PYTHON KEYWORDS

and:

Used to combine two conditions in a boolean expression.

Example:

```
x > 0 and y > 0
```

as:

Used to create an alias for a module name.

Example:

```
import numpy as np
```

assert:

Used for debugging purposes to check if a condition is true, and raise an error if it's not.

Example:

```
assert x > 0
```

/idineshkrishnan

async:

Used to declare an asynchronous function,

Example:

```
async def fetch_data():
```

await:

Used inside an asynchronous function to wait for a coroutine to complete.

Example:

```
data = await fetch_data()
```

break:

Used to break out of a loop.

Example:

```
for i in range(10):
 if i == 5: break
```

PYTHON KEYWORDS

class:

Used to define a new class.

Example:

```
class Dog:
```

continue:

Used to skip the current iteration of a loop and continue with the next iteration.

Example:

```
for i in range(10):
if i % 2 == 0: continue
```

def:

Used to define a new function.

Example:

```
def greet(name):
```

del:

Used to delete an object.

Example:

```
del x
```

elif:

Used in an if statement to specify additional tests.

Example:

```
if x > 0: print("positive")
elif x < 0: print("negative")
```

else:

Used in an if statement to specify a default action.

Example:

```
if x > 0: print("positive")
else: print("zero or negative")
```

PYTHON KEYWORDS

except:

Used with try statement to handle exceptions.

Example:

try: x = 1/0 except ZeroDivisionError:
print("division by zero")

False:

Boolean value representing false.

Example:

if False: print("this won't be printed")

finally:

Used with try statement to specify code that should be executed no matter what.

Example:

try: x = 1/0 finally:
print("this will be printed")

PYTHON KEYWORDS

for:

Used to create a for loop.

Example:

```
for i in range(10):
print(i)
```

from:

Used to import specific elements from a module.

Example:

```
from math import pi
```

global:

Used to declare a global variable.

Example:

```
global x; x = 10
```

PYTHON KEYWORDS

if:

Used to create a conditional statement.

Example:

```
if x > 0:
print("positive")
```

import:

Used to import a module.

Example:

```
import math
```

in:

Used to check if a value is in a sequence.

Example:

```
if x in [1, 2, 3]:
print("found")
```

PYTHON KEYWORDS

is:

Used to test object identity.

Example:

```
if x is None:
print("x is None")
```

lambda:

Used to create anonymous functions.

Example:

```
f = lambda x:
x + 1
```

None:

Represents a null value.

Example:

```
x = None
```

nonlocal:

Used to declare a non-local variable.

Example:

```
def outer_function():
    x = 10
    def inner_function():
        nonlocal x
        x += 1
        print(x)
    inner_function()
```

not:

Used to negate a boolean expression.

Example:

```
if not x:
print("x is False")
```

PYTHON KEYWORDS

or:

Used to combine two conditions in a boolean expression.

Example:

```
if x > 0 or y > 0:
print("at least one is positive")
```

pass:

Used as a placeholder where a statement is required,
but no action is needed.

Example:

```
if x < 0: pass.
```

raise:

Used to raise an exception.

Example:

```
raise ValueError("invalid value")
```

PYTHON KEYWORDS

return:

Used to return a value from a function.

Example:

```
def square(x):
return x * x
```

True:

Boolean value representing true.

Example:

```
if True:
print("this will be printed")
```

try:

Used to catch exceptions.

Example:

```
try: x = 1/0 except ZeroDivisionError:
print("division by zero")
```

PYTHON KEYWORDS

while:

Used to create a while loop.

Example:

```
while x < 10:
x += 1
```

with:

Used with the with statement to simplify exception handling.

Example:

```
with open("file.txt") as f:
data = f.read()
```

yield:

Used in a generator function to return a value and remember the state of the generator function.

Example:

```
def count_up_to(x):
i = 1 while i <= x:
yield i i += 1
```