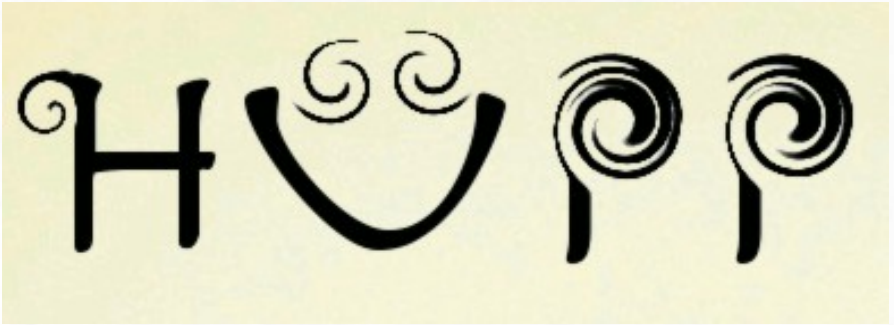


HUPP-Python

3rd Meeting

Book: *Invent with Pyhton, Albert Sweigart*

Chapter – 4 : Guess the Number

Chapter – 5 : Jokes

Chapter 4: GUESS THE NUMBER

Outline: Topics Covered In This Chapter:

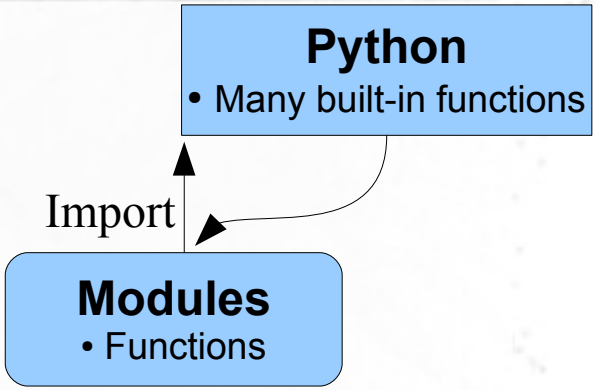
- import statements
- Modules
- Arguments
- while statements
- Conditions
- Blocks
- Booleans
- Comparison operators
- The difference between = and ==.
- if statements
- The break keyword.
- The str() and int() functions.
- The random.randint() function.

Sample Run of "Guess the Number"

```
Hello! What is your name?
Albert
Well, Albert, I am thinking of a number between 1 and 20.
Take a guess.
10
Your guess is too high.
Take a guess.
2
Your guess is too low.
Take a guess.
4
Good job, Albert! You guessed my number in 3 guesses!
```

Import Statement

Statements are not functions! No parentheses after their name..



- They are **instructions** that perform some action but do not evaluate to a value like expressions do
- Some functions exist in separate programs called **modules**

import + module name

- Import statement → import keyword + module name
  - *import random* → random is a module

```
>>> randint(1,20)
Traceback (most recent call last):
  File "<pyshell#49>", line 1, in <module>
    randint(1,20)
NameError: name 'randint' is not defined
>>>
```

The **random.randint()** function

Function calls are **expressions** because they evaluate to a value.

- The **randint() function** will return a random integer between (and including) the two integers we give it.

random.randint(int1, int2)

Example 1: Let's try to write a code which generates random numbers :)

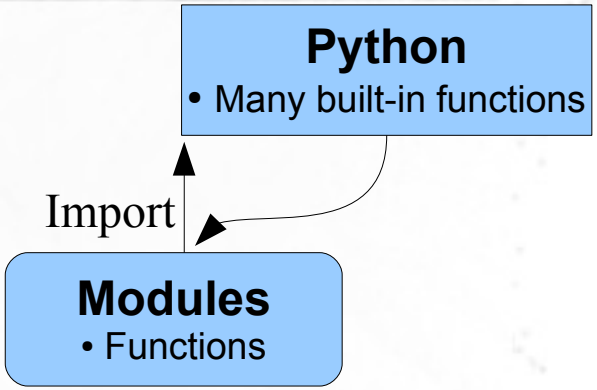
Import Statement

Statements are not functions! No parentheses after their name..

- They are **instructions** that perform some action but do not evaluate to a value like expressions do
- Some functions exist in separate programs called **modules**

import + module name

- Import statement → import keyword + module name
  - *import random* → random is a module




```
>>> randint(1,20)
Traceback (most recent call last):
  File "<pyshell#49>", line 1, in <module>
    randint(1,20)
NameError: name 'randint' is not defined
>>>
```

The **random.randint()** function

Function calls are **expressions** because they evaluate to a value.

- The **randint() function** will return a random integer between (and including) the two integers we give it.

random.randint(int1, int2)

Example 1: Let's try to write a code which generates random numbers :)

```
1. # This is a guess the number game.
2. import random
3.
4. guessesTaken = 0
5.
6. print('Hello! What is your name?')
7. myName = input()
8.
9. number = random.randint(1, 20)
10. print('Well, ' + myName + ', I am thinking of a number between 1 and
20.')
11.
```

number is the function call's return value

while statements (loop)

while + **condition** :

Always **a colon** (the : sign) after the condition.

- *while guessesTaken < 3:*

- **Conditions:** An expression that combines two values with a comparison operator (such as < or >) and **always** evaluates to a Boolean value, either True or False
- **Booleans** : The Boolean data type has only two values: *True* or *False*

```
>>> 0 <6
True
>>> 6<0
False
>>> 50 < 10
False
>>> 10 < 11
True
>>> 10 < 10
False
```

```
>>> 10 == 10
True
>>> 10 == 11
False
>>> 11 == 10
False
>>> 10 != 10
False
>>> 10 != 11
True
```

```
>>> 'Hello' == 'Hello'
True
>>> 'Hello' == 'Good bye'
False
>>> 'Hello' == 'HELLO'
False
>>> 'Good bye' != 'Hello'
True
```

```
>>> 42 == 'Hello'
False
>>> 42 != '42'
True
```

- **Comparison Operators** : The sign " < " .

The comparison operator is used to compare two values and evaluate to a True or False Boolean value.

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

- **Blocks** : One or more lines of code grouped together with the **same amount of indentation**

```
>>> while guessesTaken < 3:
        print("tahmin et bakalim")
        s = input()
    print()
```

while statements (loop)

while + **condition** :

- while guessesTaken < 3:

Always **a colon** (the : sign) after the condition.

Example 2: Try to write a code with a while loop ;)

```
Hello! What is your name?
Albert
Well, Albert, I am thinking of a number between 1 and 20.
Take a guess.
10
Your guess is too high.
Take a guess.
2
Your guess is too low.
Take a guess.
4
Good job, Albert! You guessed my number in 3 guesses!
```

while statements (loop)

while + **condition** :

- while guessesTaken < 3:

Always **a colon** (the : sign) after the condition.

Example 2: Try to write a code with a while loop ;)

```
Hello! What is your name?
Albert
Well, Albert, I am thinking of a number between 1 and 20.
Take a guess.
10
Your guess is too high.
Take a guess.
2
Your guess is too low.
Take a guess.
4
Good job, Albert! You guessed my number in 3 guesses!
```

```
12. while guessesTaken < 6:
13. ····print('Take a guess.')
14. ····guess = input()
15. ····guess = int(guess)
16.
17. ····guessesTaken = guessesTaken + 1
18.
19. ····if guess < number:
20. ········print('Your guess is too low.')
21.
22. ····if guess > number:
23. ········print('Your guess is too high.')
```

1 2 3

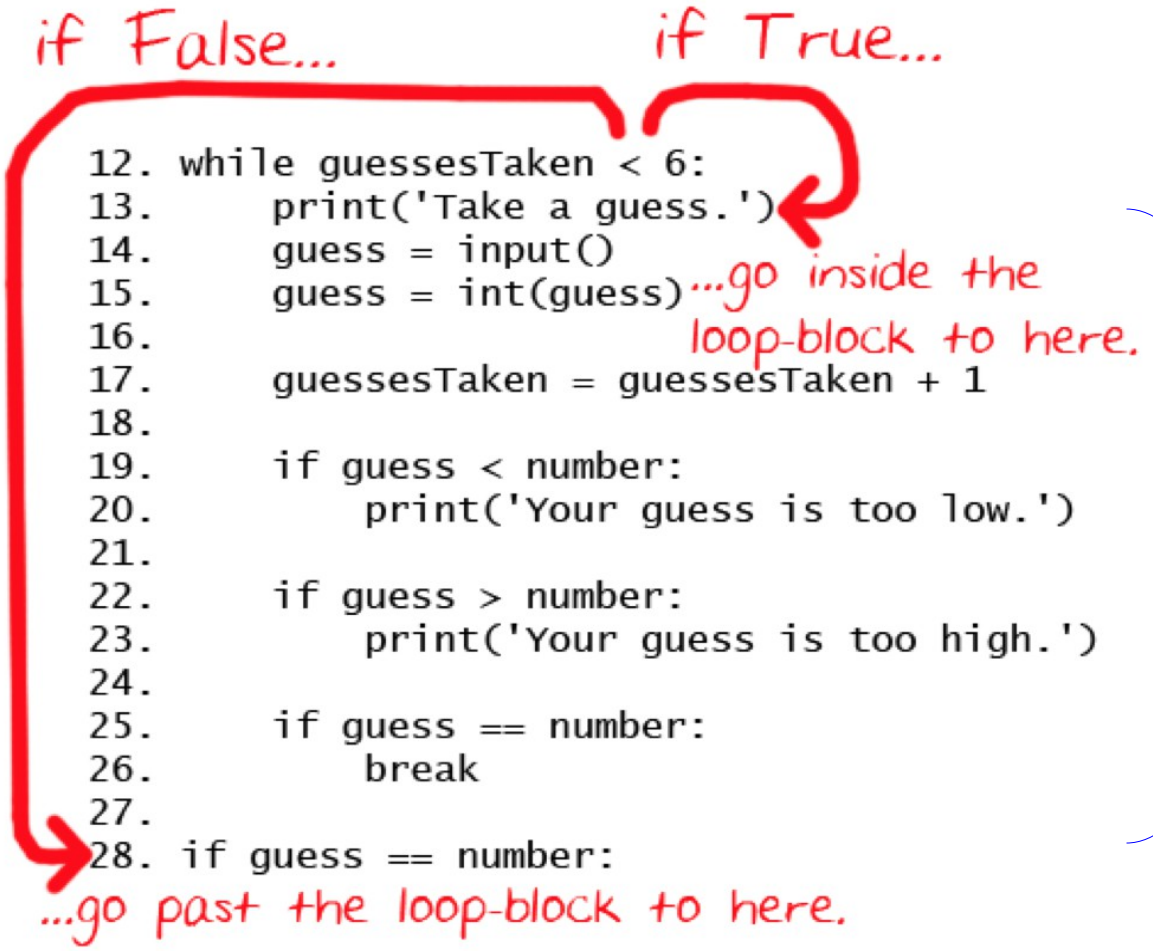
** The block ends when there is a line of code with the same indentation before the block started

** A block can contain just one line(blocks 2 & 3)

while statements (loop)

while + condition :

Always **a colon** (the : sign) after the condition.




```
12. while guessesTaken < 6:
13.     print('Take a guess.')
14.     guess = input()
15.     guess = int(guess)
16.
17.     guessesTaken = guessesTaken + 1
18.
19.     if guess < number:
20.         print('Your guess is too low.')
21.
22.     if guess > number:
23.         print('Your guess is too high.')
24.
25.     if guess == number:
26.         break
27.
28. if guess == number:
```

Ask a guess and
store it in a variable called guess
AND **convert** string to integer

Increase the number of guesses

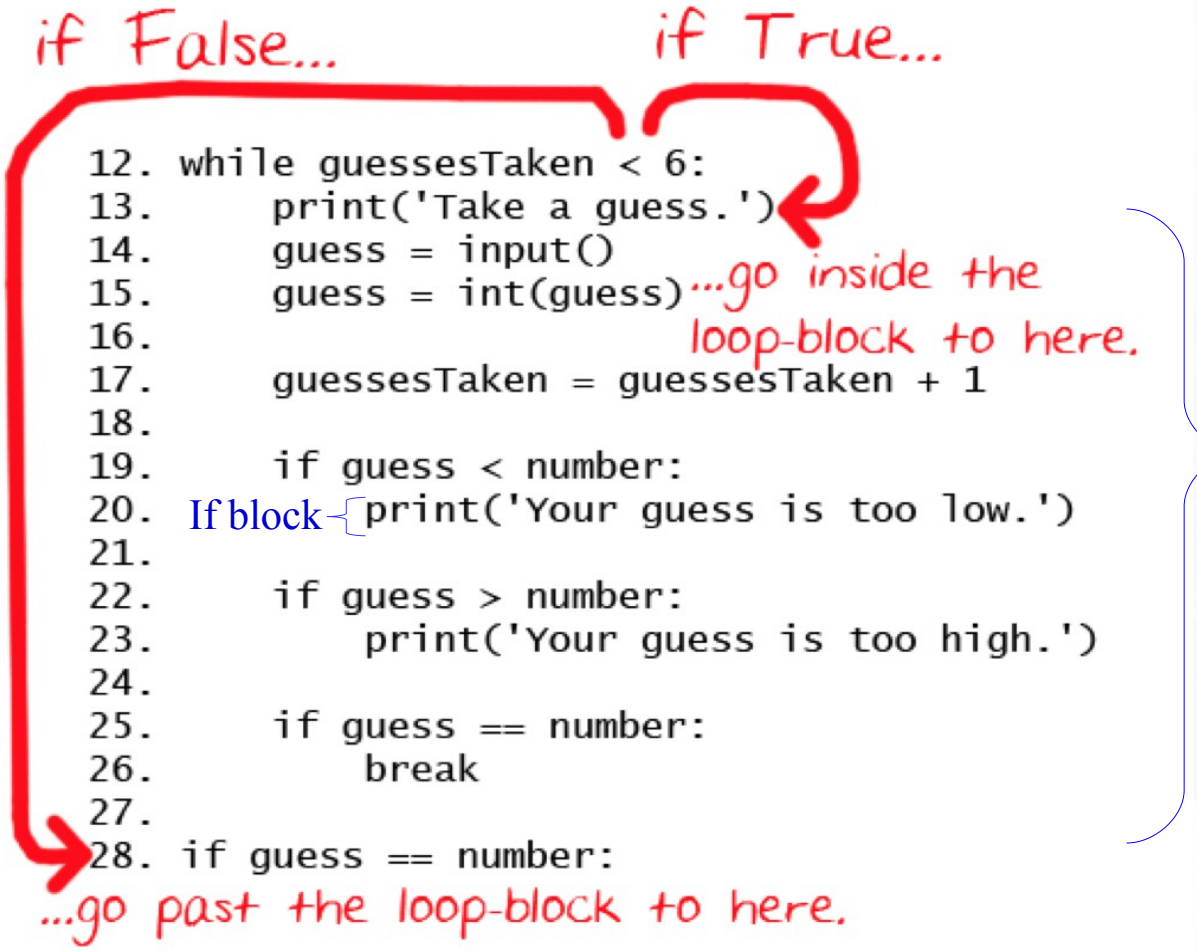
while block

- compare the player's guess with the random number the computer came up with which is integer. So string should be converted to integer!

while statements (loop)

while + condition :

Always **a colon** (the : sign) after the condition.




```
12. while guessesTaken < 6:
13.     print('Take a guess.')
14.     guess = input()
15.     guess = int(guess)
16.
17.     guessesTaken = guessesTaken + 1
18.
19.     if guess < number:
20.         print('Your guess is too low.')
21.
22.     if guess > number:
23.         print('Your guess is too high.')
24.
25.     if guess == number:
26.         break
27.
28. if guess == number:
```

Ask a guess and
store it in a variable called guess
AND **convert** string to integer

Increase the number of guesses

while block

if statements(no loop)

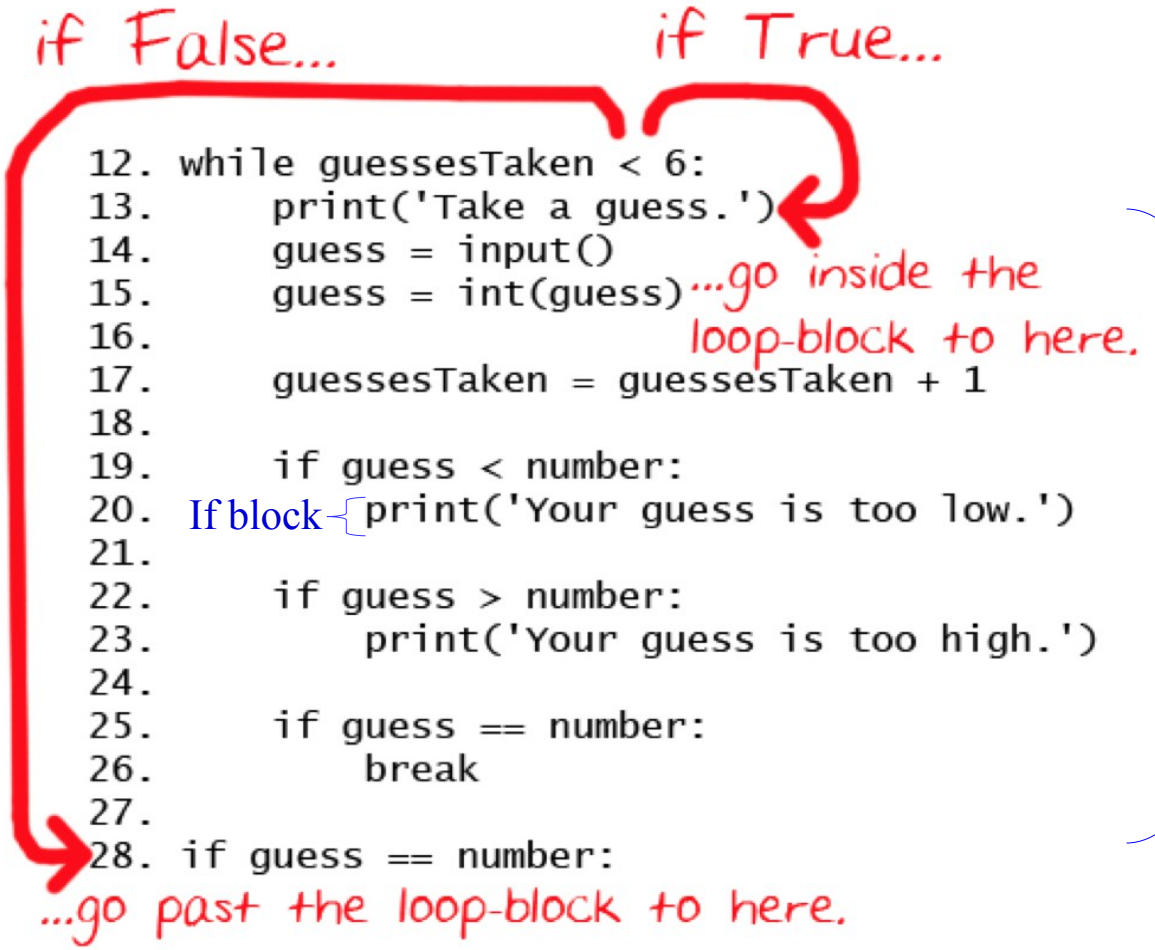
if + condition :

- compare the player's guess with the random number the computer came up with which is integer. So string should be converted to integer!

while statements (loop)

while + condition :

Always **a colon** (the : sign) after the condition.



Ask a guess and
store it in a variable called guess
AND **convert** string to integer

Increase the number of guesses

while block

if statements(no loop)

if + condition :

break statement (no condition)

jump out of the while-block to the first line after the end of the while-block.

- compare the player's guess with the random number the computer came up with which is integer. So string should be converted to integer!

int( ) Function

int( ) function

- The int() function takes one argument.
- input() function returns a string of text
  - If the player enters 5 as their guess,
  - the input() function will return the string value '5' and not the integer value 5

```
>>> int('42')
42
>>> int(42)
42
>>> int('hello')
Traceback (most recent call last):
  File "<pyshell#4>", line 1, in <module>
int('hello')
ValueError: invalid literal for int() with base 10: 'hello'
>>> int('forty-two')
Traceback (most recent call last):
  File "<pyshell#5>", line 1, in <module>
int('forty-two')
ValueError: invalid literal for int() with base 10: 'forty-two'
>>> int(' 42 ')
42
>>> 3 + int('2')
5
```

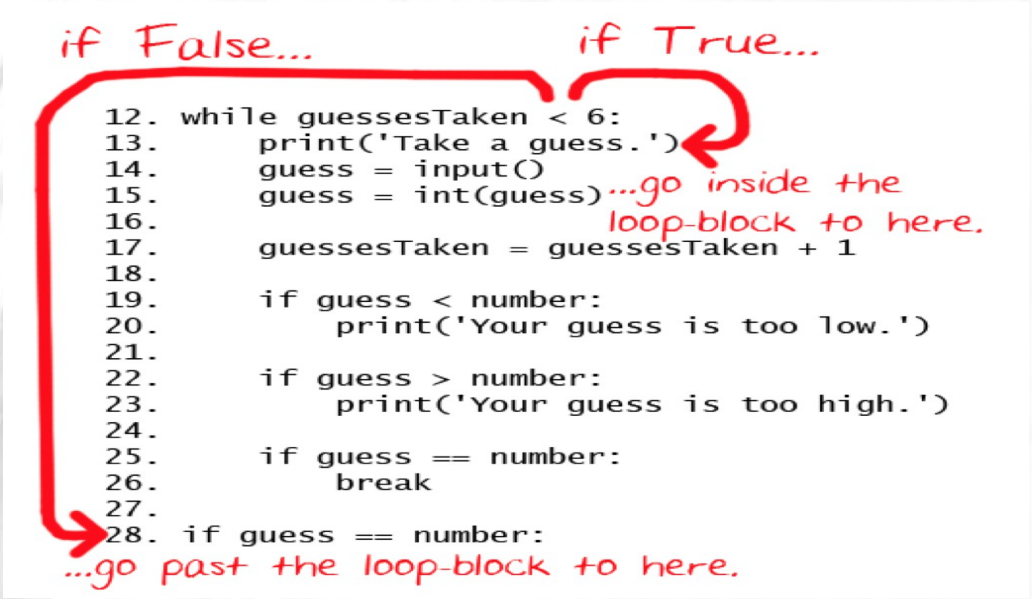
The string we pass to int() must be made up of **numbers**

The integer we pass to int() must also be **numerical**

Allowed to leave **spaces**

Integer can be **added** to a **string** which has been **converted to an integer**

str( ) Function




```
12. while guessesTaken < 6:
13.     print('Take a guess.')
14.     guess = input()
15.     guess = int(guess)
16.
17.     guessesTaken = guessesTaken + 1
18.
19.     if guess < number:
20.         print('Your guess is too low.')
21.
22.     if guess > number:
23.         print('Your guess is too high.')
24.
25.     if guess == number:
26.         break
27.
28. if guess == number:
```

```
28. if guess == number:
29.     guessesTaken = str(guessesTaken)
30.     print('Good job, ' + myName + '! You guessed my number in ' +
guessesTaken + ' guesses!')
31.
32. if guess != number:
33.     number = str(number)
34.     print('Nope. The number I was thinking of was ' + number)
```

If it is true!

check to see if the player guessed correctly

`str() function` which returns the string form of an argument.

- Convert integer to string! Because, strings can be added only to the strings, not to integers..
  - **str( ) function**

Summary

import statement

variable

while statement

if statement

break statement

```
 1. # This is a guess the number game.
 2. import random
 3.
 4. guessesTaken = 0
 5.
 6. print('Hello! What is your name?')
 7. myName = input()
 8.
 9. number = random.randint(1, 20)
10. print('Well, ' + myName + ', I am thinking of a number between 1 and
20.')
11.
12. while guessesTaken < 6:
13.     print('Take a guess.') # There are four spaces in front of print.
14.     guess = input()
15.     guess = int(guess)
16.
17.     guessesTaken = guessesTaken + 1
18.
19.     if guess < number:
20.         print('Your guess is too low.') # There are eight spaces in
front of print.
21.
22.     if guess > number:
23.         print('Your guess is too high.')
24.
25.     if guess == number:
26.         break
27.
28. if guess == number:
29.     guessesTaken = str(guessesTaken)
30.     print('Good job, ' + myName + '! You guessed my number in ' +
guessesTaken + ' guesses!')
31.
32. if guess != number:
33.     number = str(number)
34.     print('Nope. The number I was thinking of was ' + number)
```

Comment line

random module

randint() function

int() function

str() function

Chapter 5: JOKES

*Outline:*Topics Covered In This Chapter:

- Using print()'s end keyword argument to skip newlines.
- Escape characters.
- Using single quotes and double quotes for strings.

Sample Run of Jokes

```
Frostbite!
What do dentists call an astronaut's cavity?
A black hole!
Knock knock.
Who's there?
Interrupting cow.
Interrupting cow wh-MOO!
```

Source Code:

```
1. print('What do you get when you cross a snowman with a vampire?')
2. input()
3. print('Frostbite!')
4. print()
5. print('What do dentists call a astronaut's cavity?')
6. input()
7. print('A black hole!')
8. print()
9. print('Knock knock.')
10. input()
11. print("Who's there?")
12. input()
13. print('Interrupting cow.')
14. input()
15. print('Interrupting cow wh', end='')
16. print('-MOO!')
```

How the Code Works

```
1. print('What do you get when you cross a snowman with a vampire?')
2. input()
3. print('Frostbite!')
4. print()
```

- Three **print( )** function calls:
  - Because don't want to tell the player second print string(what the joke's punch line is) immediately!
  - So, use input() function after 1st print() function
  - User can type a string or just hit enter after reading
  - Since input() value is not stored, the program will forget about it
  - And move to the next line
  - 3rd print() function has no string argument. It will print a blank line

Escape Characters

```
5.print('What do dentists call a astronaut\'s cavity?')
6.input()
7.print('A black hole!')
8.print()
```

- In the 1st print() function, escape character, \' , to write single quote(') properly

An escape character helps us print out letters that are hard to enter into the source code.

- Without backslash, the Python interpreter would think that this quote meant the end of the string.

```
>>> print('He flew away in a green\teal helocopter.')
He flew away in a green	eal helocopter.
>>>
```

Escape Character	What Is Actually Printed
\\	Backslash (\)
\'	Single quote (')
\"	Double quote (")
\n	Newline
\t	Tab

Quotes and Double Quotes

```
>>> print('Hello world')
Hello world
>>> print("Hello world")
Hello world
```

```
>>> print('Hello world")
SyntaxError: EOL while scanning single-quoted string
>>>
```

- Both single quotes and double quotes can be used, but don't mix them

```
>>> print('I asked to borrow Abe\'s car for a week. He said, "Sure."')
I asked to borrow Abe's car for a week. He said, "Sure."
>>> print("He said, \"I can't believe you let him borrow your car.\"")
He said, "I can't believe you let him borrow your car."
```

- The Python interpreter is smart enough to know that if a string starts with one type of quote, the other type of quote doesn't mean the string is ending

The end Keyword Argument

```
 9. print('Knock knock.')
10. input()
11. print("Who's there?")
12. input()
13. print('Interrupting cow.')
14. input()
15. print('Interrupting cow wh', end='')
16. print('-MOO!')
```

In the 4st print() function, end keyword is used to avoid to have a blank after the print() function

- print() adds a newline character to the end of the string it prints
- It has a second parameter named "end"
- **The blank string** we are passing is called a **keyword argument**
- The end parameter has a specific name, and to pass an argument to this specific parameter we need to use the end= syntax.
  - Type **the keyword + the keyword argument**, you use only one = sign.
  - It is end =' ', and not end =' '
  - It tells the print() function to not add a newline at the end of the string, but instead add a blank string

Summary

```
Frostbite!
What do dentists call an astronaut's cavity?
A black hole!
Knock knock.
Who's there?
Interrupting cow.
Interrupting cow wh-MOO!
```

Source Code:

non-storing input

print a blank line

Single quote

Double quote

```
1. print('What do you get when you cross a snowman with a vampire?')
2. input()
3. print('Frostbite!')
4. print()
5. print('What do dentists call a astronaut\'s cavity?')
6. input()
7. print('A black hole!')
8. print()
9. print('Knock knock.')
10. input()
11. print("Who's there?")
12. input()
13. print('Interrupting cow.')
14. input()
15. print('Interrupting cow wh', end='')
16. print('-MOO!')
```

\ escape character

end='' keyword