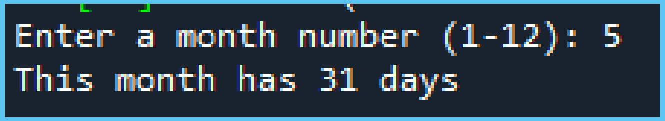
x	2
a	False
b	True
c	True
d	True
e	False
f	True
g	False
h	True
i	False



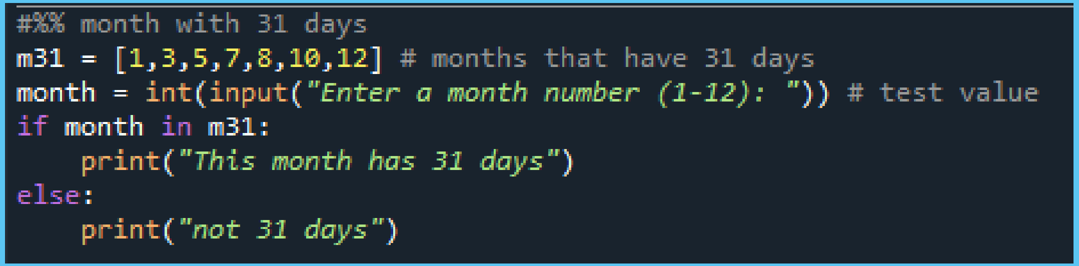
Assume that **x** is 3 and **y** is 5. Write the values of the following expressions:

a	`x == y`	**False**
b	`x > y - 3`	**True**
c	`x <= y - 2`	**True**
d	`x == y or x > 2`	**True**
e	`x != 6 and y > 10`	**False**
f	`x > 0 and x < 100`	**True**

Write a Python program that takes a user input for a month number (1-12) and displays whether the month entered has 31 days.
Tip: use the Python keyword “in”



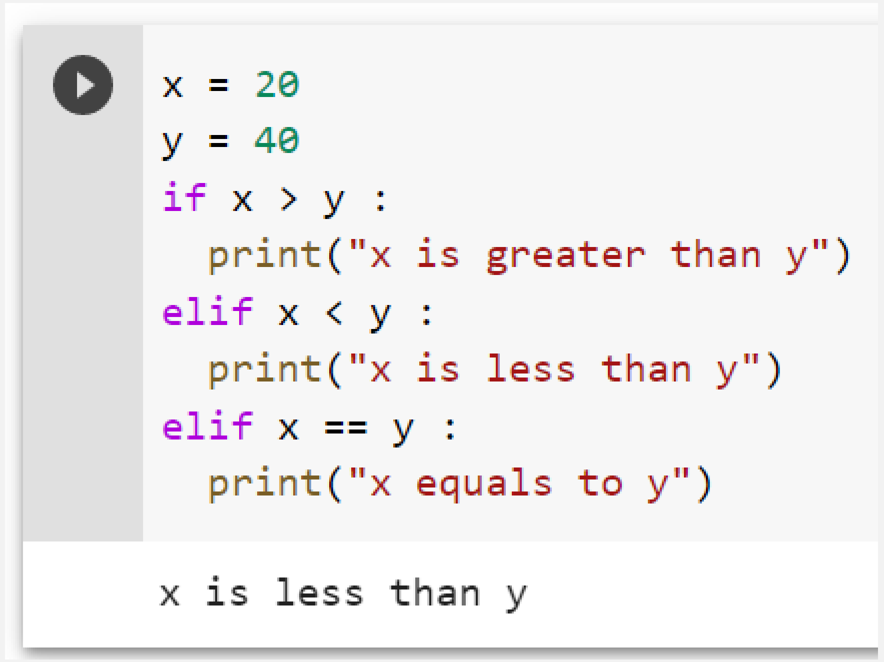
```
Enter a month number (1-12): 5
This month has 31 days
```



```python
#%% month with 31 days
m31 = [1,3,5,7,8,10,12] # months that have 31 days
month = int(input("Enter a month number (1-12): ")) # test value
if month in m31:
    print("This month has 31 days")
else:
    print("not 31 days")
```

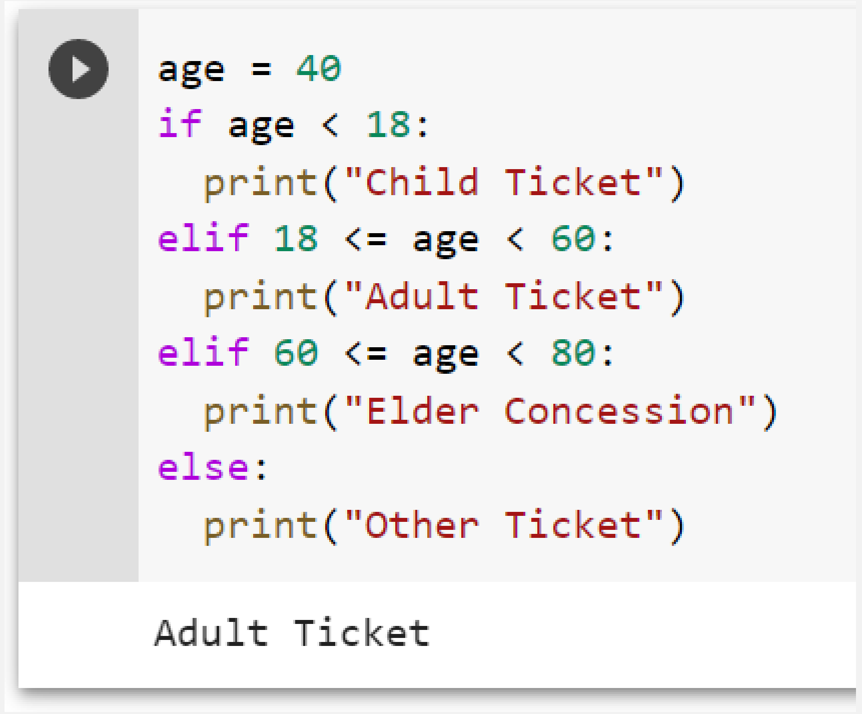
< Selection: if – elif- else >

- Used when there are more than two selection paths
- We can have multiple "*elif*" in the statement
- The ending "else" is NOT mandatory



```python
x = 20
y = 40
if x > y :
  print("x is greater than y")
elif x < y :
  print("x is less than y")
elif x == y :
  print("x equals to y")
```

```
x is less than y
```



```python
age = 40
if age < 18:
  print("Child Ticket")
elif 18 <= age < 60:
  print("Adult Ticket")
elif 60 <= age < 80:
  print("Elder Concession")
else:
  print("Other Ticket")
```

```
Adult Ticket
```

Q: how do you conduct an interval checking in Matlab?

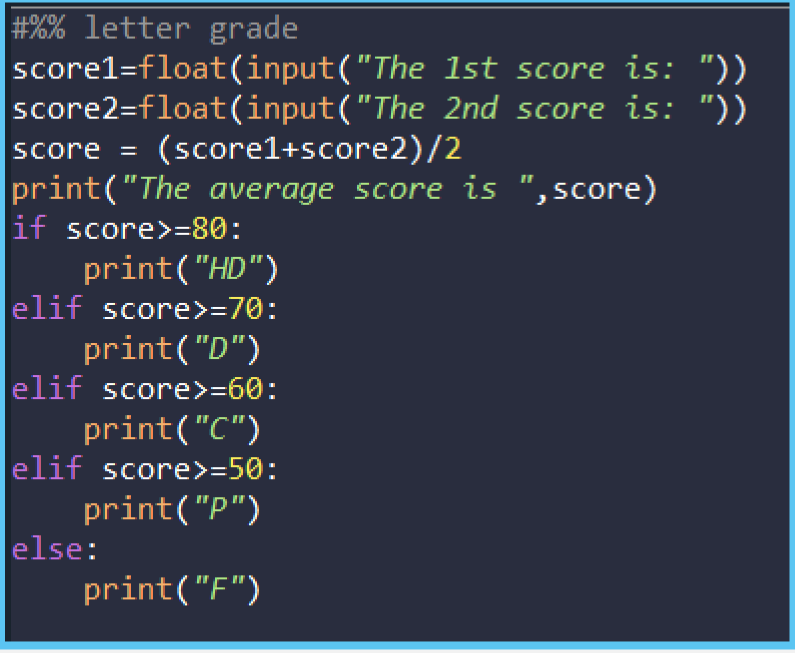
A: 18<= age && age<=60

< Exercise >

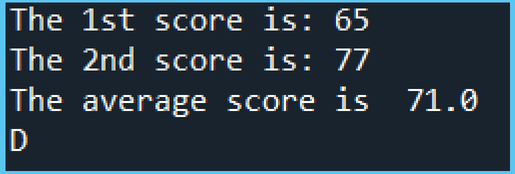
We have two test scores for a student. The two scores are added and then divided by 2 for the average score. A letter Grade is then worked out for that average score and is displayed using:

- 80 and above = grade of HD
- 70 and above = grade of D
- 60 and above = grade of C
- 50 and above = grade of P
- Below 50 = grade of F

(Only one of these Grades can be applied to a student)



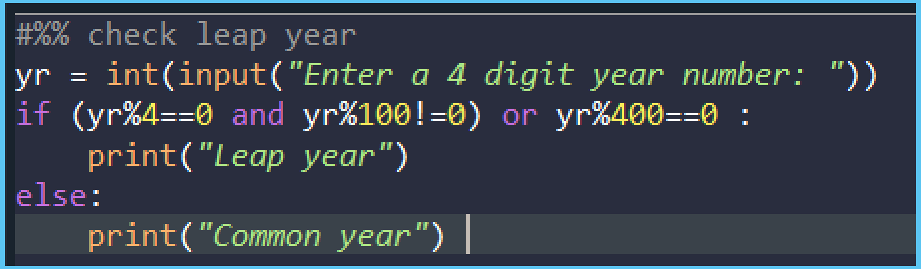
```python
#%% letter grade
score1=float(input("The 1st score is: "))
score2=float(input("The 2nd score is: "))
score = (score1+score2)/2
print("The average score is ",score)
if score>=80:
    print("HD")
elif score>=70:
    print("D")
elif score>=60:
    print("C")
elif score>=50:
    print("P")
else:
    print("F")
```



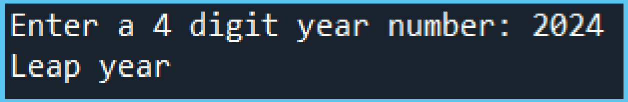
```
The 1st score is: 65
The 2nd score is: 77
The average score is  71.0
D
```

< Exercise >

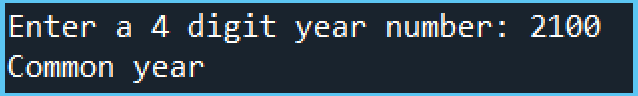
Write a Python program to check if the entered year is a leap year.
A leap year is the year that is evenly divisible by 4 and not divisible by 100, OR the year is evenly divisible by 400.



```python
#%% check leap year
yr = int(input("Enter a 4 digit year number: "))
if (yr%4==0 and yr%100!=0) or yr%400==0 :
    print("Leap year")
else:
    print("Common year")
```



```
Enter a 4 digit year number: 2024
Leap year
```



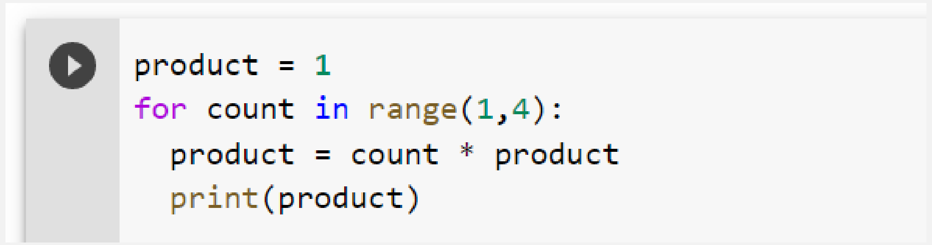
```
Enter a 4 digit year number: 2100
Common year
```

< Contents >

< Loop >

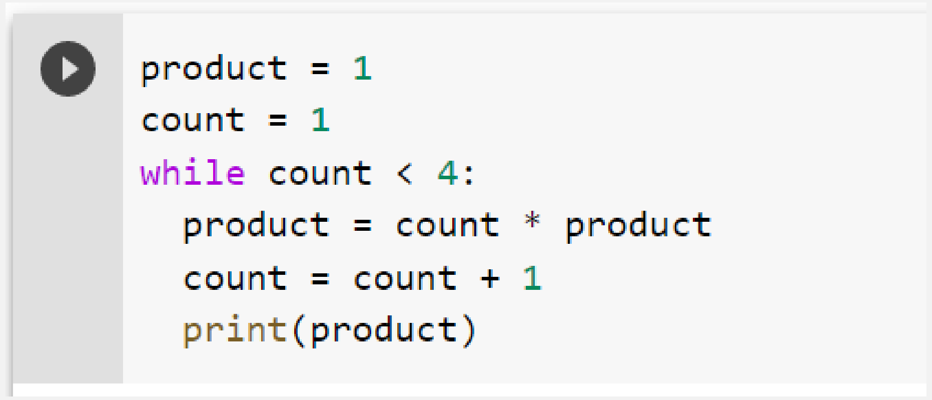
"Loop" is a repetition structure that enables to execute the same code many times

- **"for"** loop - definite interaction, loops with predetermine number of time



```
product = 1
for count in range(1,4):
  product = count * product
  print(product)
```

- **"while"** loop - indefinite Iteration, the program determine when to stop

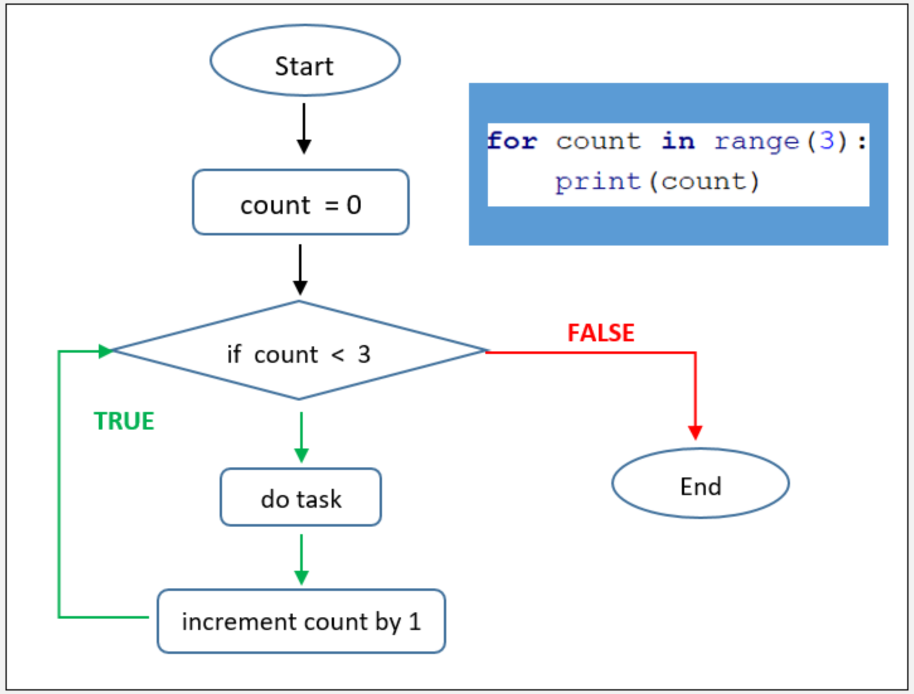


```
product = 1
count = 1
while count < 4:
  product = count * product
  count = count + 1
  print(product)
```

Same results !

< “For” Loop to iterate over a range>

“for” loop is normally a count-controlled loop that is useful for looping over a list of values



```python
for count in range(3):
    print(count)
```

range( ) is a built-in function used to generate a sequence of numbers.
range(start, stop, step) :

- Start: (optional) the starting value, 0 by default
- Stop: the ending value, the sequence includes numbers up to, but NOT include, this value
- Step: (optional) the increment between numbers, 1 by default

Q: what are the values in range(3)?

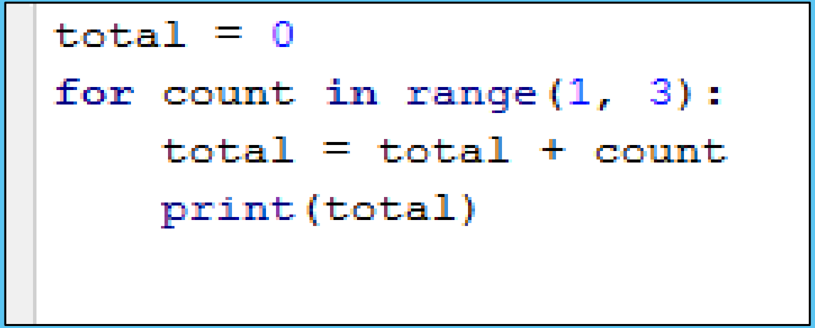
A: 0, 1, 2

Q: what are the values in range(2,10,2)?

A: 2, 4, 6, 8

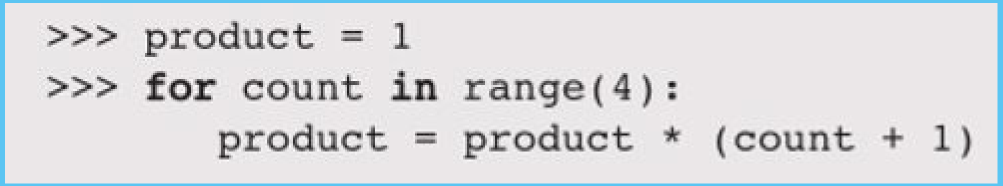
< Example >

What are the values in **count** and **total** in each iteration in the following program?



```
total = 0
for count in range(1, 3):
    total = total + count
    print(total)
```

1st iteration	count = 1	total = 0+1 = 1
2nd iteration	count = 2	total = 1+2 = 3

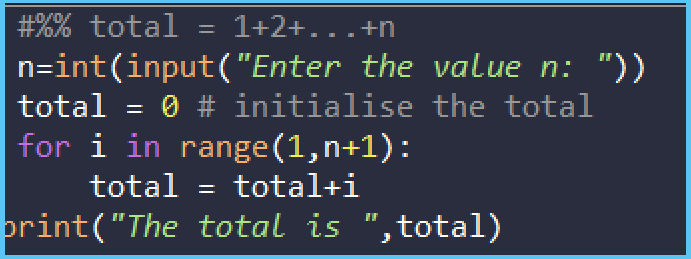


```
>>> product = 1
>>> for count in range(4):
        product = product * (count + 1)
```

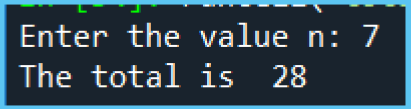
1st iteration	count = 0	product = 1*(0+1) = 1
2nd iteration	count = 1	product = 1*(1+1) = 2
3rd iteration	count = 2	product = 2*(2+1) = 6
4th iteration	count = 3	product = 6*(3+1) = 24

< Exercise >

Use a loop to compute total = 1+2+3+4+….+n, where n is entered by the user.

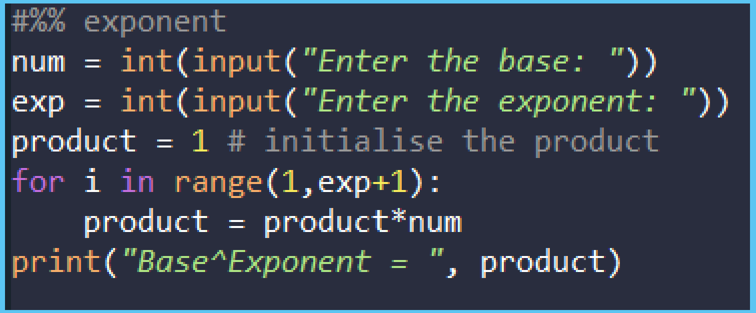


```
#%% total = 1+2+...+n
n=int(input("Enter the value n: "))
total = 0 # initialise the total
for i in range(1,n+1):
    total = total+i
print("The total is ",total)
```

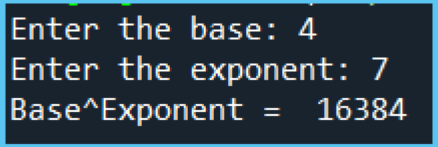


```
Enter the value n: 7
The total is  28
```

Use a loop to compute product = num ^ exp, where **num** is the base and **exp** is the exponent. Both values are entered by the end user.



```
#%% exponent
num = int(input("Enter the base: "))
exp = int(input("Enter the exponent: "))
product = 1 # initialise the product
for i in range(1,exp+1):
    product = product*num
print("Base^Exponent = ", product)
```



```
Enter the base: 4
Enter the exponent: 7
Base^Exponent =  16384
```

< Some tricks >

Augmented assignment: assignment symbol can be combined with the arithmetic and concatenation operators to provide augmented assignment operations

```
a = 17
s = "hi"

a += 3              # Equivalent to a = a + 3
a -= 3              # Equivalent to a = a - 3
a *= 3              # Equivalent to a = a * 3
a /= 3              # Equivalent to a = a / 3
a %= 3              # Equivalent to a = a % 3
s += " there"       # Equivalent to s = s + " there"
```

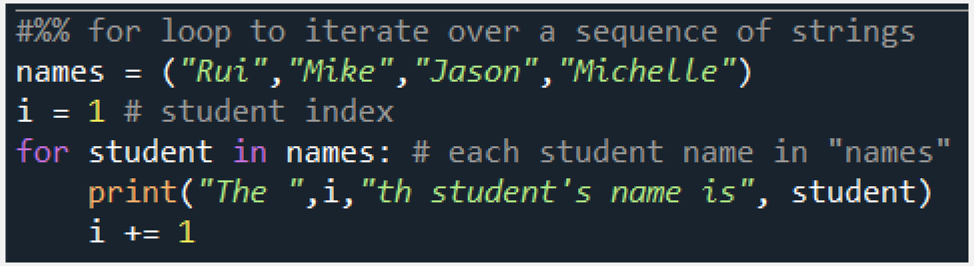
Traversing: list all the contents of a sequence

```
>>> for number in [1, 2, 3]:
        print(number, end = " ")

1 2 3
>>> for character in "Hi there!":
        print(character, end = " ")

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

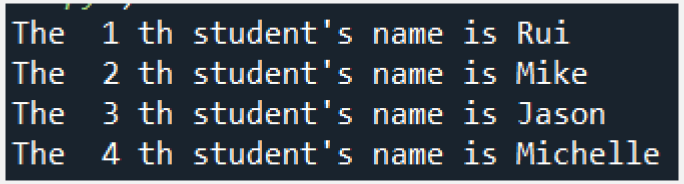
```
>>> list(range(4))
[0, 1, 2, 3]
>>> list(range(1, 5))
[1, 2, 3, 4]
>>>
```

< "For" Loop to iterate over a sequence >

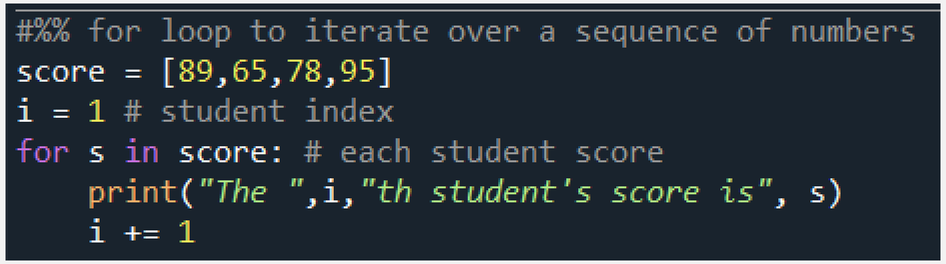
"for" loop to iterate over each element in a sequence:



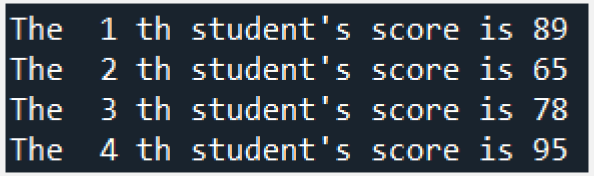
```python
#%% for loop to iterate over a sequence of strings
names = ("Rui","Mike","Jason","Michelle")
i = 1 # student index
for student in names: # each student name in "names"
    print("The ",i,"th student's name is", student)
    i += 1
```



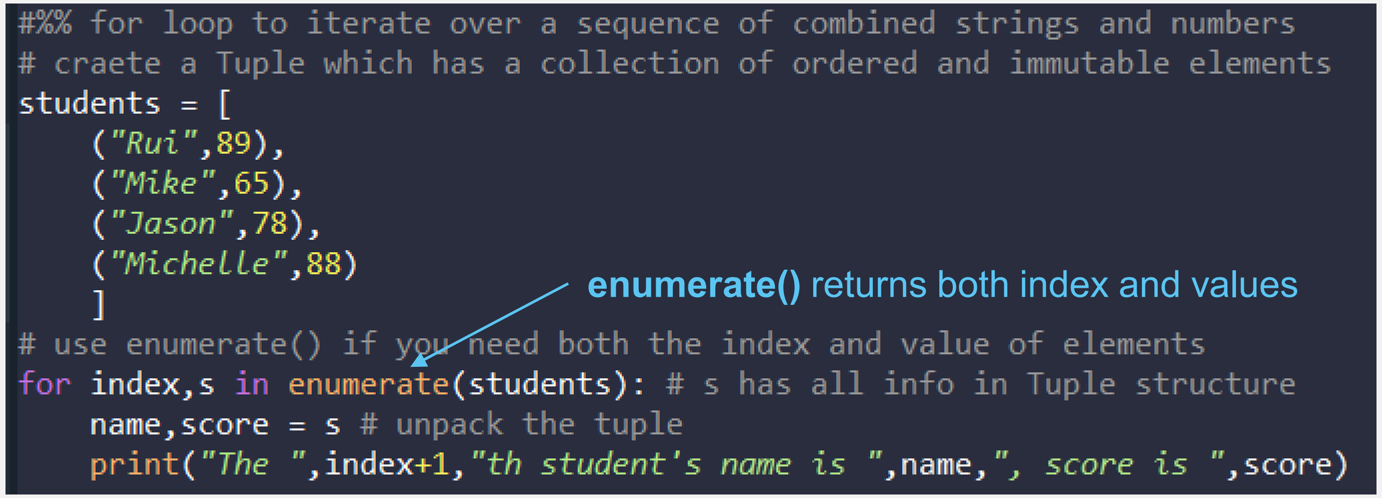
```
The  1 th student's name is Rui
The  2 th student's name is Mike
The  3 th student's name is Jason
The  4 th student's name is Michelle
```



```python
#%% for loop to iterate over a sequence of numbers
score = [89,65,78,95]
i = 1 # student index
for s in score: # each student score
    print("The ",i,"th student's score is", s)
    i += 1
```

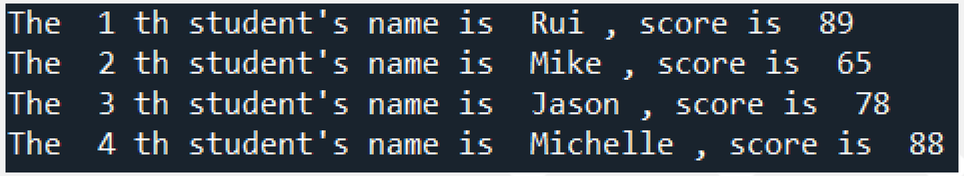


```
The  1 th student's score is 89
The  2 th student's score is 65
The  3 th student's score is 78
The  4 th student's score is 95
```



```python
#%% for loop to iterate over a sequence of combined strings and numbers
# craete a Tuple which has a collection of ordered and immutable elements
students = [
    ("Rui",89),
    ("Mike",65),
    ("Jason",78),
    ("Michelle",88)
    ]
# use enumerate() if you need both the index and value of elements
for index,s in enumerate(students): # s has all info in Tuple structure
    name,score = s # unpack the tuple
    print("The ",index+1,"th student's name is ",name,", score is ",score)
```

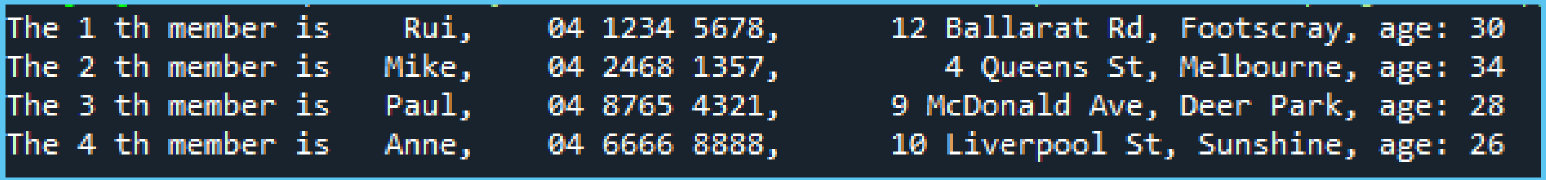
enumerate() returns both index and values



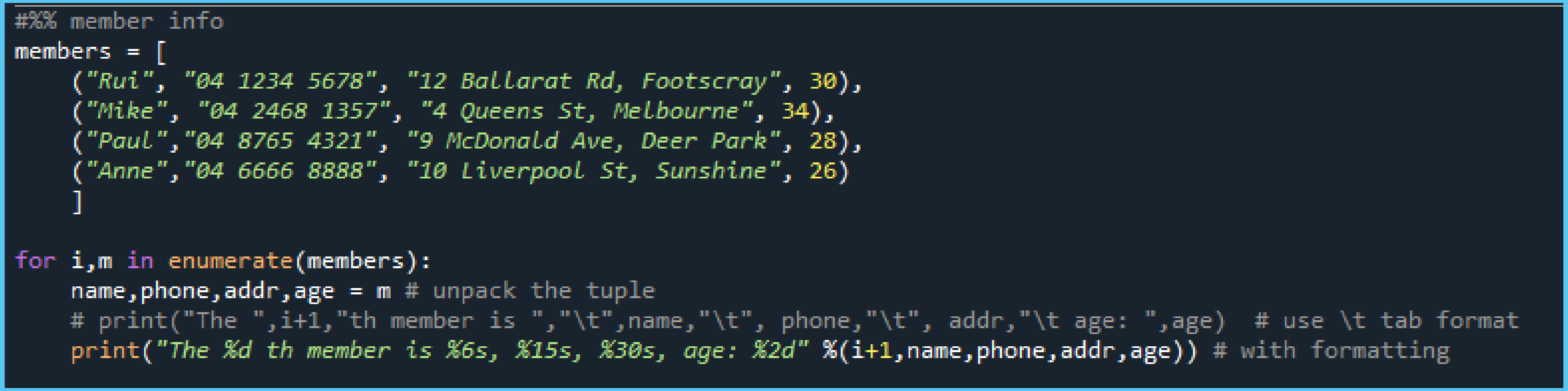
```
The  1 th student's name is  Rui , score is  89
The  2 th student's name is  Mike , score is  65
The  3 th student's name is  Jason , score is  78
The  4 th student's name is  Michelle , score is  88
```

< Exercise >

Collect 3 students' personal information that includes
Name, **Phone number**, **Address, and Age** (no need to be real)
and define such information in a Tuple structure named "**members**".
Write a **for** loop to print each member's personal information.



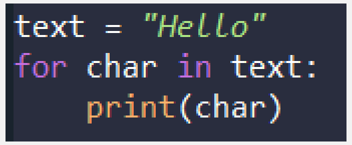
```
The 1 th member is    Rui,    04 1234 5678,      12 Ballarat Rd, Footscray, age: 30
The 2 th member is   Mike,    04 2468 1357,         4 Queens St, Melbourne, age: 34
The 3 th member is   Paul,    04 8765 4321,      9 McDonald Ave, Deer Park, age: 28
The 4 th member is   Anne,    04 6666 8888,      10 Liverpool St, Sunshine, age: 26
```



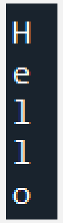
```
#%% member info
members = [
    ("Rui", "04 1234 5678", "12 Ballarat Rd, Footscray", 30),
    ("Mike", "04 2468 1357", "4 Queens St, Melbourne", 34),
    ("Paul","04 8765 4321", "9 McDonald Ave, Deer Park", 28),
    ("Anne","04 6666 8888", "10 Liverpool St, Sunshine", 26)
    ]

for i,m in enumerate(members):
    name,phone,addr,age = m # unpack the tuple
    # print("The ",i+1,"th member is ","\t",name,"\t", phone,"\t", addr,"\t age: ",age)  # use \t tab format
    print("The %d th member is %6s, %15s, %30s, age: %2d" %(i+1,name,phone,addr,age)) # with formatting
```

< other “for” Loop >

Iterate over a string:

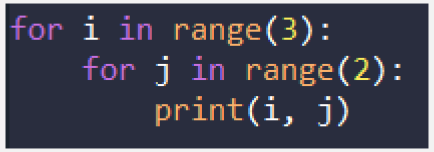


```python
text = "Hello"
for char in text:
    print(char)
```

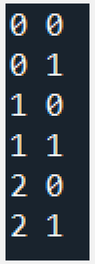


```
H
e
l
l
o
```

Nested Loops: nest for loops within each other to create nested iterations

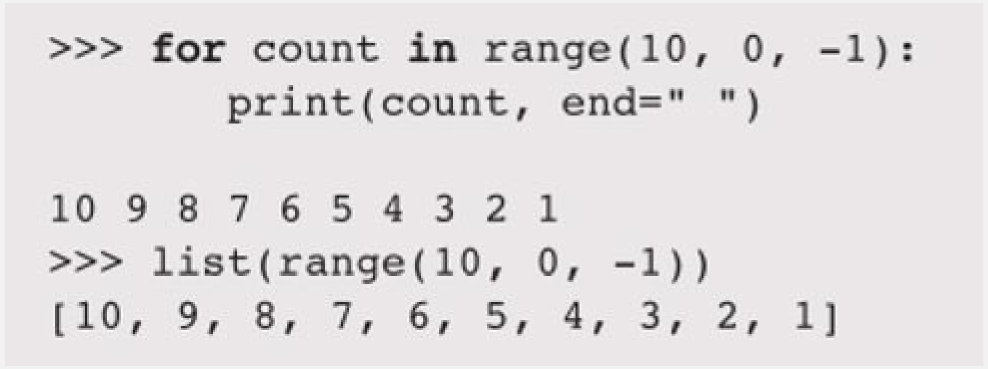


```python
for i in range(3):
    for j in range(2):
        print(i, j)
```

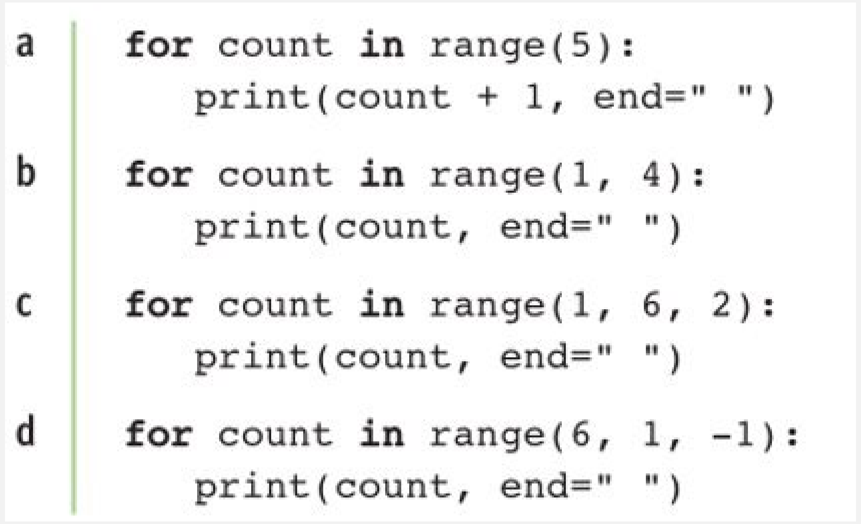


```
0 0
0 1
1 0
1 1
2 0
2 1
```

< Count Down Loop >



```
>>> for count in range(10, 0, -1):
        print(count, end=" ")

10 9 8 7 6 5 4 3 2 1
>>> list(range(10, 0, -1))
[10, 9, 8, 7, 6, 5, 4, 3, 2, 1]
```



```
a   for count in range(5):
       print(count + 1, end=" ")
b   for count in range(1, 4):
       print(count, end=" ")
c   for count in range(1, 6, 2):
       print(count, end=" ")
d   for count in range(6, 1, -1):
       print(count, end=" ")
```

1 2 3 4 5

1 2 3

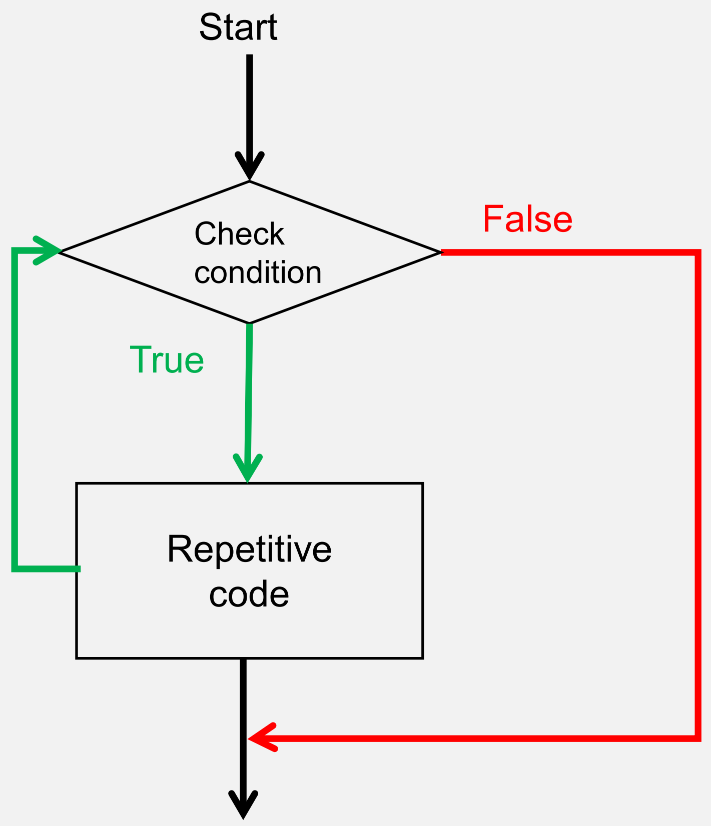
1 3 5

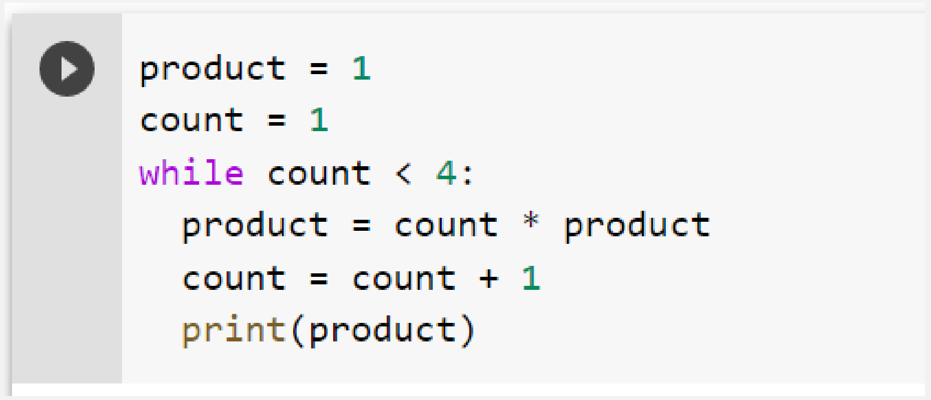
6 5 4 3 2

< Contents >

< "while" loop >

- While loop displays conditional iteration.
- Based on the value of the condition, the program determine when to stop



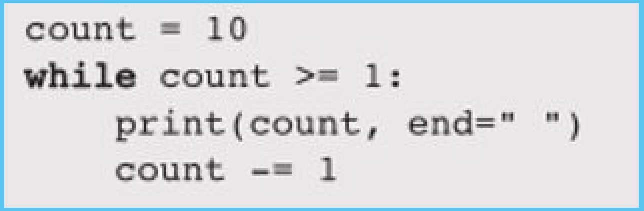


```
product = 1
count = 1
while count < 4:
  product = count * product
  count = count + 1
  print(product)
```

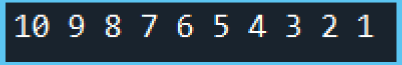
1st iteration	count = 1	"while count<4" is True	product = 1
2nd iteration	count = 2	"while count<4" is True	product = 2
3rd iteration	count = 3	"while count<4" is True	product = 6
4th iteration	count = 4	"while count<4" is False	Jump out of the loop

< Exercise >

Q1: What are the outputs of the following “while” loop?

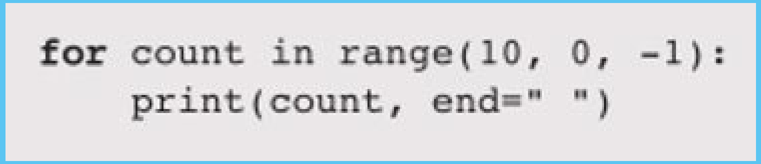


```
count = 10
while count >= 1:
    print(count, end=" ")
    count -= 1
```



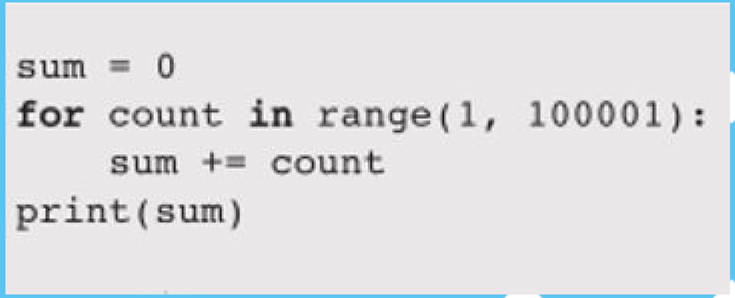
```
10 9 8 7 6 5 4 3 2 1
```

What is the equivalent “for” loop for the above?

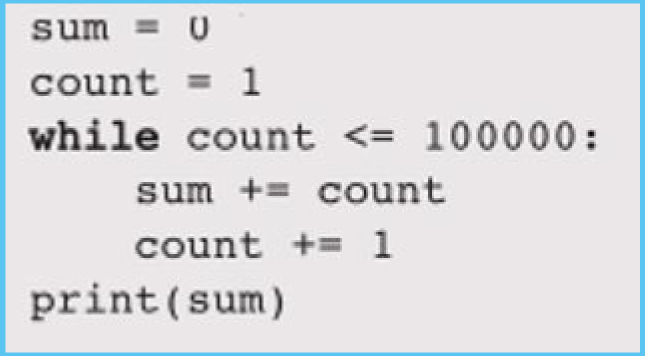


```
for count in range(10, 0, -1):
    print(count, end=" ")
```

Q2: How to rewrite the following “for” loop using a “while” loop?



```
sum = 0
for count in range(1, 100001):
    sum += count
print(sum)
```

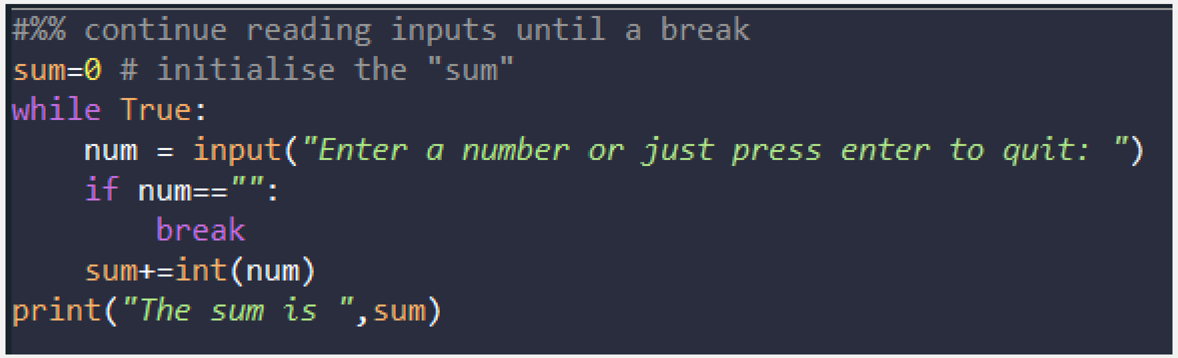


```
sum = 0
count = 1
while count <= 100000:
    sum += count
    count += 1
print(sum)
```

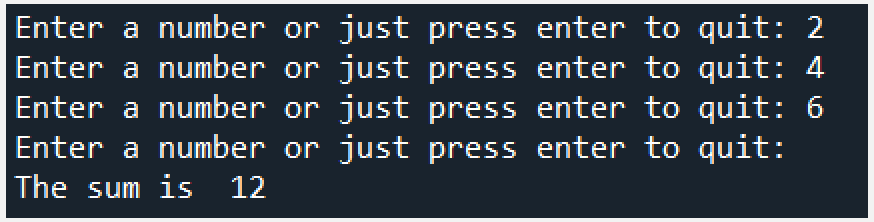
< “break” from the loop>

break statement will cause an exit from the loop, no further iteration will be executed

e.g. Write a Python program to calculate the sum of all the user-entered numbers. The program should continue reading numbers from the user until a blank line is entered (i.e. press enter to quit). Once a blank line is detected, the program should display the sum of all the entered numbers.



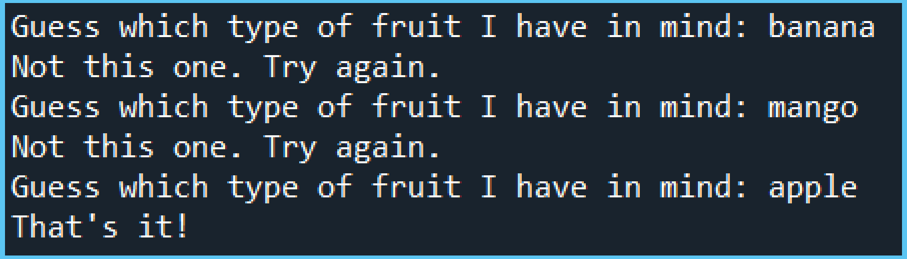
```python
#%% continue reading inputs until a break
sum=0 # initialise the "sum"
while True:
    num = input("Enter a number or just press enter to quit: ")
    if num=="":
        break
    sum+=int(num)
print("The sum is ",sum)
```



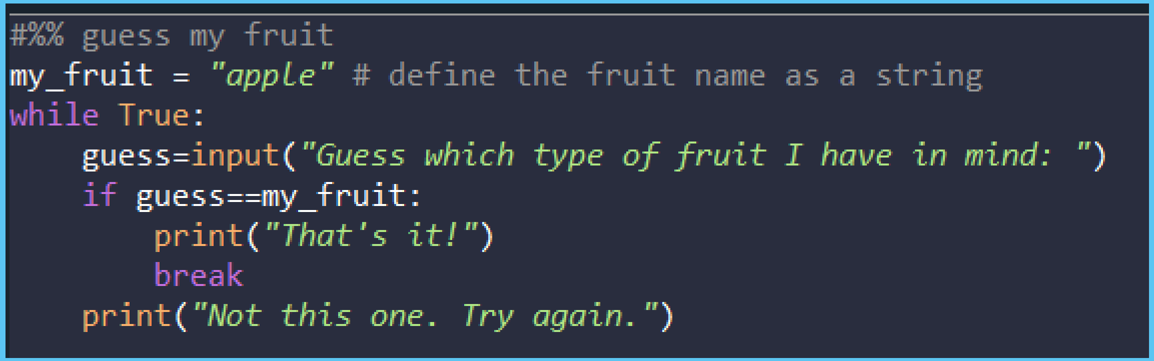
```
Enter a number or just press enter to quit: 2
Enter a number or just press enter to quit: 4
Enter a number or just press enter to quit: 6
Enter a number or just press enter to quit:
The sum is  12
```

< Exercise >

Write a Python program that asks the user to guess the type of fruit you have in mind. The program should continue prompting the user for guesses until they correctly guess the fruit you are thinking of.



```
Guess which type of fruit I have in mind: banana
Not this one. Try again.
Guess which type of fruit I have in mind: mango
Not this one. Try again.
Guess which type of fruit I have in mind: apple
That's it!
```



```python
#%% guess my fruit
my_fruit = "apple" # define the fruit name as a string
while True:
    guess=input("Guess which type of fruit I have in mind: ")
    if guess==my_fruit:
        print("That's it!")
        break
    print("Not this one. Try again.")
```