

Internet of Things 420-420-LE

Week 2: Introduction to Python

CHAMPLAIN COLLEGE

Rev.: Jan. 15th, 2025

© Champlain College, 2025, By: Gabriel Astudillo

The Python interpreter

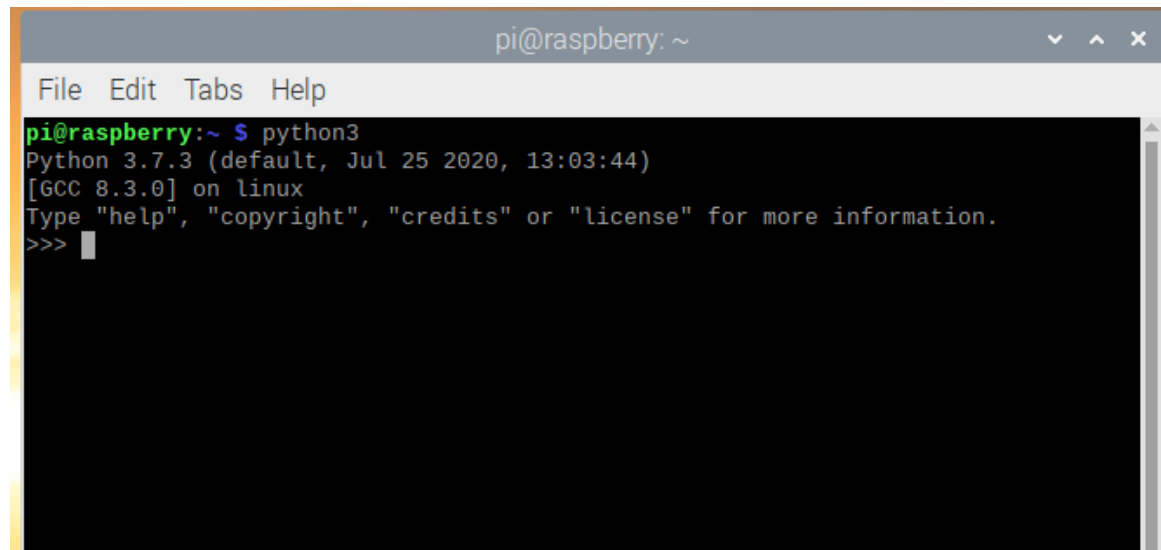
- Python is an interpreted programming language, instead of a compiled one.
- A compiled programming language has all its program's language statements turned into binary code at once, before it can be executed.
- With an interpreted programming language, each of its programming statement, one at a time, is checked for syntax errors, translated into binary code, and then executed.

The Python interpreter

- Different tools that let you enter Python code:
 - **The interactive shell** enables you to enter a single Python statement and have immediately checked for errors and executed.
 - **The Development environment**, this tool provides many features to assist in the development of Python programs. It has an interactive shell where each Python statement is interpreted as it is entered. It also contains a text editor where entire Python programs, called scripts, can be developed.
 - **Text editors**, it's a program that allows you to create and modify regular text files.

Learning about the Python Interactive Shell

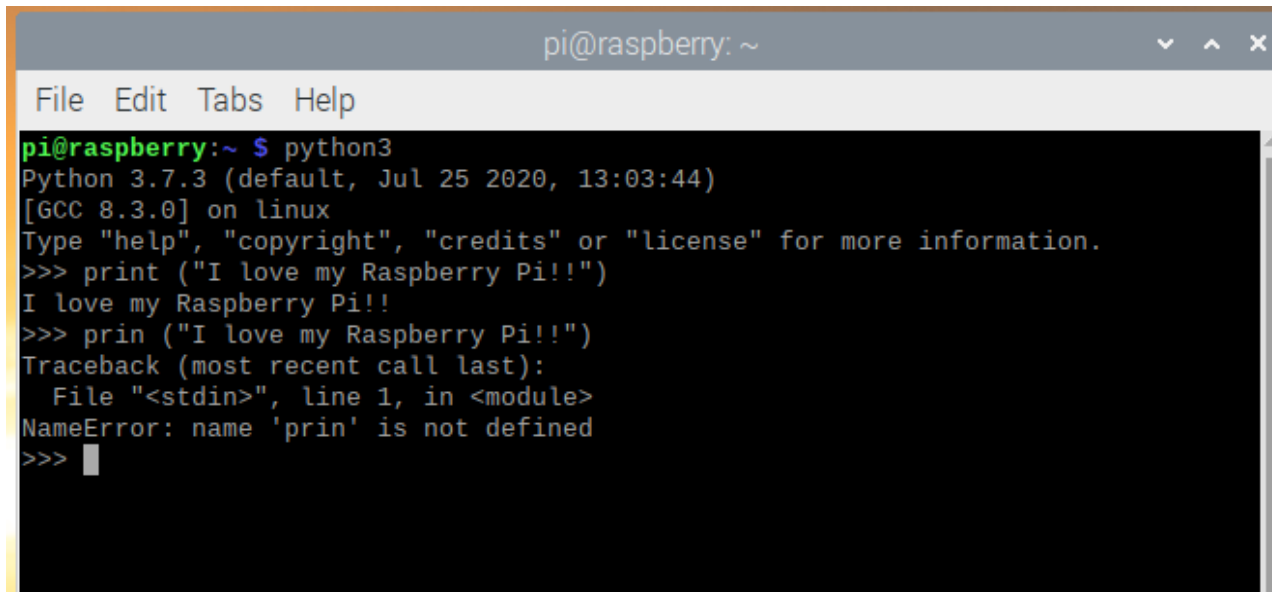
- The Python interactive shell is primarily used to try Python statements and check syntax.
- To enter the interactive Python shell, type in the command `python` or `python3` in a terminal and press enter.



```
pi@raspberrypi: ~  
File Edit Tabs Help  
pi@raspberrypi:~ $ python3  
Python 3.7.3 (default, Jul 25 2020, 13:03:44)  
[GCC 8.3.0] on linux  
Type "help", "copyright", "credits" or "license" for more information.  
>>> 
```

Learning about the Python Interactive Shell

- At this point you can enter a Python statement and press Enter to have the shell interpret it.
- The Python interpreter checks the statement's syntax. If the syntax is correct, the statement is translated into binary code and executed.



```
pi@raspberrypi: ~  
File Edit Tabs Help  
pi@raspberrypi:~ $ python3  
Python 3.7.3 (default, Jul 25 2020, 13:03:44)  
[GCC 8.3.0] on linux  
Type "help", "copyright", "credits" or "license" for more information.  
>>> print("I love my Raspberry Pi!!")  
I love my Raspberry Pi!!  
>>> prin("I love my Raspberry Pi!!")  
Traceback (most recent call last):  
  File "<stdin>", line 1, in <module>  
NameError: name 'prin' is not defined  
>>> 
```

Learning about the Python Interactive Shell

- To get help using the interactive shell you can type `help()` and pass as parameter the name of the function you need help with.
- Ex.: If I want help about the print function use: `help(print)`

```
Help on built-in function print in module builtins:

print(...)
    print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False)

    Prints the values to a stream, or to sys.stdout by default.
    Optional keyword arguments:
    file: a file-like object (stream); defaults to the current sys.stdout.
    sep:  string inserted between values, default a space.
    end:  string appended after the last value, default a newline.
    flush: whether to forcibly flush the stream.

(END)
```

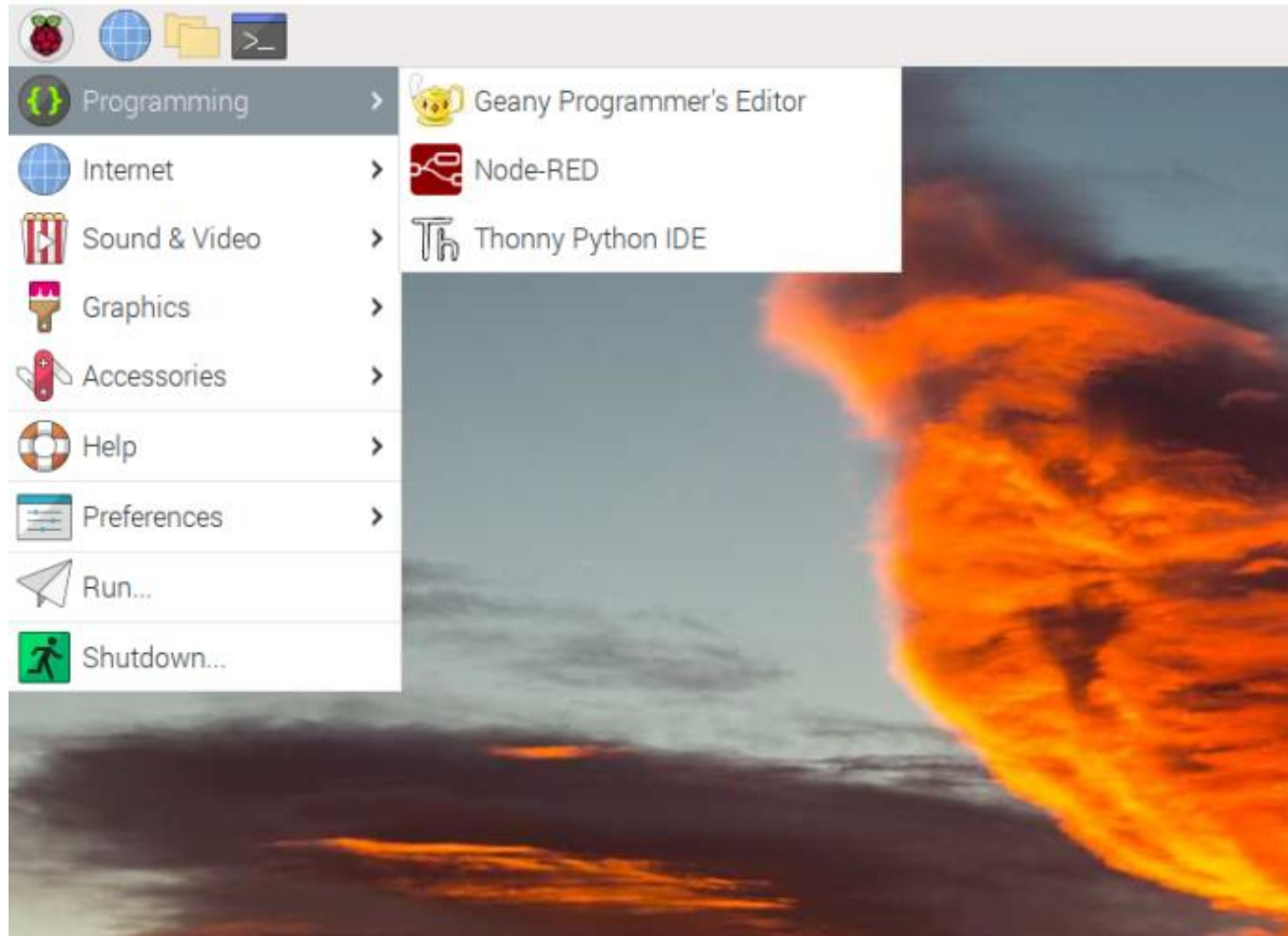
Using a Python development environment

- A development environment is a single tool for creating, running, testing and modifying Python scripts.
- Often development environments color code key syntax for easier identification of various statement features.
- This color coding helps with a script's testing, modifications and debugging.
- In addition to these features, a development environment can provide syntax checking so that you can find any incorrect Python syntax without having to run the entire Python script.

Using a Python development environment

- Several options to make a choice but I suggest VS Code with remote SSH to your Pi
- A text editor with MobaXterm (over ssh)
- Last option (less desirable): Thonny
 - Thonny is a small, simple and open-source IDE that let you create, debug and execute Python scripts (and comes installed by default).

Using a Python development environment



Using VS Code with remote access to your RPi

The screenshot shows the Visual Studio Code interface with a remote connection to a Raspberry Pi. The Explorer sidebar on the left displays the file system of the remote machine, including directories like `.vnc`, `.vscode-server`, `Adept_Compact_Sensor_Kit_for_RPi`, `Bookshelf`, `certs`, `Desktop`, `Documents`, `Downloads`, `Music`, `Pictures`, `Public`, `Templates`, `Videos`, and a `workspace` directory. The main editor area shows a file named `myfirst.py` with the following code:

```
1 print('hello')
2
```

The bottom panel shows the TERMINAL output with the command `/bin/python /home/pi/workspace/myfirst.py` and the output `hello`.

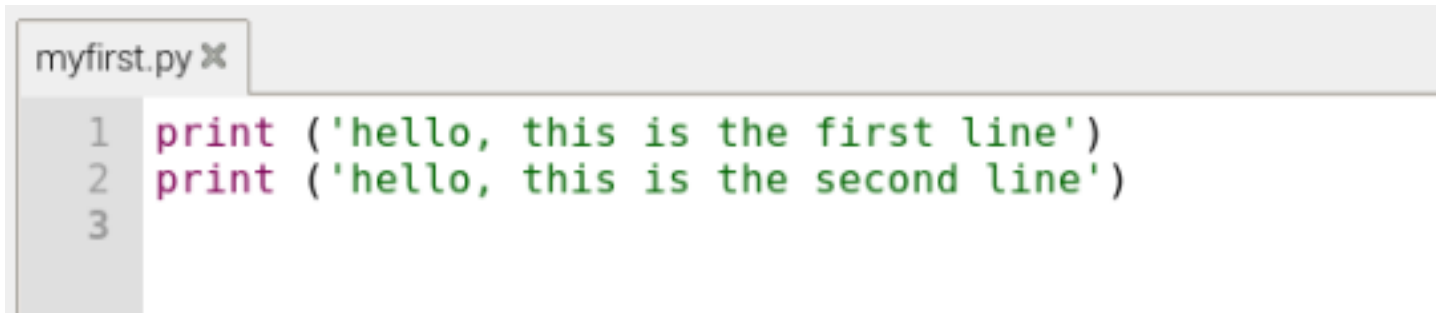
<https://singleboardblog.com/coding-on-raspberry-pi-remotely-with-vscode/>

How to create and run a Python script

- Instead of typing in each Python statement every time you need to run a program, you can create a whole file of Python statements and then run them.
- These files full of Python statements are called Python scripts

How to create and run a Python script

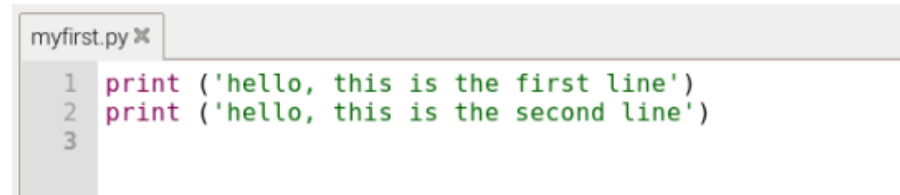
- Create a workspace directory (to keep your work organized)
 - `mkdir workspace`
 - `cd workspace`
- Create a file named `first.py`




```
myfirst.py ✕  
1 print ('hello, this is the first line')  
2 print ('hello, this is the second line')  
3
```

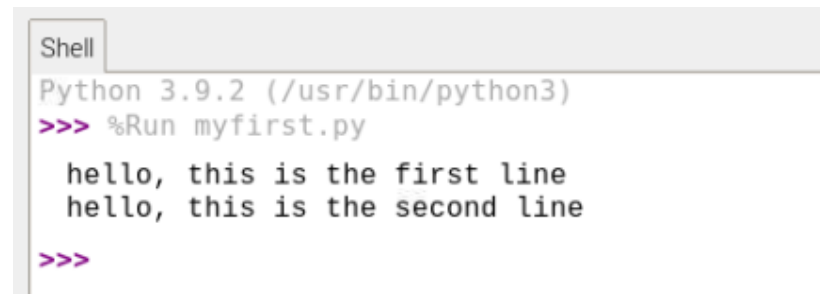
How to create and run a Python script

- You can Edit the script in the top section of Thonny



```
myfirst.py ✖  
1 print ('hello, this is the first line')  
2 print ('hello, this is the second line')  
3
```

- And where you are ready to run your script, you can click on the run button. 
- And you will get the output in the bottom section



```
Shell  
Python 3.9.2 (/usr/bin/python3)  
>>> %Run myfirst.py  
hello, this is the first line  
hello, this is the second line  
>>>
```

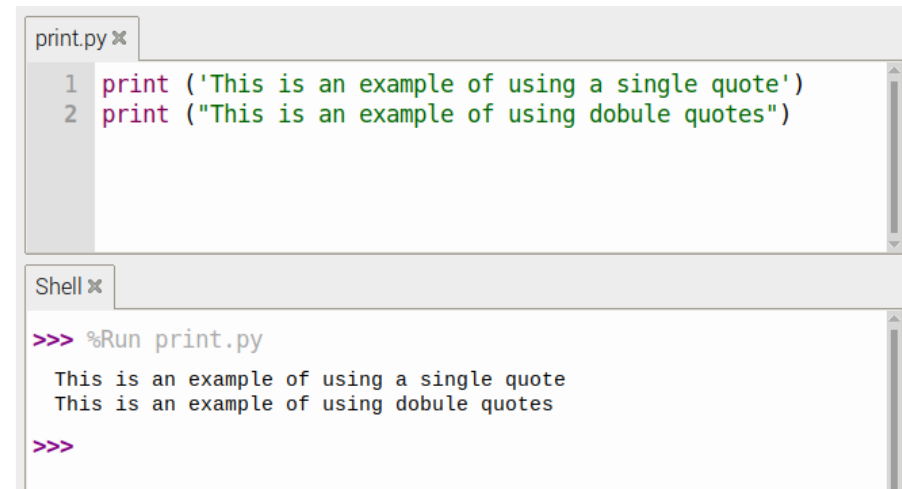
Understanding Python Basics

How to produce output from a script

- The `print()` function let you put on the screen some items.
- In Python v2 `print` is not a function, it became a function when Python v3 was created.
- This is important to know, in case you are trying to execute an old script in Python v2
- The function `print()` is a built-in function, because it is part of the Python standard functions library. You don't need to do anything special to get this function.
- The argument of the `print()` function can be characters, or values stored in variables.

How to produce output from a script

- If you want to print text, the argument of the print() function should be enclosed in either a set of single quotes or double quotes .



The screenshot shows a code editor window titled 'print.py' with two lines of Python code:

```
1 print ('This is an example of using a single quote')
2 print ("This is an example of using dobule quotes")
```

Below the code editor is a shell window titled 'Shell' showing the command to run the script and its output:

```
>>> %Run print.py
This is an example of using a single quote
This is an example of using dobule quotes
>>>
```

- What if I want to print multiple lines?

How to produce output from a script

- To print a blank line:

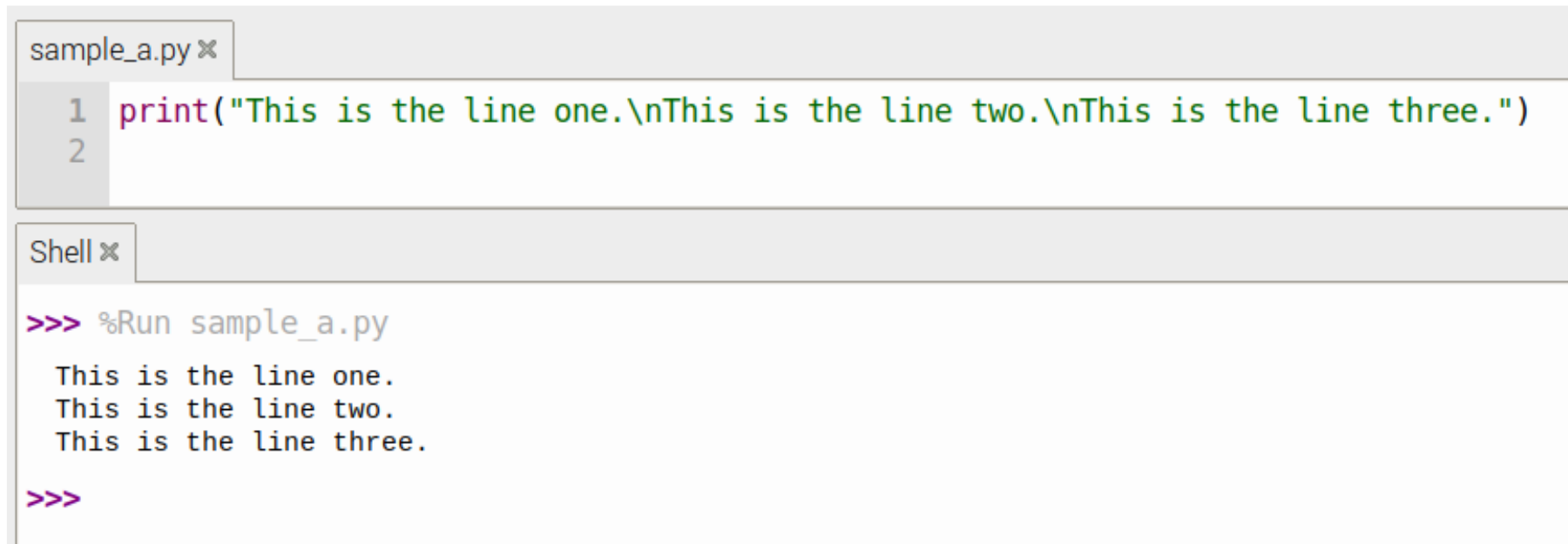
```
sample_a.py x
1 print("This is the first line.")
2 print()
3 print("This is the first line after a blank line.")
4

Shell x
>>> %Run sample_a.py
    This is the first line.

    This is the first line after a blank line.
>>> |
```

How to produce output from a script

- As in other programming languages we can use the Escape Sequences.



The image shows a code editor window with a tab labeled 'sample_a.py'. The code in the editor is:

```
1 print("This is the line one.\nThis is the line two.\nThis is the line three.")
2
```

Below the code editor is a shell window with a tab labeled 'Shell'. The shell shows the command to run the script and its output:

```
>>> %Run sample_a.py
This is the line one.
This is the line two.
This is the line three.
>>>
```

How to produce output from a script

- A Few Python Escape Sequences

<code>\"</code>	Print the next character as a double quote, not a string closer
<code>\'</code>	Print the next character as a single quote, not a string closer
<code>\n</code>	Print a new line character (remember our print statements?)
<code>\t</code>	Print a tab character
<code>\r</code>	Print a carriage return (not used very often)
<code>\\$</code>	Print the next character as a dollar, not as part of a variable
<code>\\</code>	Print the next character as a backslash, not an escape character

How to produce output from a script

- Thanks to the Unicode escape sequence, you can print all kinds of characters in your output.
- You can display Unicode character by using the `\u` escape sequence.
- Each Unicode character is represented by a hexadecimal number.
- These hexadecimal numbers are found at www.unicode.org/charts

```
sample_a.py *  
1 print("I love my Raspberry \u03c0!")  
2  
3  
Shell  
>>> %Run sample_a.py  
I love my Raspberry π!  
>>>
```

Making a script readable

- In scripts, comments are notes from the Python script author.
- A comment's purpose is to provide understanding of the script's syntax and logic.
- The Python interpreter ignores any comments.
- However, comments are invaluable to humans who need to modify or debug scripts.

Making a script readable

- To add a comment to a script, you precede it with the pound or hash symbol (#).
- The Python interpreter ignores anything that follows the hash symbol.

Thonny - /home/pi/workspace/myfirst.py @ 8:1

New Load Save Run Debug Over Into Out Stop

myfirst.py ✕

```

1 # myfirst.py - Demonstrate how to insert blank lines
2 # author: Gabriel Astudillo
3 # date: 15-01-2023
4 # version: 2.0
5 print('hello, this is the first line')
6 print() #this print a blank line
7 print('hello, this is the second line')
8

```

Shell

```

>>>
>>>
>>>
>>>
>>> %Run myfirst.py
hello, this is the first line
hello, this is the second line
>>>

```

How to use variables

- A variable is a name that stores a value for later use in a script.
- But... In fact, Python doesn't have variables. Instead, there are objects references!
- Because in python everything is an object!
- Rules about the name of a variable:
 - You cannot use a Python keyword as a variable name
 - The first character of a variable name cannot be a number
 - No spaces are allowed in a variable name

A note on strings

- We can also denote a string with triple quotes:

```
"""my string"""
```

```
'''my string'''
```

```
"my string"
```

```
'my string'
```

- are all the same thing. The difference with triple quotes is that it allows a string to extend over multiple lines.

```
my_str = """It was the best of times,  
it was the worst of times..."""
```

```
print(my_str)  
It was the best of times,  
it was the worst of times...
```


How to use variables

- The Python keywords list changes so often. Therefore, it is a good idea to look at the current keywords list before you start creating variable names.

A screenshot of a Python Shell window. The window has a title bar that says "Shell". Inside, there is a list of Python keywords. The code entered is: >>> import keyword, >>> print (keyword.kwlist). The output is a list of 33 keywords: ['False', 'None', 'True', '__peg_parser__', 'and', 'as', 'assert', 'async', 'await', 'break', 'class', 'continue', 'def', 'del', 'elif', 'else', 'except', 'finally', 'for', 'from', 'global', 'if', 'import', 'in', 'is', 'lambda', 'nonlocal', 'not', 'or', 'pass', 'raise', 'return', 'try', 'while', 'with', 'yield']. The prompt >>> is shown at the bottom left of the shell area. The text "Python 3.9.2" is visible in the bottom right corner of the window.

```
Shell
>>>
>>>
>>>
>>> import keyword
>>> print (keyword.kwlist)
['False', 'None', 'True', '__peg_parser__', 'and', 'as', 'assert', 'async', 'await', 'break', 'cl
ass', 'continue', 'def', 'del', 'elif', 'else', 'except', 'finally', 'for', 'from', 'global', 'if
', 'import', 'in', 'is', 'lambda', 'nonlocal', 'not', 'or', 'pass', 'raise', 'return', 'try', 'wh
ile', 'with', 'yield']
>>> |
```

Python 3.9.2

How to use variables

- For the first character in your Python variable name, you must not use a number.
- The first character in a variable name can be any of the following:
 - A letter a-z,
 - A letter A-Z,
 - The underscore character (`_`)
- After the first character in a variable name, the other character can be any of the following:
 - The numbers 0-9
 - The letters a-z
 - The letters A-Z
 - The underscore (`_`)

How to use variables

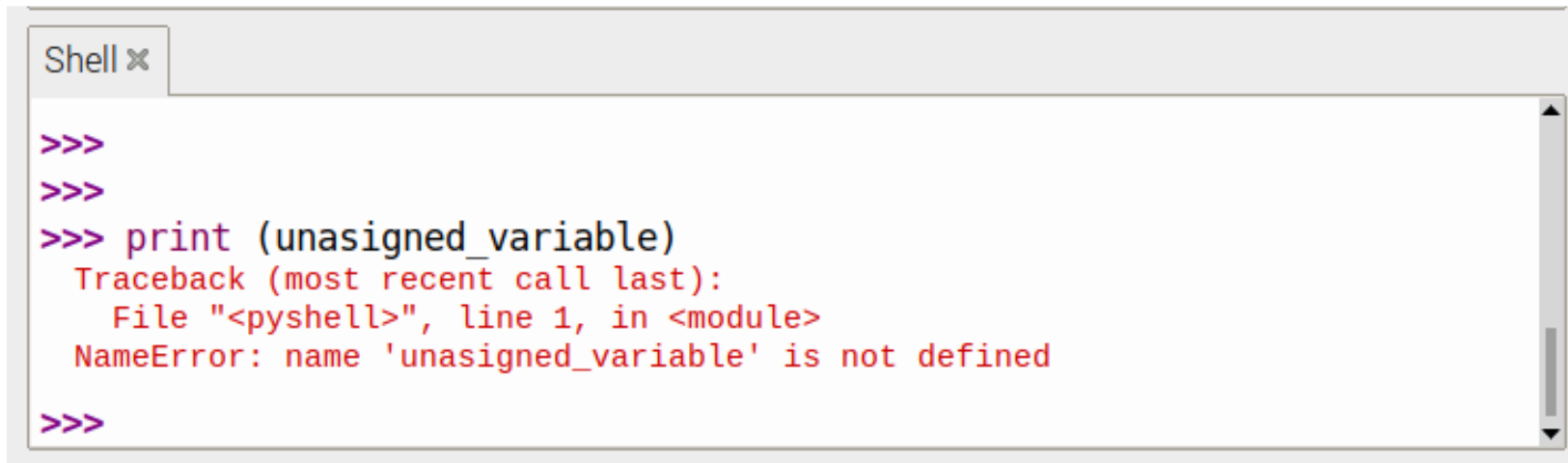
- Assigning a value to a Python variable is fairly straightforward. You put the variable name first, then an equal sign (=), and finish up with the value you are assigning to the variable.
- You can print text alongside a variable, separated by commas, in one print statement.
- Many more use cases:

<https://www.freecodecamp.org/news/python-print-variable-how-to-print-a-string-and-variable>

```
myfirst.py ✕  
1 first_name = "John"  
2 last_name = "Doe"  
3  
4 print("Hello", first_name, last_name, "good to see you")  
  
Shell  
  
>>> %Run myfirst.py  
Hello John Doe good to see you  
  
>>>
```

How to use variables

- You cannot use a variable until you have assigned a value to it.

A screenshot of a Python Shell window titled "Shell x". The window contains a Python prompt ">>>" followed by two blank lines, then the command "print (unsigned_variable)". Below this, a red traceback message is displayed: "Traceback (most recent call last):", "File "<pyshell>", line 1, in <module>", and "NameError: name 'unsigned_variable' is not defined". The prompt ">>>" appears again at the bottom of the window.

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

Type of data

- When a variable is created by an assignment such as `variable = value`, Python determines and assigns a data type to the variable.
- A data type defines how the variable is stored and the rules governing how the data can be manipulated.
- Python uses the variable's assigned value to determine its type.
- The basic data types:
 - float
 - int
 - long
 - str

Type of data

- You can determine which data type Python has assigned to a variable by using the `type()` function.

```
Shell x
>>>
>>> variable1 = 1
>>> type (variable1)
<class 'int'>
>>> variable2 = 1.3
>>> type (variable2)
<class 'float'>
>>> variable3 = "Hi!"
>>> type (variable3)
<class 'str'>
```

How to input information into a script?

- Sometimes you might need a script user to provide data into your script from the keyboard.
- To accomplish this task, Python provides the *input* function.
- The input function is a built-in function and has the following syntax:

```
variable = input (user prompt)
```

How to input information into a script?

- For the user prompt, you can enclose the prompt's string characters in either a single or double quote.
- The prompt is shown enclosed in double quotes.
- The input function treats all inputs **as strings**. This is different from how Python handles other variable assignments.

```
>>> variable = input ("Enter a value: ")
Enter a value: 5
>>> print(variable)
5
>>> type(variable)
<class 'str'>
>>> █
```


How to input information into a script?

- To convert variables (input from the keyboard) from strings, you can use the `int` function().
- The `int` function will convert a number from a string data type to an integer data type.
- You can use the `float()` function to convert a number from a string to a floating point data type.

How to input information into a script?

```
Shell x
>>> variable_int = input ("Enter an integer value:")
Enter an integer value:5
>>> variable_float = input ("Enter a float value:")
Enter a float value:3.14
>>> type (variable_int)
<class 'str'>
>>> type (variable_float)
<class 'str'>
>>> variable_int = int(input ("Enter an integer value: "))
Enter an integer value: 5
>>> variable_float = float(input ("Enter a float value :"))
Enter a float value :3.14
>>> type (variable_int)
<class 'int'>
>>> type (variable_float)
<class 'float'>
```

Operators

- Let's review the operators in Moodle

Lab. 2

CHECK THE LAB DOCUMENTS ON MOODLE