Loops (while, for)

while Loop

```
while condition1:
    indented_block
```

- With a while statement, the first thing that happens is that the condition is evaluated before the intendeted block of code .
- If the condition evaluates to False, then the statements in the intendeted block of code are never executed.
- If the condition evaluates to True, then the while loop block (intendeted block of code) is executed.
- After each iteration of the loop block, the condition is again checked, and if it is True, the intendeted block of code is iterated again.
- Each repetition of the loop block (intendeted block of code) is called an iteration of the loop.
- This process continues until the condition evaluates to False and at this point the while statement exits.
- Execution then continues with the first statement after the while loop.

Example1

- Write Python Program to Display First 10 Numbers Using while Loop Starting from 0

```
1 i = 0
2 while i < 10:
3     print(f"Current value of i is {i}")
4     i = i + 1
```

```
Current value of i is 0
Current value of i is 1
Current value of i is 2
Current value of i is 3
Current value of i is 4
Current value of i is 5
Current value of i is 6
Current value of i is 7
Current value of i is 8
Current value of i is 9
```

Example2

- Write a Program to Find the Average of n Natural Numbers. Where n Is the Input from the User

```
1  number = int(input("Enter a number up to which you want to find the average"))
2  i = 0
3  sum = 0
4  count = 0
5  while i < number:
6      i = i + 1
7      sum = sum + i
8      count = count + 1
9  average = sum/count
10 print(f"The average of {number} natural numbers is {average}")
```

```
Enter a number up to which you want to find the average5
The average of 5 natural numbers is 3.0
```

Example3

- Write Python Program to Find the Sum of Digits in a Number

In [67]:

```
number = int(input('Enter a number'))
result = 0
remainder = 0
while number != 0:
    remainder = number % 10
    result = result + remainder
    number = int(number / 10)
print(f"The sum of all digits is {result}")
```

```
Enter a number205030
The sum of all digits is 10
```

Example 4

- Write a python program that prints the Fibonacci series (the sum of two elements defines the next) up to n terms where n is provided by the user.

In [70]:

```
1 n=int(input("enter the number: "))
2 previous, current=0, 1
3
4 while previous<=n:
5     print(previous, end=', ')
6     previous, current=current, previous+current
7
8
```

```
enter the number: 10
0, 1, 1, 2, 3, 5, 8,
```

Example

- Write a python program that take a positive number from the user to calculate and print its factorial.
- Write a python program that take two number from the user to calculate the first number to the power of the second number.
- Write a python program that takes twonumbers fro the user to calculate the sum and the product of the numbers between them.

The break Statement

With the break statement we can stop the loop even if the while condition is true:

Example

- Exit the loop when i is 3:

[166]:
```
i = 1
while i < 6:
  print(i)
  if i == 3:
    break
  i += 1
```

```
1
2
3
```

The continue Statement

With the continue statement we can stop the current iteration, and continue with the next:

Example

- Continue to the next iteration if i is 3:

```
[167]: i = 0
       while i < 6:
         i += 1
         if i == 3:
           continue
         print(i)
```

```
1
2
4
5
6
```

The else Statement

With the else statement we can run a block of code once when the condition no longer is true:

Example

- Print a message once the condition is false:

```
[168]: i = 1
       while i < 6:
         print(i)
         i += 1
       else:
         print("i is no longer less than 6")
```

```
1
2
3
4
5
i is no longer less than 6
```

TRAPPED IN AN INFINITE LOOP?

If you ever run a program that has a bug causing it to get stuck in an infinite loop, press CTRL-C or select **Shell ▸ Restart Shell** from IDLE's menu. This will send a `KeyboardInterrupt` error to your program and cause it to stop immediately. Try stopping a program by creating a simple infinite loop in the file editor, and save the program as *infiniteLoop.py*.

```
while True:
    print('Hello, world!')
```

When you run this program, it will print `Hello, world!` to the screen forever because the `while` statement's condition is always `True`. CTRL-C is also handy if you want to simply terminate your program immediately, even if it's not stuck in an infinite loop.

"TRUTHY" AND "FALSEY" VALUES

Conditions will consider some values in other data types equivalent to `True` and `False`. When used in conditions, `0`, `0.0`, and `''` (the empty string) are considered `False`, while all other values are considered `True`. For example, look at the following program:

```
name = ''
❶ while not name:
    print('Enter your name:')
    name = input()
print('How many guests will you have?')
numOfGuests = int(input())
❷ if numOfGuests:
    ❸ print('Be sure to have enough room for all your guests.')
print('Done')
```

You can view the execution of this program at *https://autbor.com/howmanyguests/*. If the user enters a blank string for `name`, then the `while` statement's condition will be `True` ❶, and the program continues to ask for a name. If the value for `numOfGuests` is not `0` ❷, then the condition is considered to be `True`, and the program will print a reminder for the user ❸.

You could have entered `not name != ''` instead of `not name`, and `numOfGuests != 0` instead of `numOfGuests`, but using the truthy and falsey values can make your code easier to read.

for Loop

for Loop

```
for iteration_variable in sequence:
    indented_block
```

- A ***for*** loop is used for iterating over a range of numbers or a sequence (that is either a list, a tuple, a dictionary, a set, or a string).

- With the for loop we can execute a set of statements (indented_block), one for each item in a list, tuple, set etc.

for Loop: Looping through a String

```
for iteration_variable in string:
    indented_block
```

- Even strings are iterable objects, they contain a sequence of characters:

Example

- Loop through the letters in the word "banana":

```
[170]: for x in "banana":
         print(x)

b
a
n
a
n
a
```

for Loop: Looping through range()

```
for iteration_variable in range():
    indented_block
```

- ***range()***: function that generates a sequence of numbers
- It has 3 definitions:

range(start, stop, step)

- generate numbers begining from **start** value (**included**) to the **stop** value(**excluded**) with **step** value =**step**
- range(0, 10, 3)
  - 0, 3, 6, 9
- range(-10, -100, -30)
  - -10, -40, -70

range(start, stop)

- generate numbers from the **start** value (included) to the **stop** value(excluded) with **step** =**1**
- range(5, 10) = range(5, 10, 1)
  - 5, 6, 7, 8, 9

range(stop)

- generate numbers from **zero** to the **stop** value(excluded) with **step** =1
- range (5) = range (0, 5)
  - 0, 1, 2, 3, 4

for Loop: Looping through range() .....

- To iterate over a sequence of numbers, the built-in function range() comes in handy.

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

```
0
1
2
3
4
```

```
for x in range(2, 6):
  print(x)
```

```
2
3
4
5
```

```
for x in range(2, 30, 3):
  print(x)
```

```
2
5
8
11
14
17
20
23
26
29
```

Example 1

- Write a program that add numbers from 0 to 100

```
1  total = 0
2  for num in range(101):
3      total = total + num
4  print(total)
```

```
5050
```

Example 2

- Write a program that prints a string in a reverse order

```
1 fruit=input("enter the fruit name: ")
2 for i in range(-1, -(len(fruit)+1), -1):
3     print(fruit[i], end='')
```

```
enter the fruit name: apple
elppa
```

Example 3

- Write a Program to Find the Sum of All Odd and Even Numbers up to a Number Specified by the User.

```
1  number = int(input("Enter a number"))
2  even = 0
3  odd = 0
4  for i in range(number):
5      if i % 2 == 0:
6          even = even + i
7      else:
8          odd = odd + i
9  print(f"Sum of Even numbers are {even} and Odd numbers are {odd}")
```

```
Enter a number5
Sum of Even numbers are 6 and Odd numbers are 4
```

Example 4

- Write a Program to Find the Factorial of a Number

Note: i.e., n! = n * (n – 1) * (n – 2) * (n – 3)… 3 * 2 * 1

```
number = int(input('Enter a number'))
factorial = 1
if number < 0:
    print("Factorial doesn't exist for negative numbers")
elif number == 0:
    print('The factorial of 0 is 1')
else:
    for i in range(1, number + 1):
        factorial = factorial * i
print(f"The factorial of number {number} is {factorial}")
```

```
Enter a number5
The factorial of number 5 is 120
```

for Loop: break Statement

With the break statement we can stop the loop before it has looped through all the items:

Example

- Exit the loop when x is "banana":

```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
  print(x)
  if x == "banana":
    break
```

```
apple
banana
```

Example

- Exit the loop when x is "banana", but this time the break comes before the print:

```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
  if x == "banana":
    break
  print(x)
```

```
apple
```

for loop: continue statement

With the continue statement we can stop the current iteration of the loop, and continue with the next value of the for loop's counter:

Example

- Do not print banana:

```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
  if x == "banana":
    continue
  print(x)
```

```
apple
cherry
```

Example 1

- Write a Program to Check Whether a Number Is Prime or Not

```
1 number = int(input('Enter a number > 1: '))
2 for i in range(2, number):
3     if number % i == 0:
4         break
5 else: print(number, "is a prime number")
```

```
Enter a number > 1: 5
5 is a prime number
```

Example 2:

- Write a program trhat check whether a string is a plandrome or not

```
1  txt=input("enter your text: ")
2  txt_size=len(txt)
3  for i in range(int(txt_size/2)):
4      if txt[i]!=txt[txt_size-i-1]: break
5  else:
6      print("your text: ", txt, "is palandrom")
```

```
enter your text: madam
your text:  madam is palandrom
```

A Short Program: Guess the Number

The examples I've shown you so far are useful for introducing basic concepts, but now let's see how everything you've learned comes together in a more complete program. In this section, I'll show you a simple "guess the number" game. When you run this program, the output will look something like this:

```
I am thinking of a number between 1 and 20.

Take a guess.

10

Your guess is too low.

Take a guess.

15

Your guess is too low.

Take a guess.

17

Your guess is too high.

Take a guess.

16

Good job! You guessed my number in 4 guesses!
```

Solution

```
# This is a guess the number game.

import random

secretNumber = random.randint(1, 20)
```

First, a comment at the top of the code explains what the program does. Then, the program imports the `random` module so that it can use the `random.randint()` function to generate a number for the user to guess. The return value, a random integer between 1 and 20, is stored in the variable `secretNumber`.

Solution ...

```
print('I am thinking of a number between 1 and 20.')

# Ask the player to guess 6 times.

for guessesTaken in range(1, 7):

    print('Take a guess.')

    guess = int(input())
```

The program tells the player that it has come up with a secret number and will give the player six chances to guess it. The code that lets the player enter a guess and checks that guess is in a `for` loop that will loop at most six times. The first thing that happens in the loop is that the player types in a guess. Since `input()` returns a string, its return value is passed straight into `int()`, which translates the string into an integer value. This gets stored in a variable named `guess`.

Solution ...

```
    if guess < secretNumber:
        print('Your guess is too low.')
    elif guess > secretNumber:
        print('Your guess is too high.')
```

These few lines of code check to see whether the guess is less than or greater than the secret number. In either case, a hint is printed to the screen.

```
    else:
        break    # This condition is the correct guess!
```

If the guess is neither higher nor lower than the secret number, then it must be equal to the secret number—in which case, you want the program execution to break out of the `for` loop.

Solution ...

```
if guess == secretNumber:
    print('Good job! You guessed my number in ' + str(guessesTaken) + ' guesses!')
else:
    print('Nope. The number I was thinking of was ' + str(secretNumber))
```

After the `for` loop, the previous `if...else` statement checks whether the player has correctly guessed the number and then prints an appropriate message to the screen. In both cases, the program displays a variable that contains an integer value (`guessesTaken` and `secretNumber`). Since it must concatenate these integer values to strings, it passes these variables to the `str()` function, which returns the string value form of these integers. Now these strings can be concatenated with the `+` operators before finally being passed to the `print()` function call.

Complete Program

```
# This is a guess the number game.
import random
secretNumber = random.randint(1, 20)
print('I am thinking of a number between 1 and 20.')

# Ask the player to guess 6 times.
for guessesTaken in range(1, 7):
    print('Take a guess.')
    guess = int(input())

    if guess < secretNumber:
        print('Your guess is too low.')
    elif guess > secretNumber:
        print('Your guess is too high.')
    else:
        break    # This condition is the correct guess!

if guess == secretNumber:
    print('Good job! You guessed my number in ' + str(guessesTaken) + 'guesses!')
else:
    print('Nope. The number I was thinking of was ' + str(secretNumber))
```

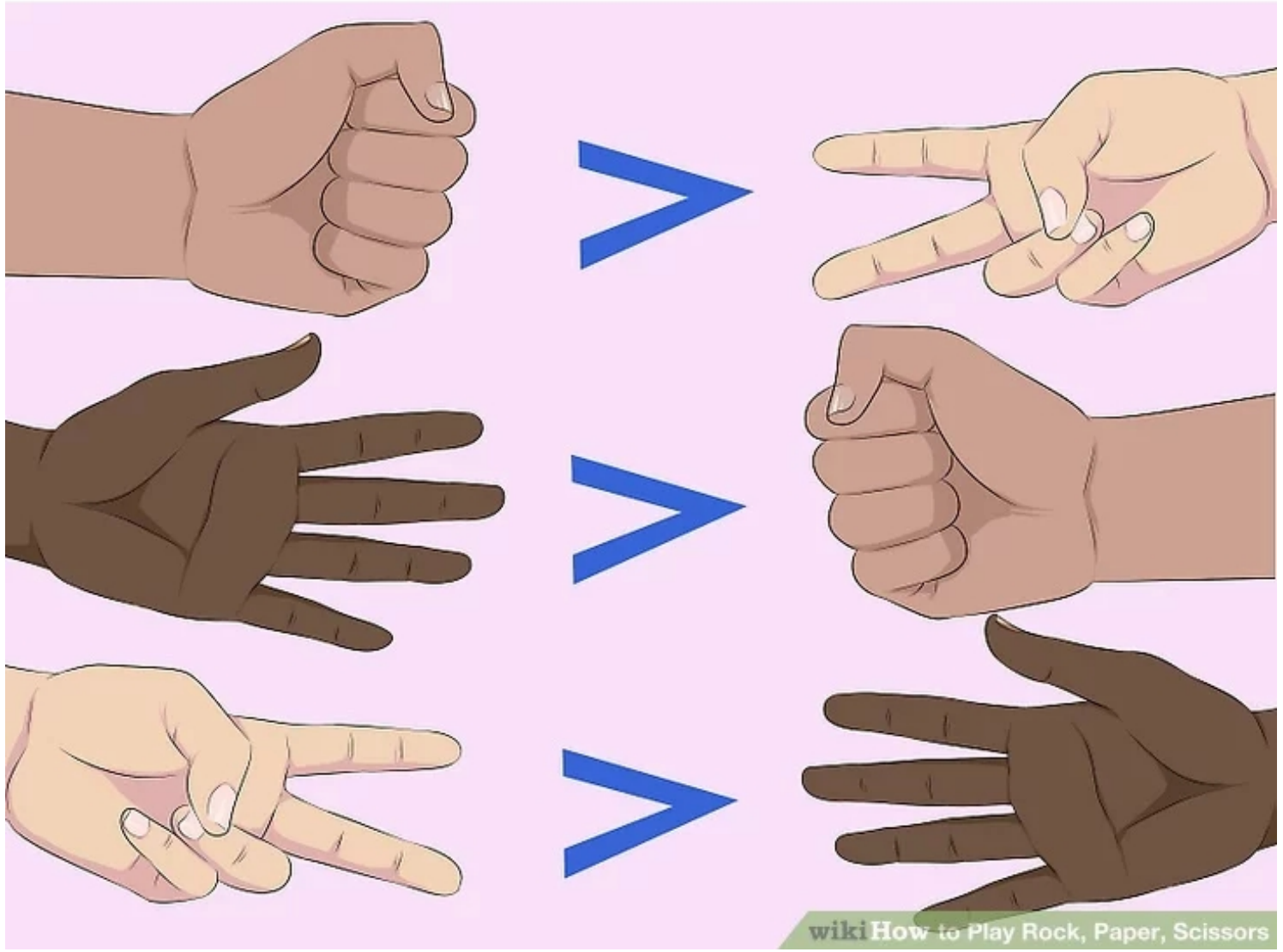
```
I am thinking of a number between 1 and 20.
Take a guess.
17
Good job! You guessed my number in 1guesses!
```

A Short Program: Rock, Paper, Scissors

Let's use the programming concepts we've learned so far to create a simple rock, paper, scissors game. The output will look like this:

```
ROCK, PAPER, SCISSORS

0 Wins, 0 Losses, 0 Ties

Enter your move: (r)ock (p)aper (s)cissors or (q)uit

p

PAPER versus...

PAPER

It is a tie!

0 Wins, 1 Losses, 1 Ties

Enter your move: (r)ock (p)aper (s)cissors or (q)uit

s

SCISSORS versus...

PAPER

You win!

1 Wins, 1 Losses, 1 Ties

Enter your move: (r)ock (p)aper (s)cissors or (q)uit

q
```

Win **Lose**



Solution

```
import random, sys

print('ROCK, PAPER, SCISSORS')

# These variables keep track of the number of wins, losses, and ties.

wins = 0

losses = 0

ties = 0
```

First, we import the `random` and `sys` module so that our program can call the `random.randint()` and `sys.exit()` functions. We also set up three variables to keep track of how many wins, losses, and ties the player has had.

Solution ...

```
while True: # The main game loop.
    print('%s Wins, %s Losses, %s Ties' % (wins, losses, ties))
    while True: # The player input loop.
        print('Enter your move: (r)ock (p)aper (s)cissors or (q)uit')
        playerMove = input()
        if playerMove == 'q':
            sys.exit() # Quit the program.
        if playerMove == 'r' or playerMove == 'p' or playerMove == 's':
            break # Break out of the player input loop.
        print('Type one of r, p, s, or q.')
```

This program uses a `while` loop inside of another `while` loop. The first loop is the main game loop, and a single game of rock, paper, scissors is played on each iteration through this loop. The second loop asks for input from the player, and keeps looping until the player has entered an `r`, `p`, `s`, or `q` for their move. The `r`, `p`, and `s` correspond to rock, paper, and scissors, respectively, while the `q` means the player intends to quit. In that case, `sys.exit()` is called and the program exits. If the player has entered `r`, `p`, or `s`, the execution breaks out of the loop. Otherwise, the program reminds the player to enter `r`, `p`, `s`, or `q` and goes back to the start of the loop.

Solution ...

```
# Display what the player chose:
if playerMove == 'r':
    print('ROCK versus...')
elif playerMove == 'p':
    print('PAPER versus...')
elif playerMove == 's':
    print('SCISSORS versus...')
```

The player’s move is displayed on the screen.

Solution ...

```
    # Display what the computer chose:
    randomNumber = random.randint(1, 3)
    if randomNumber == 1:
        computerMove = 'r'
        print('ROCK')
    elif randomNumber == 2:
        computerMove = 'p'
        print('PAPER')
    elif randomNumber == 3:
        computerMove = 's'
        print('SCISSORS')
```

Next, the computer's move is randomly selected. Since `random.randint()` can only return a random number, the `1`, `2`, or `3` integer value it returns is stored in a variable named `randomNumber`. The program stores a `'r'`, `'p'`, or `'s'` string in `computerMove` based on the integer in `randomNumber`, as well as displays the computer's move.

Solution ...

```
    # Display and record the win/loss/tie:
    if playerMove == computerMove:
        print('It is a tie!')
        ties = ties + 1
    elif playerMove == 'r' and computerMove == 's':
        print('You win!')
        wins = wins + 1
    elif playerMove == 'p' and computerMove == 'r':
        print('You win!')
        wins = wins + 1
    elif playerMove == 's' and computerMove == 'p':
        print('You win!')
        wins = wins + 1
    elif playerMove == 'r' and computerMove == 'p':
        print('You lose!')
        losses = losses + 1
    elif playerMove == 'p' and computerMove == 's':
        print('You lose!')
        losses = losses + 1
    elif playerMove == 's' and computerMove == 'r':
        print('You lose!')
        losses = losses + 1
```

Finally, the program compares the strings in `playerMove` and `computerMove`, and displays the results on the screen. It also increments the `wins`, `losses`, or `ties` variable appropriately. Once the execution reaches the end, it jumps back to the start of the main program loop to begin another game.

Complete Program

```
ROCK, PAPER, SCISSORS
0 Wins, 0 Losses, 0 Ties
Enter your move: (r)ock (p)aper (s)cissors or (q)uit
r
ROCK versus...
PAPER
You lose!
0 Wins, 1 Losses, 0 Ties
Enter your move: (r)ock (p)aper (s)cissors or (q)uit
w
Type one of r, p, s, or q.
Enter your move: (r)ock (p)aper (s)cissors or (q)uit
p
PAPER versus...
PAPER
It is a tie!
0 Wins, 1 Losses, 1 Ties
Enter your move: (r)ock (p)aper (s)cissors or (q)uit
q
```

```python
import random, sys

print('ROCK, PAPER, SCISSORS')

# These variables keep track of the number of wins, losses, and ties.
wins = 0
losses = 0
ties = 0

while True: # The main game loop.
    print('%s Wins, %s Losses, %s Ties' % (wins, losses, ties))
    while True: # The player input loop.
        print('Enter your move: (r)ock (p)aper (s)cissors or (q)uit')
        playerMove = input()
        if playerMove == 'q':
            sys.exit() # Quit the program.
        if playerMove == 'r' or playerMove == 'p' or playerMove == 's':
            break # Break out of the player input loop.
        print('Type one of r, p, s, or q.')

    # Display what the player chose:
    if playerMove == 'r':
        print('ROCK versus...')
    elif playerMove == 'p':
        print('PAPER versus...')
    elif playerMove == 's':
        print('SCISSORS versus...')

    # Display what the computer chose:
    randomNumber = random.randint(1, 3)
    if randomNumber == 1:
        computerMove = 'r'
        print('ROCK')
    elif randomNumber == 2:
        computerMove = 'p'
        print('PAPER')
    elif randomNumber == 3:
        computerMove = 's'
        print('SCISSORS')

    # Display and record the win/loss/tie:
    if playerMove == computerMove:
        print('It is a tie!')
        ties = ties + 1
    elif playerMove == 'r' and computerMove == 's':
        print('You win!')
        wins = wins + 1
    elif playerMove == 'p' and computerMove == 'r':
        print('You win!')
        wins = wins + 1
    elif playerMove == 's' and computerMove == 'p':
        print('You win!')
        wins = wins + 1
    elif playerMove == 'r' and computerMove == 'p':
        print('You lose!')
        losses = losses + 1
    elif playerMove == 'p' and computerMove == 's':
        print('You lose!')
        losses = losses + 1
    elif playerMove == 's' and computerMove == 'r':
        print('You lose!')
        losses = losses + 1
```

A Short Program: Zigzag

Let's use the programming concepts you've learned so far to create a small animation program. This program will create a back-and-forth, zigzag pattern until the user stops it by pressing the Mu editor's Stop button or by pressing CTRL-C. When you run this program, the output will look something like this:

```
    ********
   ********
  ********
 ********
********
 ********
  ********
   ********
    ********
```

Solution

```
import time, sys

indent = 0 # How many spaces to indent.

indentIncreasing = True # Whether the indentation is increasing or not.
```

First, we'll import the `time` and `sys` modules. Our program uses two variables: the `indent` variable keeps track of how many spaces of indentation are before the band of eight asterisks and `indentIncreasing` contains a Boolean value to determine if the amount of indentation is increasing or decreasing.

Solution ....

```
try:
    while True: # The main program loop.
        print(' ' * indent, end='')
        print('********')
        time.sleep(0.1) # Pause for 1/10 of a second.
```

Next, we place the rest of the program inside a try statement. When the user presses CTRL-C while a Python program is running, Python raises the `KeyboardInterrupt` exception. If there is no `try-except` statement to catch this exception, the program crashes with an ugly error message. However, for our program, we want it to cleanly handle the `KeyboardInterrupt` exception by calling `sys.exit()`. (The code for this is in the `except` statement at the end of the program.)

The `while True:` infinite loop will repeat the instructions in our program forever. This involves using `' ' * indent` to print the correct amount of spaces of indentation. We don't want to automatically print a newline after these spaces, so we also pass `end=''` to the first `print()` call. A second `print()` call prints the band of asterisks. The `time.sleep()` function hasn't been covered yet, but suffice it to say that it introduces a one-tenth-second pause in our program at this point.

Solution ....

```
        if indentIncreasing:
            # Increase the number of spaces:
            indent = indent + 1
            if indent == 20:
                indentIncreasing = False # Change direction.
```

Next, we want to adjust the amount of indentation for the next time we print asterisks. If `indentIncreasing` is `True`, then we want to add one to `indent`. But once indent reaches `20`, we want the indentation to decrease.

Solution ....

```
        else:
            # Decrease the number of spaces:
            indent = indent - 1
            if indent == 0:
                indentIncreasing = True # Change direction.
```

Meanwhile, if `indentIncreasing` was `False`, we want to subtract one from `indent`. Once indent reaches `0`, we want the indentation to increase once again. Either way, the program execution will jump back to the start of the main program loop to print the asterisks again.

Solution ....

```
except KeyboardInterrupt:
    sys.exit()
```

If the user presses `CTRL-C` at any point that the program execution is in the `try` block, the `KeyboardInterrrupt` exception is raised and handled by this `except` statement. The program execution moves inside the `except` block, which runs `sys.exit()` and quits the program. This way, even though the main program loop is an infinite loop, the user has a way to shut down the program.

```python
import time, sys
indent = 0 # How many spaces to indent.
indentIncreasing = True # Whether the indentation is increasing or not.

try:
    while True: # The main program loop.
        print(' ' * indent, end='')
        print('********')
        time.sleep(0.1) # Pause for 1/10 of a second.

        if indentIncreasing:
            # Increase the number of spaces:
            indent = indent + 1
            if indent == 20:
                # Change direction:
                indentIncreasing = False

        else:
            # Decrease the number of spaces:
            indent = indent - 1
            if indent == 0:
                # Change direction:
                indentIncreasing = True
except KeyboardInterrupt:
    try: sys.exit()
    except: print("Zigzag terminates")
```