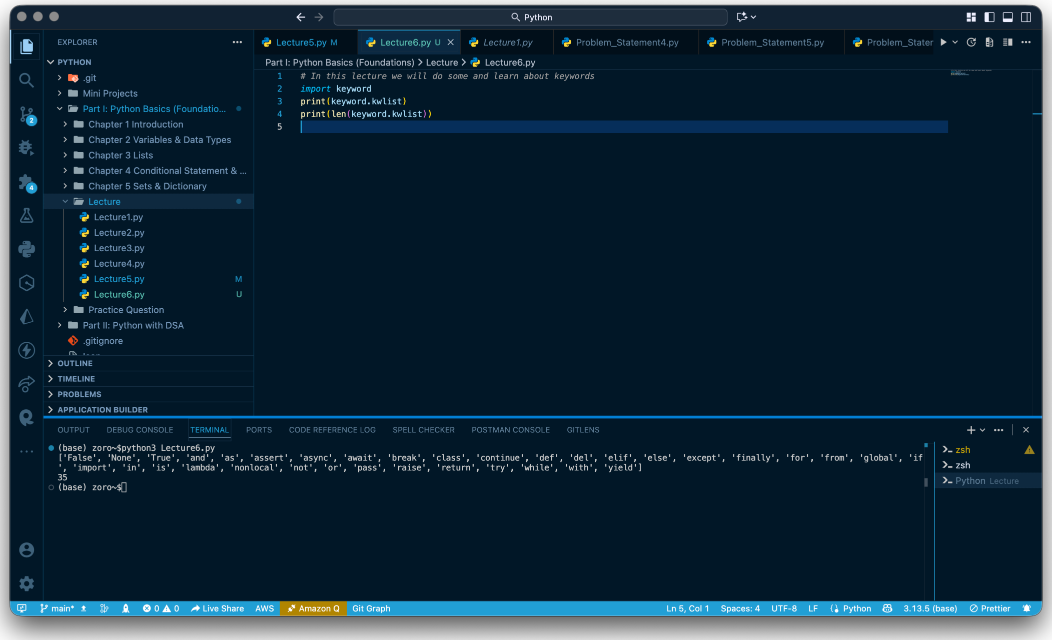
Python Keywords

Keyword	***Meaning***	***Example***	***Use Case***
False	Boolean false value	flag = False	Default/off switch in config states
True	Boolean true value	is_active = True	Feature flags, condition checks
None	No value / null	result = None	Represent missing DB values / API empty return
and	Logical AND	if x>5 and y<10:	Multi-condition validation
or	Logical OR	if a==0 or b==0:	Fall-back condition checks
not	Logical NOT	if not is_ready:	Authorization checks (not is_admin)
as	Alias	import numpy as np	Shorten module names
assert	Debug check	assert x>0	Fail fast in production validation
break	Exit loop	if x == 5: break	Stop search once result found
continue	Skip iteration	if x % 2 == 0: continue	Skip invalid/empty data rows
class	Define class	class Car:	OOP models (User, Cart, API client)
def	Define function	def send_mail():	Encapsulate logic / microservices
del	Delete variable	del x	Remove cached or temporary objects
elif	Else-if condition	elif score >= 50:	Multi-branch logic evaluations
else	Default block	else: print("Fail")	Fallback logic
if	Condition	if x > 0:	Decision-making logic
try	Try block	try:	Prevent errors crashing app
except	Handle error	except ZeroDivisionError:	Graceful error recovery
finally	Always runs	finally:	Close DB/file connections
raise	Raise error	raise ValueError("Error")	Enforce rules / error handling
for	Loop	for i in range(5):	Iterate lists, DB rows, JSON
while	Loop until false	while x > 0:	Event waiting, retry logic
from	Import specific	from math import sqrt	Import only what needed (lightweight)
import	Load module	import os	Add external libs
global	Modify global var	global count	Change values across functions
nonlocal	Outer function variable	nonlocal y	Nested function state management
in	Membership check	if 5 in arr:	Search in list/string/dictionary
is	Identity check	if x is None:	Check None or singleton object
lambda	Anonymous function	lambda x: x*2	Short callbacks (sorting, filtering)
pass	Do nothing	pass	Placeholder while designing class/functions
return	Return value	return total	Output from functions/API
yield	Generator return	yield x	Stream large data efficiently

with	Context manager	with open('a.txt'):	Auto manage resources (file/DB)
match	Pattern match	match value:	Cleaner switch-case routing (Py 3.10+)
case	Match branch	case 1:	Route logic based on value patterns

A Quick Verification Code:



```python
# In this lecture we will do some and learn about keywords
import keyword
print(keyword.kwlist)
print(len(keyword.kwlist))
```

```
(base) zoro~$python3 Lecture6.py
['False', 'None', 'True', 'and', 'as', 'assert', 'async', 'await', 'break', 'class', 'continue', 'def', 'del', 'elif', 'else', 'except', 'finally', 'for', 'from', 'global', 'if
', 'import', 'in', 'is', 'lambda', 'nonlocal', 'not', 'or', 'pass', 'raise', 'return', 'try', 'while', 'with', 'yield']
35
(base) zoro~$
```