Programming in python™

A presentation by:

- ❑ Arun Kumar

Python Basic Syntaxes

- Identifiers
- Reserved keywords
- Lines and Indentation
- Multi-Line, Quotations, comment
- user input
- Multiple statements on Single Line
- Command Line Arguments

Identifiers

“An identifier is a name given to an entity”

In very simple words, an identifier is a user-defined name to represent the basic building blocks of Python. It can be a **variable**, a **function**, a **class**, a **module**, or any other object.

Naming Rules for Identifiers

1. The Python identifier is made with a **combination** of **lowercase** or **uppercase letters**, **digits** or an **underscore**.
These are the valid characters.
•Lowercase letters (a to z)
•Uppercase letters (A to Z)
•Digits (0 to 9)
•Underscore (_)

Examples of a valid identifier:

•num1
•FLAG
•get_user_name
•userDetails
•_1234
2. An identifier cannot start with a **digit**. If we create an identifier that starts with a digit then we will get a **syntax error**.

Reserved Keywords

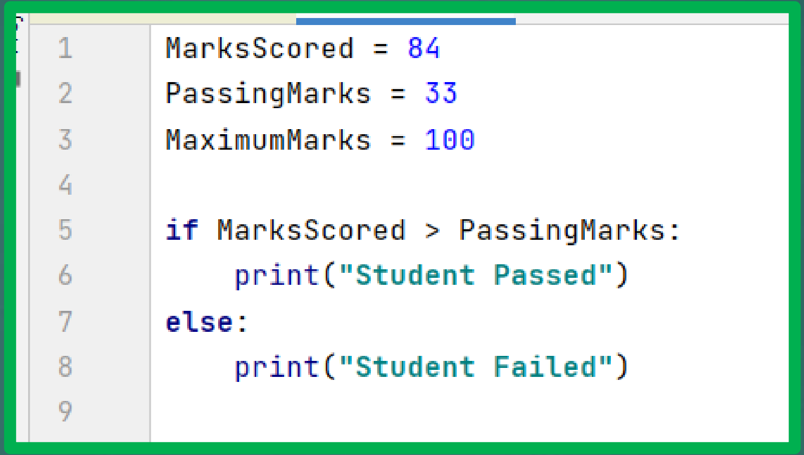
The following list shows the Python keywords. These are reserved words and you cannot use them as constant or variable or any other identifier names. All the Python keywords contain lowercase letters only.

and	exec	not
assert	finally	or
break	for	pass
class	from	print
continue	global	raise
def	if	return
del	import	try
elif	in	while
else	is	with
except	lambda	yield

Lines and Indentation

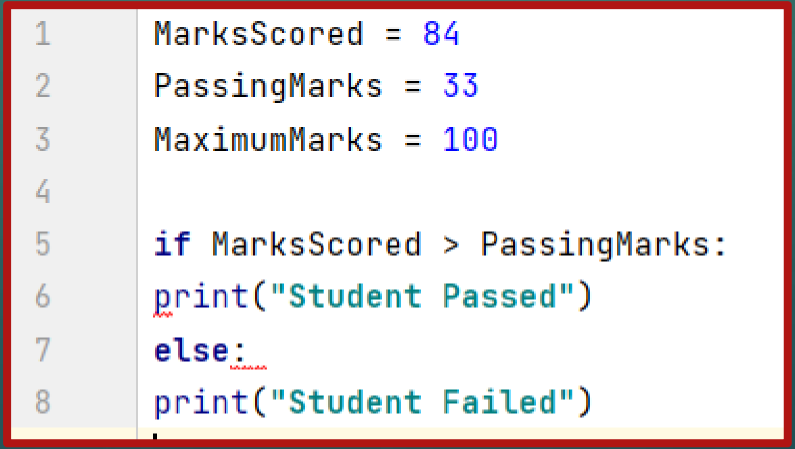
Python provides no braces to indicate blocks of code for class and function definitions or flow control. Blocks of code are denoted by line indentation, which is rigidly enforced.
The number of spaces in the indentation is variable, but all statements within the block must be indented the same amount.

For example, Correct Indentation −



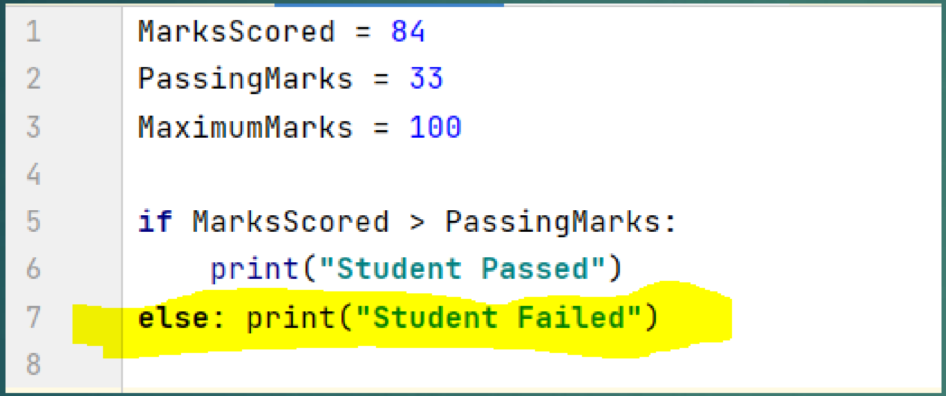
```python
MarksScored = 84
PassingMarks = 33
MaximumMarks = 100

if MarksScored > PassingMarks:
    print("Student Passed")
else:
    print("Student Failed")

```

However, the following block generates an error −



```python
MarksScored = 84
PassingMarks = 33
MaximumMarks = 100

if MarksScored > PassingMarks:
print("Student Passed")
else:
print("Student Failed")
```

However , A single line code can still be aligned in same line without indentation

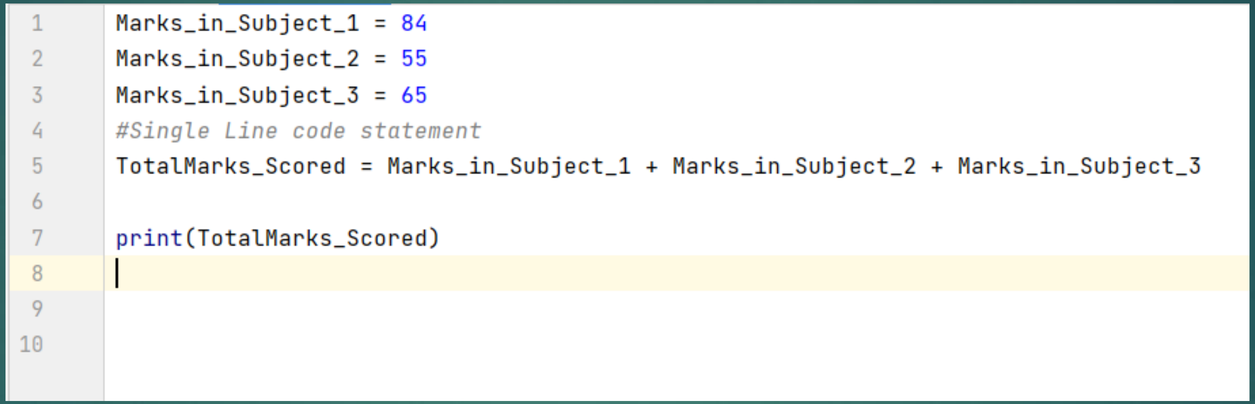


```
MarksScored = 84
PassingMarks = 33
MaximumMarks = 100

if MarksScored > PassingMarks:
    print("Student Passed")
else: print("Student Failed")

```

Multi-Line Code statements

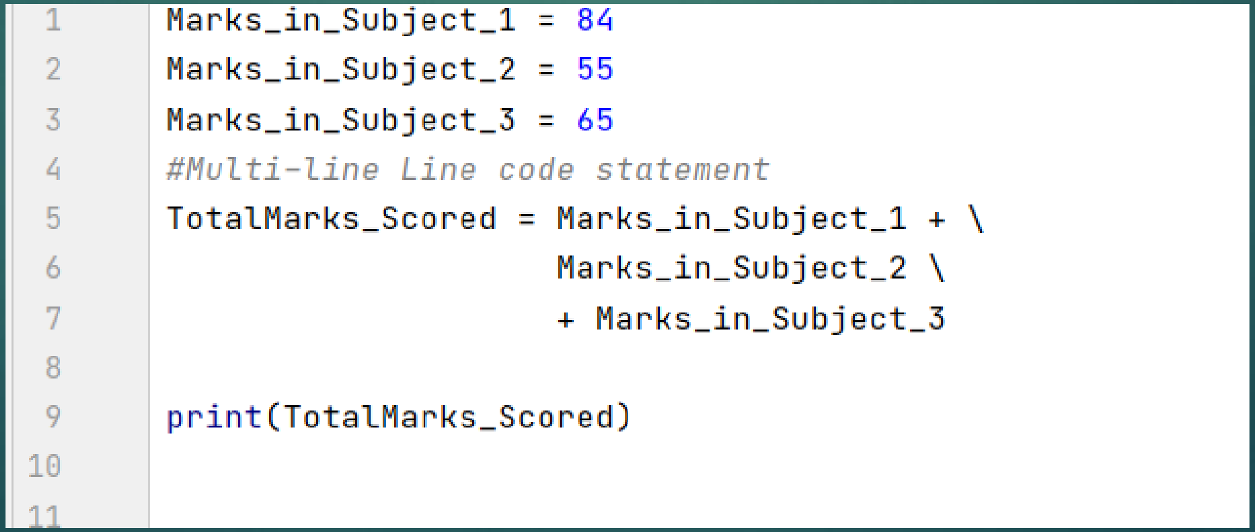
Statements in Python typically end with a new line. Python does, however, allow the use of the line continuation character (\) to denote that the line should continue.

Single Line code statement



```python
Marks_in_Subject_1 = 84
Marks_in_Subject_2 = 55
Marks_in_Subject_3 = 65
#Single Line code statement
TotalMarks_Scored = Marks_in_Subject_1 + Marks_in_Subject_2 + Marks_in_Subject_3

print(TotalMarks_Scored)
```

Multi-Line code statement



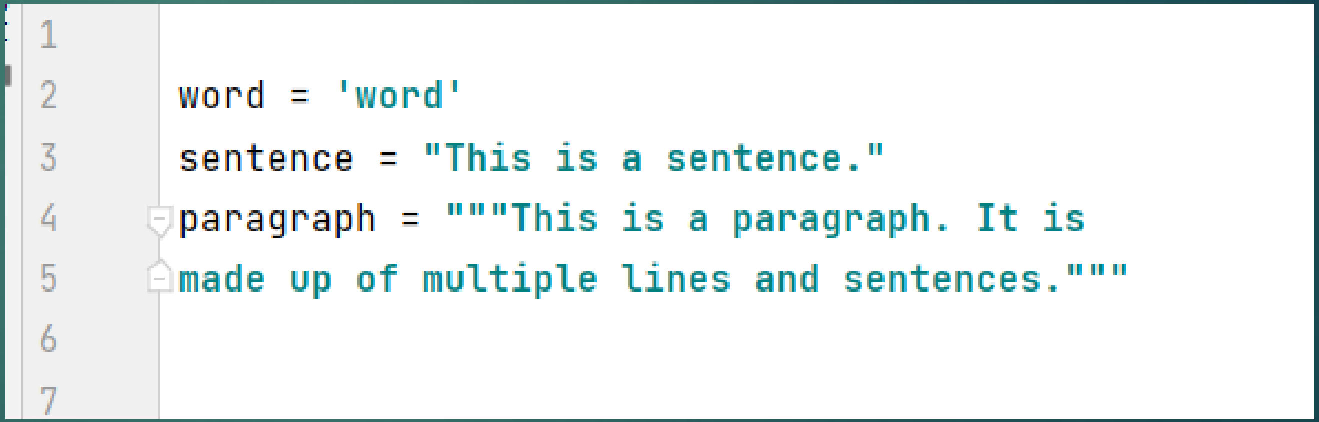
```python
Marks_in_Subject_1 = 84
Marks_in_Subject_2 = 55
Marks_in_Subject_3 = 65
#Multi-line Line code statement
TotalMarks_Scored = Marks_in_Subject_1 + \
                    Marks_in_Subject_2 \
                    + Marks_in_Subject_3

print(TotalMarks_Scored)
```

Quotation in Python

Python accepts single ('), double (") and triple (''' or """) quotes to denote string literals, as long as the same type of quote starts and ends the string.

The triple quotes are used to span the string across multiple lines.

For example, all these are legal −



```

word = 'word'
sentence = "This is a sentence."
paragraph = """This is a paragraph. It is
made up of multiple lines and sentences."""


```

Comments Python

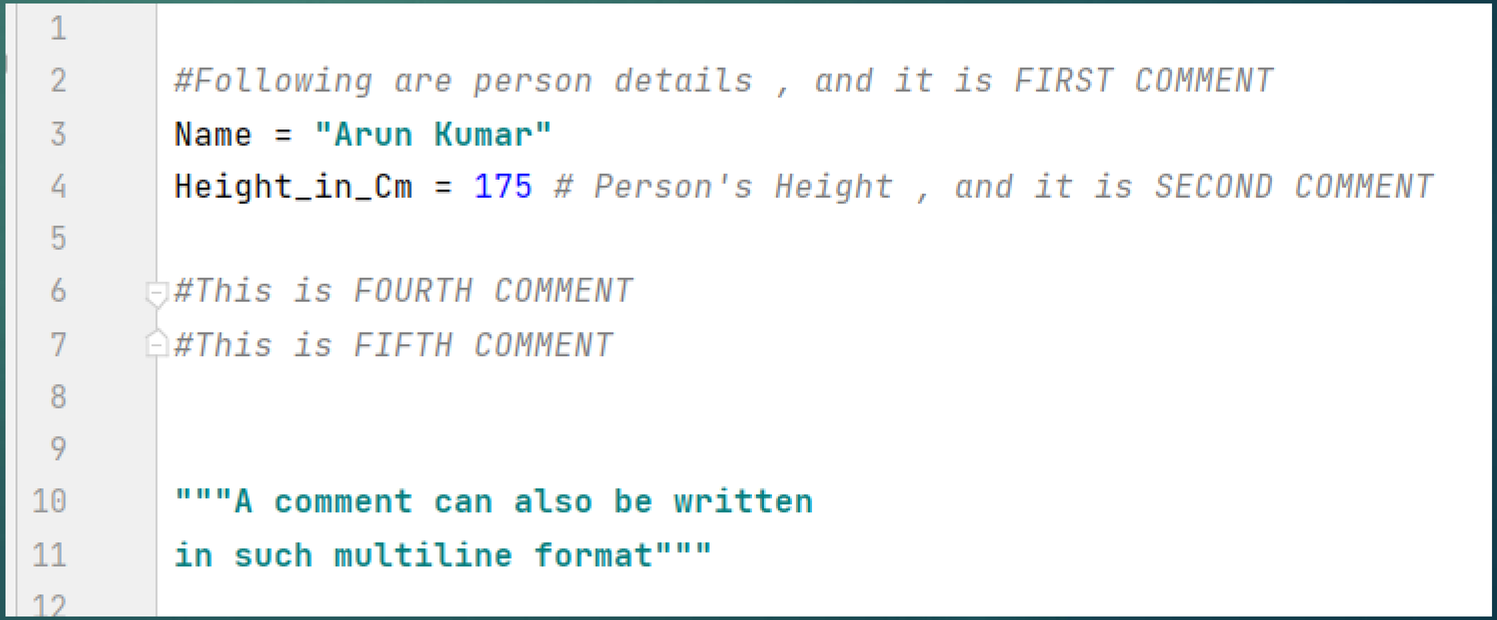
A comment is a statement in Python that is ignored by Python-interpreter while converting code to machine language.

A hash sign (#) that is not inside a string literal begins a comment.

All characters after the # and up to the end of the physical line are part of the comment and the Python interpreter ignores them.

Also , A comment can be contained inside quotes.

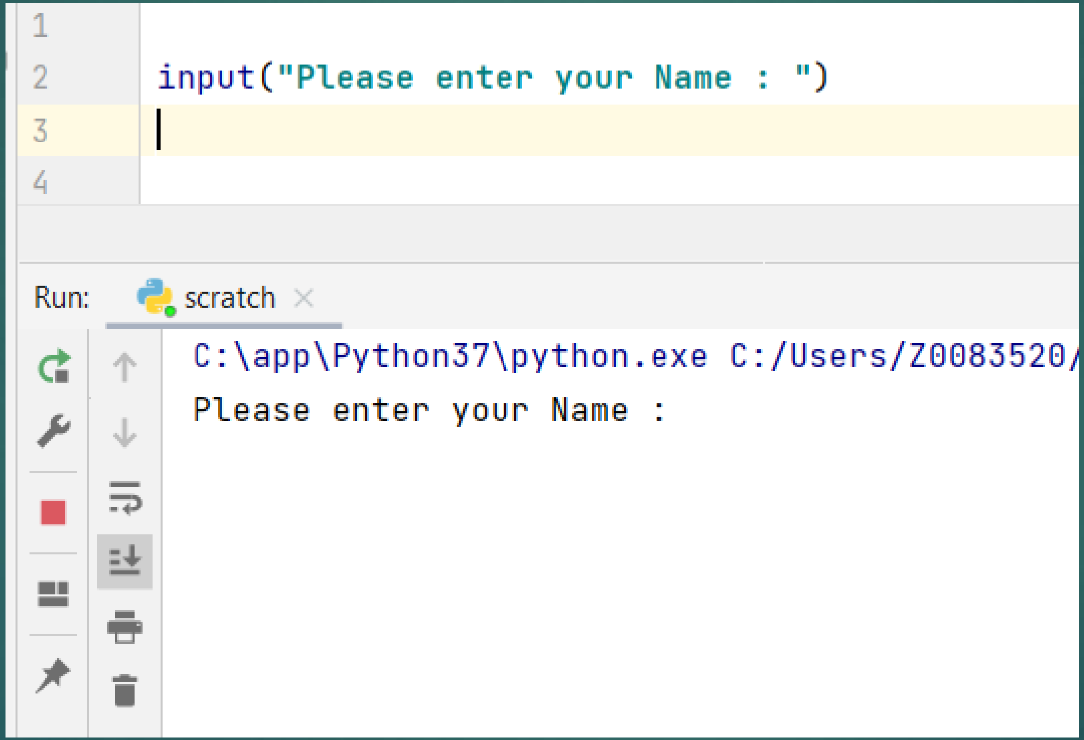
For example, all these are legal −



```

#Following are person details , and it is FIRST COMMENT
Name = "Arun Kumar"
Height_in_Cm = 175 # Person's Height , and it is SECOND COMMENT

#This is FOURTH COMMENT
#This is FIFTH COMMENT


"""A comment can also be written
in such multiline format"""
```

User Input

The following line of the program displays the prompt, the statement saying “Please enter your name”, and waits for the user to provide details−



```

input("Please enter your Name : ")


```

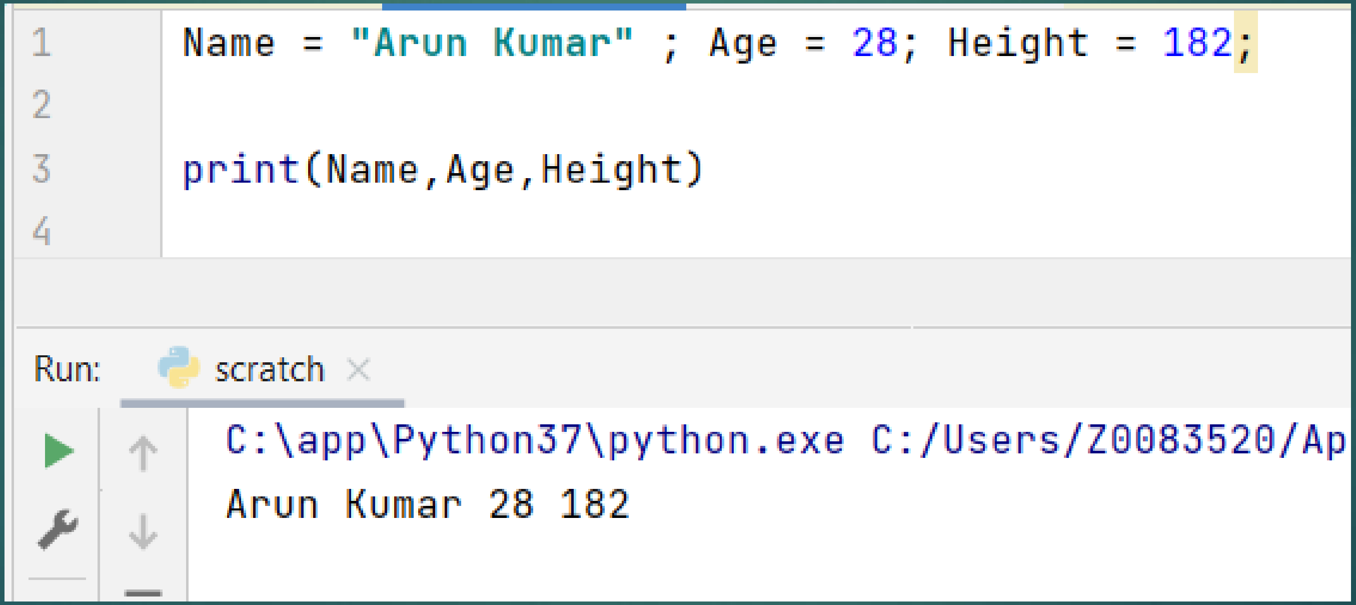
Run: scratch

```
C:\app\Python37\python.exe C:/Users/Z0083520/
Please enter your Name :
```

Multiple Statements on a Single Line

The semicolon (;) allows multiple statements on the single line given that neither statement starts a new code block.

Here is a sample snip using the semicolon −



```
Name = "Arun Kumar" ; Age = 28; Height = 182;

print(Name,Age,Height)

```

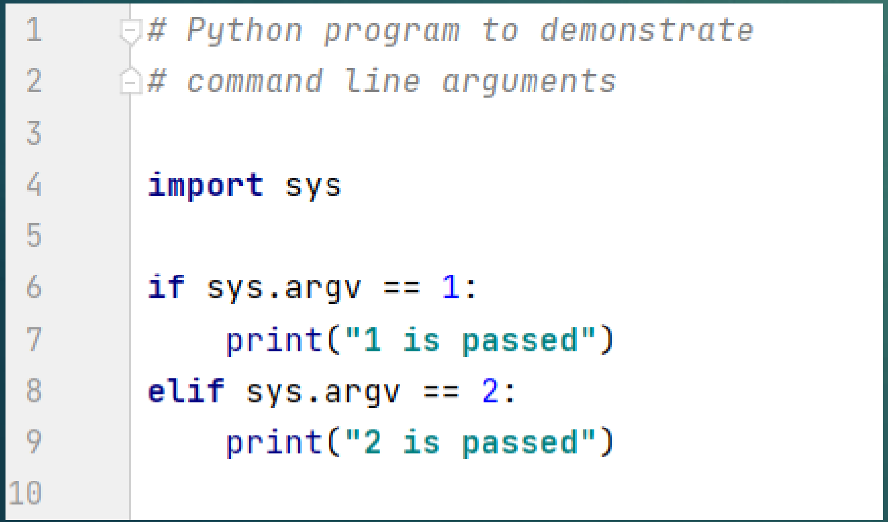
```
C:\app\Python37\python.exe C:/Users/Z0083520/Ap
Arun Kumar 28 182
```

Command Line arguments/Parameters

The arguments that are given after the name of the program in the command line shell of the operating system are known as **Command Line Arguments**. Python provides various ways of dealing with these types of arguments.

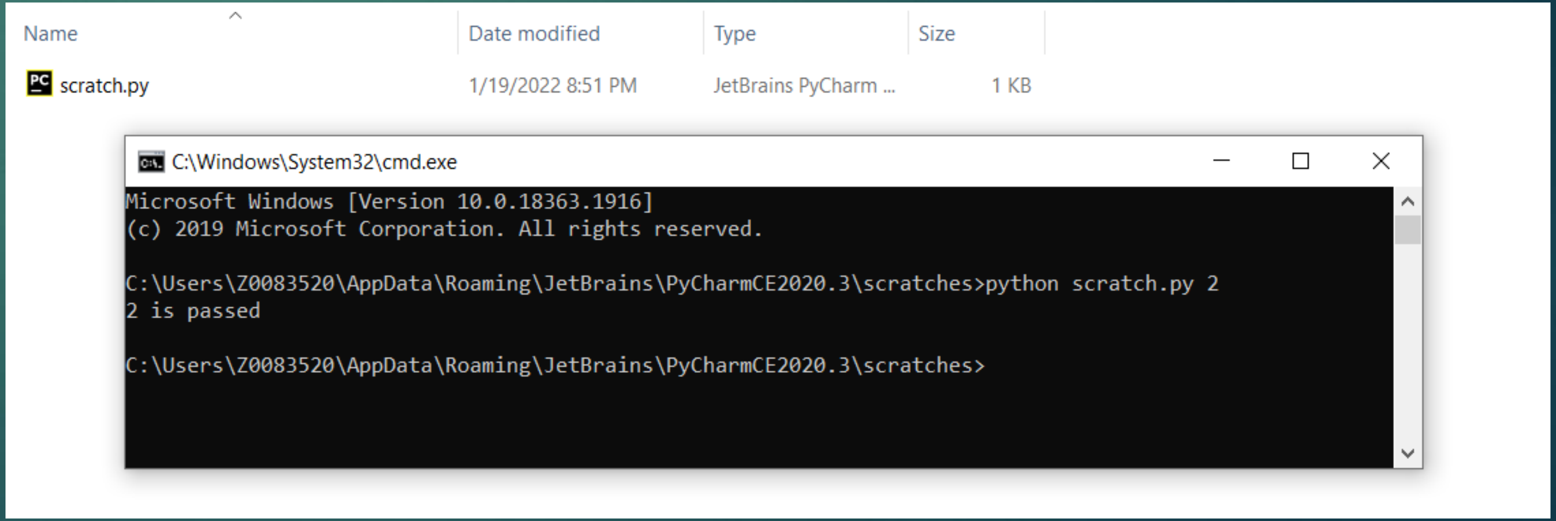
The three most common are:
Using sys.argv
Using getopt module
Using argparse module

Code



```python
# Python program to demonstrate
# command line arguments

import sys

if sys.argv == 1:
    print("1 is passed")
elif sys.argv == 2:
    print("2 is passed")

```

Output



Name	Date modified	Type	Size
scratch.py	1/19/2022 8:51 PM	JetBrains PyCharm ...	1 KB

C:\Windows\System32\cmd.exe

```
Microsoft Windows [Version 10.0.18363.1916]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\Z0083520\AppData\Roaming\JetBrains\PyCharmCE2020.3\scratches>python scratch.py 2
2 is passed

C:\Users\Z0083520\AppData\Roaming\JetBrains\PyCharmCE2020.3\scratches>
```