Python del Keyword

It is used to delete objects. In Python everything is an object, so the del keyword can also be used to delete variables, lists, tuples, sets..!!

Syntax
```
del var1[,var2[,var3[....,varN]]]]
```

Example:
```
a=10;del a
print(a)#NameError: name 'a' is not defined
```

Example:
```
x=1;y=2;z=3
print(x,y,z);1 2 3
del x,y,z#Remove multiple objects at a time
```

Mnemonic Variable Names

This can confuse beginning students because well-named variables often “sound” so good. There is no use in realtime industry...!!

Example:
```
x1q3z9ocd = 35.0
x1q3z9afd = 12.50
x1q3p9afd = x1q3z9ocd * x1q3z9afd
print(x1q3p9afd)
```

Example:
```
a = 3;b = 2;print(a*b)
```

Writing numbers in Binary, Octal & HexaDecimal

Number System	Prefix
Binary	0b or 0B
Octal	0o or 0O
Hexadecimal	0x or 0X

Binary literals (base 2)

Binary literals can easily be written as well. They have to be prefixed by a leading "0", followed by a "b" or "B":

Syntax:
```
bin(number)
```

Example:
```
x = 0b101010
print(x )
```

Example
```
x = bin(65)
print(x)
```

Octal literals (base 8)

A number prefixed by 0o (zero and a lowercase "o" or uppercase "O") will be interpreted as an octal number

Syntax:
```
oct(number)
```

Example
```
x = oct(65)
print(x)
```

Hexadecimal literals (base 16)
Hexadecimal literals have to be prefixed either by "0x" or "0X". (Zero followed by x or X)

Syntax:
```
hex(number)
```

Decimal ==> 0-9 (10)
Hexa ==> 6 ==> A/a =>10, B/b =>11, C/c =>12
D/d =>13, E/e =>14, F/f =>15
Hexa+Decimal=6+10=16

Example:
```
x = hex(19)
print(x)
```

Example:
```
x = hex(64)
print(x)
```

Example:Output of the following SCRIPT:
```
x=10;y=0o10;z=0X10
print(x);print(y);print(z)
a=0XAB;print(a)
```

None Data Type:
The None keyword is used to define a null value, or no value at all. If the value is not available, then to handle such type of cases None introduced.

Example:
```
PySpe=None
print(type(PySpe))#<class 'NoneType'>
print(PySpe)#None
```

Order of Operations
When an expression contains more than one operator, the order of evaluation depends on the order of operations. For mathematical operators, Python follows mathematical convention. The acronym PEMDAS is a useful way.

PEMDAS
Parentheses Exponentiation Multiplication Division Addition Subtraction

Parentheses
```
2 * (3-1)
(1+1)**(5-2)
```

Exponentiation

1 + 2**3 ==> 9, not 27,
2 * 3**2 ==> 18, not 36.

Multiplication and Division have higher precedence than Addition and Subtraction.
2*3-1 ==> 5, not 4,
6+4/2 ==> 8, not 5.

Pdb Module (Python Debugger)
pdb is a debugging tool that is part of python’s standard library. It is an interactive source code debugger for Python programs. Using pdb, we can set breakpoints at any point of our program to stop it and check for errors or the status of our running program.

Syntax:

```
import pdb;
```

PDB Options:
l (list) - Display 11 lines around the current line.
r (return) - Continue execution until the current function returns.
b (break) - Set a breakpoint (depending on the argument provided).
n (next) - Continue execution until the next line in the current function is reached.
j (jump) - Jump to the next line to be executed.
q (quit)   Quit from the debugger. The program being executed is aborted.

Open run command, enter cmd

```
C:\cd desktop
C:\desktop\ notepad Hello.py
write required Python script, save it..!!
C:\desktop\python Hello.py
C:\desktop\python -m pdb Hello.py (m=> Module=> PythonProgram)
> the prompt of the debugger
(pdb) l
List the lines of code..!!
```