In Python, break, continue, [return], and pass are control flow statements used within loops and conditionals.

```
# The break statement is used to exit a loop prematurely, regardless of the loop's condition.
# When break is executed, the loop stops immediately, and the program continues with the next statement after the loop.

for i in range(1, 11):
    if i == 5:
        break
    print(i)
```

```
1
2
3
4
```

```
# The continue statement is used to skip the rest of the code inside the loop for the current iteration and move on to the next :
# Unlike break, it doesn't exit the loop entirely but skips the current loop iteration.

for i in range(1, 11):
    if i == 5 or i == 6:
        continue
    print(i)
```

```
1
2
3
4
7
8
9
10
```

```
# The pass statement is a null operation; it doesn’t do anything when executed.
# It’s often used as a placeholder in situations where a statement is syntactically required but you don’t want to write any cc

for i in range(1, 6):
    if i == 3:
        pass
    print(i)

def myfunct():
  pass

myfunct()
```

```
1
2
3
4
5
```

```
# The return statement in Python is used to exit a function and send a value (or multiple values) back to the caller.
# It is an essential part of defining functions that process data and return results.

def sum(a,b):
  return a+b

print(sum(12,3))


def sorting(a,b):
  return ((a,b) if a>=b else (b,a))

print(sorting(1,3))
```

```
15
(3, 1)
```

```
def get_coordinates():
    x, y = 10, 20
    return x, y  # Returning multiple values as a tuple
```

```
coordinates = get_coordinates()
print(coordinates)  # Output: (10, 20)
```

```
(10, 20)
```

```
def get_numbers():
    return [1, 2, 3, 4, 5]  # Returning a list

numbers = get_numbers()
print(numbers)  # Output: [1, 2, 3, 4, 5]
```

```
[1, 2, 3, 4, 5]
```

```
def get_person():
    return {"name": "Alice", "age": 25}  # Returning a dictionary

person = get_person()
print(person)  # Output: {'name': 'Alice', 'age': 25}
```

```
{'name': 'Alice', 'age': 25}
```

```
# If a function does not have a return statement, it automatically returns None.
def greet():
    pass

result = greet()
print(result)


def greet():
    print("Hello!")

result = greet()  # No return statement
print(result)  # Output: Hello! \n None
```

```
None
Hello!
None
```

```
def evenOdd(a):
  if(a%2==0):
    return "even"
  return "odd"

print(evenOdd(5))
```

```
odd
```

```
def factorial(n):
    if n == 0 or n == 1:
        return 1
    return n * factorial(n - 1)

print(factorial(5))
```

```
120
```