Statements and Syntax

Session 03

Session Overview

In this session, you will be able to:

- Explain the different statements that are used in Python
- Describe the different branching and looping statements
- Explain how to use the import statement to access libraries and other functions

Statements in Python (1-5)

General Statements used in Python program:

Statement	Role	Example
Assignment	Creating references	`a, b = 'good', 'bad'`
Calls and Other operations	Running functions	`log.write("spam, ham")`
print calls	printing objects	`print('The Killer', joke)`
if/else/elif	Selecting actions	`if "python" in text:` `  print(text)`
for/else	Iteration	`for x in mylist:` `  print(x)`

Statements in Python (2-5)

Statement	Role	Example
while/else	General loops	`while X > Y:` `  print('hello')`
Pass	Empty Placeholder	`while True:` `  pass`
Break	Loop exit	`while True:` `  if exittest(): break`
continue	Loop continue	`while True:` `  if skiptest(): continue`
def	Deleting reference	`def f(a, b, c=1, *d):` `  print(a+b+c+d[0])`

Statements in Python (3-5)

Statement	Role	Example
return	Function Results	`def f(a, b, c=1, *d):` `  return a+b+c+d[0]`
yeild	Generator Functions	`def gen(n):` `  for i in n: yield i*2`

global	Namespaces	`x = 'old'`
		`def function():`
		`  global x, y; x = 'new'`

nonlocal	Namespace(3.x)	`def outer():`
		`  x = 'old`
		`  def function():`
		`  nonlocal x; x = 'new'`

Statements in Python (4-5)

Statement	Role	Example
import	Module access	`import sys`
from	Attribute access	`from sys import stdin`

class	Building objects	`class Subclass(Superclass):`
		`  staticData = []`
		`  def method(self): pass`

Statements in Python (5-5)

Statement	Role	Example

try/except/finally	Catching Exceptions	`try:`
		`  action()`
		`except:`
		`  print('action error')`

raise	Triggering Exceptions	`raise EndSearch(location)`
assert	Debugging Checks	`assert X > Y, 'X too small'`

del	Deleting References	`del data[k]`
		`del data[i:j]`
		`del obj.attr`

Assignment Statements (1-2)

Properties of assignment statements:

- Save references to objects in names or data structure elements
- Pre-declaring the names is not required
- Important to assign names before referencing them
- Some operations execute assignments implicitly

Assignment Statements (2-2)

Assignment statements along with syntax:

Statement form	Syntax
Basic form	`spam = 'Spam'`
Tuple assignment (positional)	`spam, ham = 'yum', 'YUM'`
List assignment (positional)	`[spam, ham] = ['yum', 'YUM']`
Sequence assignment, generalised	`a, b, c, d = 'spam'`
Extended sequence unpacking (Python 3.X)	`a, *b = 'spam'`
Multiple-target assignment	`spam = ham = 'lunch'`
Augmented assignment (equivalent to spams = spams + 33)	`spams += 33`

Syntax of Python

- A new syntax used is the colon character (:).
- In Python, all compound statements use a general pattern.
- Parentheses are optional in the code.
- The second important component that cannot be seen in Python code is the semicolon (;).

Print in Python (1-2)

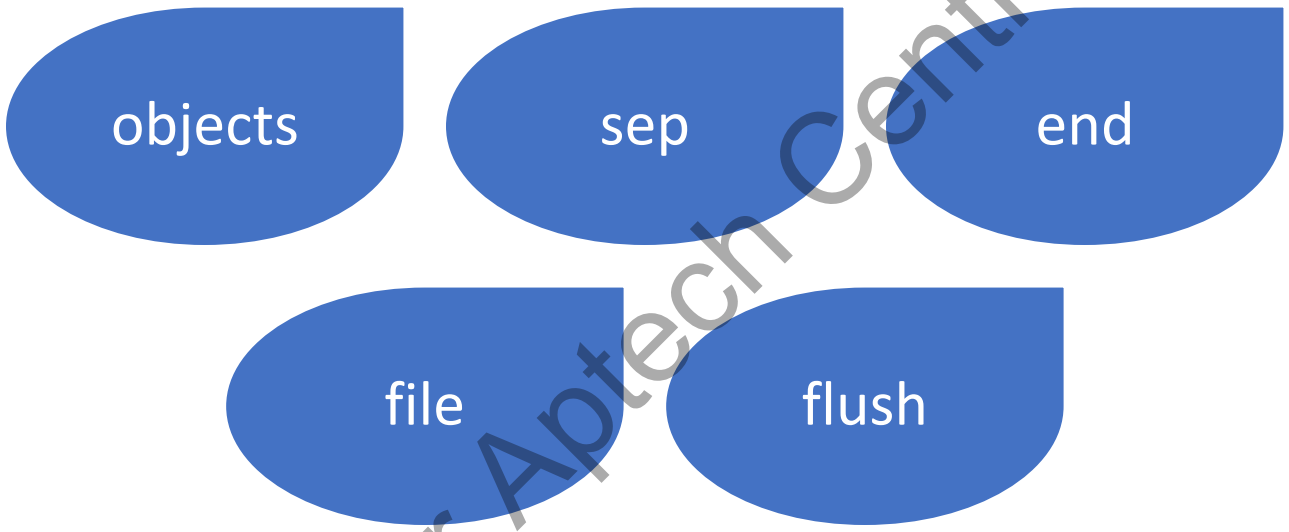
The `print()` function helps to print a given object to the standard output device.

Code Snippet:

```
print(*objects, sep=' ',
end='\n', file=sys.stdout,
flush=False)
```

Print in Python (2-2)

`print()` parameters:



Inputs in Python

`input():`

Code Snippet:

```
>>> input([prompt])
```

Here, prompt indicates user message needs to be displayed on user screen (prompt) to instruct user.

Branching in Python (1-2)

If Statement

Code Snippet:

```
if test_expression:
statement1
statement2
```

If else Statement

Code Snippet:

```
if test
expression:
Body of if
else:
Body of else
```

Branching in Python (2-2)

If…elif…else Statement

Code Snippet:

```
if test expression:
Body of if
elif test expression:
Body of elif
else:
Body of else
```

Loops in Python (1-2)

for Loop

Code Snippet:

```
for val in sequence:
    Body of for
```

range() Function

Code Snippet:

```
>>>print(range(10))
```

Output:

```
range(0, 10)
```

Loops in Python (2-2)

while Loop

Code Snippet:

```
while test_expression:
Body of while
```

while Loop Along with else

Code Snippet:

```
>>>counter = 0
>>>while counter <=3:
print("counter alive with value ",counter)
counter +=1
else:
print("counter expired")
```

Loop Control Statements (1-3)

`break` Statement

Code Snippet:

```
>>>for val in "string":
if val == "i":
break
print(val)
print("The end")
```

Loop Control Statements (2-3)

continue Statement

Code Snippet:

```
>>>while x:
x = x-1
if x%2 != 0 :
continue
else:
print (x)
```

Loop Control Statements (3-3)

pass in Loops

Code Snippet:

```
while True:
pass
```

Import in Python

Python imports are runtime operations.

Steps for Python program when first time it imports a given file:

Step 1
- Find the module's file.

Step 2
- Compile it to byte code (if needed).

Step 3
- Run the module's code to build the objects it defines.

Summary (1-3)

- Statements can be defined as the instructions that inform Python about what a program must perform.
- Expressions help to process the objects and are part of the statements.
- In Python, each statement has its own definite function and syntax.

Summary (2-3)

- In Python, all compound statements use a general pattern. The pattern consists of a header line with a colon at the end, trailed by a nested block of code generally indented after the header line.
- The `print()` function helps to print a given object to the standard output device, such as the screen or to the text stream file.
- To accommodate flexibility, Python allows users to provide inputs using the `input()` function.

Summary (3-3)

- In Python, branching statements are used to add diversions in the code flow. They are if, if…else and if…elif…else statements.
- A loop statement is used to execute a statement or a group of statements several times.
- In Python, `break` and `continue` are loop control statements and are used to alter the flow of a normal loop.