Python

- Exit the python interpreter: `CTRL+Z` or `quit()` Or `CTRL+D` in Linux
- To enter Interactive mode in python, this can be done by passing `-i` before the script. ig: `python -i console.py`

▼ **WHY RUN A SCRIPT IN INTERACTIVE MODE:**

1. Debugging and Testing: Running scripts interactively helps identify errors and issues by inspecting variable values and behavior.

2. Immediate Feedback: Interactive mode offers instant results after script execution for quick validation.

3. Exploring Results: Interactive mode allows step-by-step exploration of script-generated data and structures.

4. Iterative Development: Running scripts interactively supports making real-time code adjustments and improvements.

5. Educational and Learning Purposes: Interactive mode aids beginners in experimenting with code and understanding Python concepts.

You can use python as a calculator too:

- `17 / 3  # classic division returns a float` *Result: 5.666*

- `17 // 3` `# floor division discards the fractional part` *Result: 5*
- `5 ** 2` `# 5 squared` *Result: 5*5 = 25*
- Assignment:

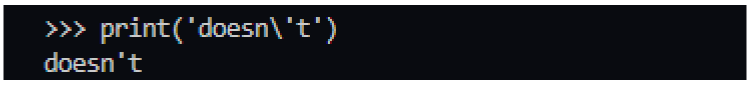
```
width = 20
height = 5 * 9
width * height
## result 900
```

- Strings so called `str` in python :

`'spam eggs'` `# single quotes`

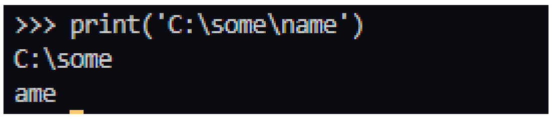
`"Paris rabbit got your back :)! Yay!"` `# double quotes`

To escape the `‘` like the word: `doesn’t` use `\` (escape cara)

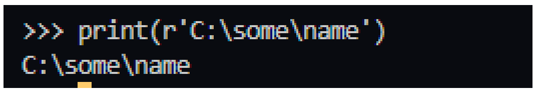


```
>>> print('doesn\'t')
doesn't
```

- Sometimes we need omitting the special characters like in this scenario:



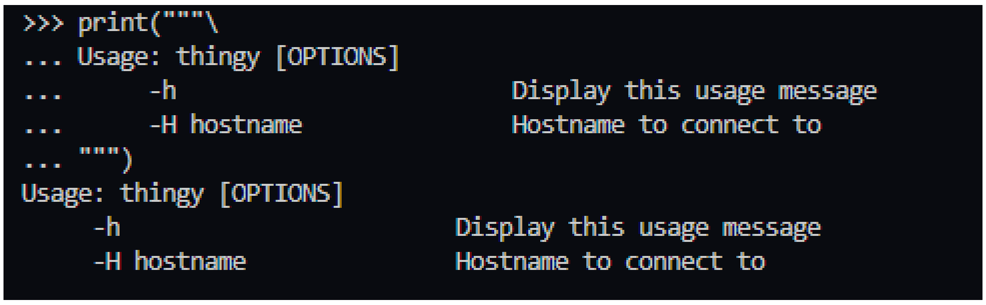
```
>>> print('C:\some\name')
C:\some
ame
```



```
>>> print(r'C:\some\name')
C:\some\name
```

Sythax: `print(r’C:\some\name’)`

- Output on the screen multiple lines at once:



```
>>> print("""\
... Usage: thingy [OPTIONS]
...      -h                        Display this usage message
...      -H hostname               Hostname to connect to
... """)
Usage: thingy [OPTIONS]
     -h                        Display this usage message
     -H hostname               Hostname to connect to
```

- **Syntax:**

```
print( “””\ kjfvlkdfjvlkfvjfd
                        lkljkdvjlvjslvksd
                        dslvksdlvdvkdskvsdvlds
            """)
## \ it means don't put a line break at the beginning of the
## must be one string that has multiple lines
```

- Strings can be concatenated (glued together) with the `+` operator, and repeated with `*`:

Sythax: `3 * 'un' + 'ium'` Result: `'unununium'`

- Two or more *string literals* (i.e. the ones enclosed between quotes) next to each other are automatically concatenated.

Sythax: `'Py' 'thon'` Result: `'Python'`

This feature is particularly useful when you want to break long strings:

```
text = ('Put several strings within parentheses '
        'to have them joined together.')
## output
##Put several strings within parentheses to have them joine
d together.'
```

NOTE: it only works with strings not variables though!

- You can create a variable using bash, like this: (`myvar`)

```
#!/bin/bash
python3 $PYFILE
```

- Then create a file python (script you’d like to be run each time you run your bash file)

Here we will call it `main.py`

```
#!/usr/bin/python3
print("Best School")
```

- Then in cmd: export `export PYFILE=main.py` (env var = the script name)
- Then you can just run the bash file we call it, `myvar` like this: `./myvar` and you get this output `Best School`

Here f-strings displays:

```
#!/usr/bin/python3
number = 98
name = "Omar"

print(f"{number:d} Battery street")
# dsplay as decimal
print(f"Hi, Si {name:s}.")
# display as string

print("Var {Number} type is:", type(number))
print("Var {Name} type is: ", type(name))
```

- Last character in a string

```
number = 123.456789
number = str(number)
print(f'Last Caracter in our string:  {number[-1]}')
```

- Display only 2 decimals after (precision 2 digits)

```
#!/usr/bin/python3
number = 3.14159
print(f"Float: {number:.2f}")
```

- Indexes a little overview:

```
name = "Boujemaa"

# print(name[0]) #B
# print(name[1]) #O
# print(name[2]) #U
# print(name[3]) #J
# print(name[4]) #E
# print(name[5]) #M
# print(name[6]) #A
# print(name[7]) #A

# print(name[1:]) # OUJEMAA
print(name[:-2])  #Boujem
```

- Supposedly we have: 123.456789 (we want to display only 23.45) 2 digit precision and 2 digits before the coma)

```
# First Solution
number = 123.456789
formatted_number = f"{number:.2f}" # 2 cara after decimal
formatted_number[-1] # output last caracter
formatted_number = formatted_number[-5:]  # Extract the last
print(f"Formatted number: {formatted_number}")



# Second Solution
number = 123.456789
number = number % 100

print(f"The digit that I want:  {number:.2f}")
```