notes

September 20, 2023

1 Comments

[1]:
```
# This is a comment
'''
This is also a comment
'''

print('Hello World')
```

```
Hello World
```

2 Multiple statements in single line

[2]:
```
a = 3; print(a)
```

```
3
```

3 Multiline statements

[3]:
```
a = 1 + 3 + 4 + \
    5 + 6 + 7
print(a)
```

```
26
```

4 Multiple variable assignmeents

[4]:
```
a,b = 4,5
print(a,b)
```

```
4 5
```

5 help() & __doc__

[5]:
```
def my_func():
    '''This is a doc-string'''
    pass

help(my_func)
print('-'*100)
print(my_func.__doc__)
```

```
Help on function my_func in module __main__:

my_func()
    This is a doc-string

----------------------------------------------------------------------------------------------------
This is a doc-string
```

6 print() and input()

[1]:
```
print?
```

```
Signature: print(*args, sep=' ', end='\n', file=None, flush=False)
Docstring:
Prints the values to a stream, or to sys.stdout by default.

sep
  string inserted between values, default a space.
end
  string appended after the last value, default a newline.
file
  a file-like object (stream); defaults to the current sys.stdout.
flush
  whether to forcibly flush the stream.
Type:      builtin_function_or_method
```

[2]:
```
input?
```

```
Signature: input(prompt='')
Docstring:
Forward raw_input to frontends

Raises
------
StdinNotImplementedError if active frontend doesn't support stdin.
File:      /usr/local/lib/python3.11/site-packages/ipykernel/kernelbase.py
Type:      method
```

7 Types of statements

7.0.1 Empty Statements

- Empty statements are also known as “pass statements”
- They serve as a placeholder and do nothing when executed
- The `pass` keyword is used to create an empty statement
- Commonly used as a temporary placeholder when writing code or as a stub for future implementation

7.0.2 Simple Statements

- Simple statements are single-line statements that perform a specific action or operation
- They typically end with a newline character or a semicolon
- Common examples include assignment statements, function calls, and print statements

7.0.3 Compound Statements (Block Statements)

- Compound statements consist of one or more simple statements grouped together into a block
- They are often used to control program flow and define structures like loops and conditional statements
- Compound statements are defined using indentation (whitespace) in Python
- Common examples include if statements, while loops, for loops, and function definitions

[3]:
```
# Empty statement
def foo():
    pass  # Placeholder for function implementation


# Simple statements
x = 5  # Assignment statement
print("Hello, World!")  # Print statement
result = min(2, 3)  # Function call and assignment


# Compound statements
if x > 0:
    print("x is positive")  # Indented block as part of the if statement
else:
    print("x is not positive")  # Indented block as part of the else statement

while x < 5:
    print(x)  # Indented block as part of the while loop
    x += 1

def greet(name):
    print(f"Hello, {name}!")  # Indented block as part of the function␣
 ↪definition
```

```
Hello, World!
x is positive
```

8 Switch/Match

[4]:
```python
x = 1

match x:
    case 1: print("x is 1")
    case 2: print("x is 2")
    case 3: print("x is 3")
    case _: print("x is something else")
```

```
x is 1
```

[6]:
```python
%%writefile my_module.py
def greet():
    print('Hello World!')
```

```
Overwriting my_module.py
```