In [19]:
```
import sys
sys.version
```

Out[19]:
```
'3.12.7 | packaged by Anaconda, Inc. | (main, Oct  4 2024, 13:17:27) [MSC v.1929 64 bit (AMD64)]'
```

python variable = identifier = object

syntax (variable = value)

In [3]:
```
v = 8
v
```

Out[3]:
```
8
```

RULES TO DECLARE PYHON VARIABLE

In [20]:
```
var = 8
VAR
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[20], line 2
      1 var = 8
----> 2 VAR

NameError: name 'VAR' is not defined
```

In [6]:
```
var
```

Out[6]:
```
8
```

In [7]:
```
v@ = 16
v@
```

```
  Cell In[7], line 1
    v@ = 16
     ^
SyntaxError: invalid syntax
```

In [8]:
```
v_ = 20
v_
```

Out[8]: 20

In [21]:
```
1var = 25
1var
```

```
  Cell In[21], line 1
    1var = 25
    ^
SyntaxError: invalid decimal literal
```

In [10]:
```
var1 = 8
var1
```

Out[10]: 8

In [11]:
```
'1var' = 9
```

```
  Cell In[11], line 1
    '1var' = 9
    ^
SyntaxError: cannot assign to literal here. Maybe you meant '==' instead of '='?
```

In [12]:
```
import keyword
keyword.kwlist
```

Out[12]:
```
['False',
 'None',
 'True',
 'and',
 'as',
 'assert',
 'async',
 'await',
 'break',
 'class',
 'continue',
 'def',
 'del',
 'elif',
 'else',
 'except',
 'finally',
 'for',
 'from',
 'global',
 'if',
 'import',
 'in',
 'is',
 'lambda',
 'nonlocal',
 'not',
 'or',
 'pass',
 'raise',
 'return',
 'try',
 'while',
 'with',
 'yield']
```

In [13]:
```
len(keyword.kwlist)
```

Out[13]:
```
35
```

In [14]:
```
for = 8
```

```
  Cell In[14], line 1
    for = 8
        ^
SyntaxError: invalid syntax
```

In [15]:
```
def = 79
```

```
  Cell In[15], line 1
    def = 79
        ^
SyntaxError: invalid syntax
```

In [16]:
```
DEF = 10
```

In [17]:
```
DEF
```

Out[17]: 10

PYTHON VARIABLE DECLARATION

15th

In [22]:
```
false = 56
```

In [23]:
```
false
```

Out[23]: 56

In [24]:
```
False = 56
```

```
  Cell In[24], line 1
    False = 56
    ^
SyntaxError: cannot assign to False
```

In [25]:
```
True = 8
```

```
  Cell In[25], line 1
    True = 8
    ^
SyntaxError: cannot assign to True
```

python data types

- int
- float
- bool
- string
- complex

In [27]:
```
i = 5
i
```

Out[27]:
```
5
```

In [29]:
```
type(i)
```

Out[29]:
```
int
```

In [30]:
```
f = 110.45
f
```

Out[30]:
```
110.45
```

In [31]:
```
type(f)
```

Out[31]:
```
float
```

In [34]:
```
i
```

Out[34]:
```
5
```

In [35]:
```
f
```

Out[35]: 110.45

In [39]:
```
i
f
```

Out[39]: 110.45

In [40]:
```
print(i)
print(f)
```

```
5
110.45
```

In [41]:
```
i + f
```

Out[41]: 115.45

In [42]:
```
i - f
```

Out[42]: -105.45

In [46]:
```
i * f
```

Out[46]: 552.25

bool

In [47]:
```
true
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[47], line 1
----> 1 true

NameError: name 'true' is not defined
```

In [48]:
```
True
```

Out[48]: True

In [49]:
```
False
```

Out[49]: False

In [50]:
```
True + False
```

Out[50]: 1

In [51]:
```
False + False
```

Out[51]: 0

In [52]:
```
False * True
```

Out[52]: 0

In [53]:
```
True / True
```

Out[53]: 1.0

In [54]:
```
True // True
```

Out[54]: 1

In [56]:
```
s = hello
s
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[56], line 1
----> 1 s = hello
      2 s

NameError: name 'hello' is not defined
```

In [57]:
```
s = 'hello'
s
```

Out[57]: 'hello'

In [58]:
```
s1 = "hello"
s1
```

Out[58]: 'hello'

In [59]:
```
s2 = ''' hello '''
s2
```

Out[59]: ' hello '

In [63]:
```
s3 = '''hello
           team'''
s3
```

Out[63]: 'hello \n   team'

In [64]:
```
c = 10 + 20j
c
```

Out[64]: (10+20j)

In [66]:
```
c.real
```

Out[66]: 10.0

In [67]:
```
c.imag
```

Out[67]: 20.0

In [68]:
```
c = 10 + 20j
d = 20 + 30j
```

In [69]:
```
print(c+d)
```

(30+50j)

In [70]:
```
print(c-d)
```

```
(-10-10j)
```

In [72]:
```
print(c*d)
```

```
(-400+700j)
```

In [76]:
```
c2 = 1 + 2J
```

Python variable completed

Python datat type completed

16th

- python intro
- python interpreter, compiler, ide
- which ide we installed
- how to check python version
- name is caled variable (variable == object == identifer)
- python dataype
- demo (who learned, what topic, placement, class , duraion, project, capstone, certificat, roadmap,
- course syllabus)
- introduce to funct & fun argument

Type Casting == Convert one Dataype to Other Dataype

In [1]:
```
int(2.4)
```

Out[1]:
```
2
```

In [2]:
```
int(2.4, 3.5)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[2], line 1
----> 1 int(2.4, 3.5)

TypeError: 'float' object cannot be interpreted as an integer
```

In [3]:
```
int(True)
```

Out[3]: 1

In [4]:
```
int(false)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[4], line 1
----> 1 int(false)

NameError: name 'false' is not defined
```

In [5]:
```
int(False)
```

Out[5]: 0

In [6]:
```
int(True, False)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[6], line 1
----> 1 int(True, False)

TypeError: int() can't convert non-string with explicit base
```

In [7]:
```
int('10')
```

Out[7]: 10

In [8]:
```
int('ten')
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[8], line 1
----> 1 int('ten')

ValueError: invalid literal for int() with base 10: 'ten'
```

In [9]:
```
int(1+2j)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[9], line 1
----> 1 int(1+2j)

TypeError: int() argument must be a string, a bytes-like object or a real number, not 'complex'
```

In [ ]:
```
# all other datatype to flaot
```

In [10]:
```
float(200000)
```

Out[10]:
```
200000.0
```

In [12]:
```
float(10, 20)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[12], line 1
----> 1 float(10, 20)

TypeError: float expected at most 1 argument, got 2
```

In [11]:
```
float(1+2j)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[11], line 1
----> 1 float(1+2j)

TypeError: float() argument must be a string or a real number, not 'complex'
```

In [13]:
```
float(True)
```

Out[13]: 1.0

In [14]:
```
float(False)
```

Out[14]: 0.0

In [15]:
```
float('10')
```

Out[15]: 10.0

In [16]:
```
float('ten')
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[16], line 1
----> 1 float('ten')

ValueError: could not convert string to float: 'ten'
```

In [17]:
```
str(8)
```

Out[17]: '8'

In [19]:
```
str(8.8)
```

Out[19]: '8.8'

In [20]:
```
str(1+2j)
```

Out[20]: '(1+2j)'

In [23]:
```
str(True, False)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[23], line 1
----> 1 str(True, False)

TypeError: str() argument 'encoding' must be str, not bool
```

In [22]:
```
str()
```

Out[22]: ''

In [24]:
```
bool(10)
```

True

In [25]:
```
bool(1.5)
```

Out[25]: True

In [26]:
```
bool()
```

Out[26]: False

In [27]:
```
bool('10')
```

Out[27]: True

In [28]:
```
bool('ten')
```

Out[28]: True

In [29]:
```
bool(0)
```

Out[29]: False

In [30]:
```
bool(1+2j)
```

Out[30]: True

In [31]:
```
complex(10)
```

Out[31]: (10+0j)

In [32]:
```
complex(10,20)
```

Out[32]: (10+20j)

In [33]:
```
complex(10,20,30)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[33], line 1
----> 1 complex(10,20,30)

TypeError: complex() takes at most 2 arguments (3 given)
```

In [35]:
```
complex(10.20,  30.8)
```

Out[35]: (10.2+30.8j)

In [36]:
```
complex(True)
```

Out[36]: (1+0j)

In [37]:
```
complex(False)
```

Out[37]: 0j

In [38]:
```
complex('10')
```

Out[38]: (10+0j)

In [39]:
```
complex('10', '20')
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[39], line 1
----> 1 complex('10', '20')

TypeError: complex() can't take second arg if first is a string
```

In [40]:
```
complex(True, False)
```

Out[40]: (1+0j)

e -> exponential

In [1]:
```
f1 = 2e1
f1
```

Out[1]: 20.0

In [5]:
```
f2 = 2.4e2
f2
```

Out[5]: 240.0

In [7]:
```
f3 = 2.5e3
f3
```

Out[7]: 2500.0

In [10]:
```
import numpy as np

a = np.nan
a
```

Out[10]: nan

In [11]:
```
i = 34
i
```

Out[11]: 34

In [12]:
```
id(i)
```

Out[12]: 140712069609416

If you assign same value to different variable then the memory address is same

In [13]:
```
p = 20
q = 20
r = 20
```

In [14]:
```
print(id(p), id(q), id(r))
```

```
140712069608968 140712069608968 140712069608968
```

Indexing

In [15]:
```
str = "hello"
str
```

Out[15]:
```
'hello'
```

forward

In [21]:
```
print(str[0])
print(str[1])
print(str[2])
print(str[3])
print(str[4])
```

```
h
e
l
l
o
```

Reverse

In [23]:
```
print(str[-1])
print(str[-2])
print(str[-3])
print(str[-4])
print(str[-5])
```

```
o
l
l
e
h
```

slicing

In [24]:
```
str
```

Out[24]: 'hello'

In [25]:
```
str[1:3]
```

Out[25]: 'el'

In [28]:
```
str[0:4] # str[m: (n-1)]
```

Out[28]: 'hell'

In [29]:
```
s = 'hellopython'
s
```

Out[29]: 'hellopython'

In [30]:
```
s[0:6]
```

Out[30]: 'hellop'

In [31]:
```
s[0:10:3]
```

Out[31]: 'hlyo'

In [32]:
```
s[1:8:2]
```

Out[32]: 'elpt'

BackwordIndexing

In [33]:
```
s
```

Out[33]: 'hellopython'

In [35]:
```
s[::2]
```

Out[35]: 'hloyhn'

In [36]:
```
s[::3]
```

Out[36]: 'hlyo'

String Concactination

In [38]:
```
s3 = 'hello'
s4 = ' hi'

print(s3 + s4)
```

```
hello hi
```

Python Operator

In [2]:
```
x1, y1 =  10, 5
```

In [5]:
```
a, b = 6 # You can't assign like this
         # No. of values is equal to the number of variable
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[5], line 1
----> 1 a, b = 6

TypeError: cannot unpack non-iterable int object
```

In [6]:
```
x1 + y1
```

Out[6]: 15

In [7]:
```
x1 / y1 # float division
```

Out[7]:
```
2.0
```

In [8]:
```
x1 // y1 # int division
```

Out[8]:
```
2
```

In [11]:
```
x1 % y1# Modulo
```

Out[11]:
```
0
```

In [12]:
```
x1 ** y1 # Square
```

Out[12]:
```
100000
```

Assignment operator

+= , -=, *=, /=, //=

In [13]:
```
x = 2
```

In [14]:
```
x =  x+2
```

In [15]:
```
x
```

Out[15]:
```
4
```

In [17]:
```
x += 2
x
```

Out[17]:
```
8
```

Unary operatory

- Unary means 1 || binary means 2

- Here we are applying unary minus operator(-) to the variable

In [18]:
```
n = 7
```

In [19]:
```
-n
```

Out[19]:
```
-7
```

Relational operator

we are using this operatory for comparing

<, >, <=, >=, ==, !=

In [20]:
```
a = 5
b = 6
```

In [21]:
```
a < b
```

Out[21]:
```
True
```

In [22]:
```
b> a
```

Out[22]:
```
True
```

Logical Operator

AND, OR, NOT

AND

In [23]:
```
a, b = 5, 4
```

In [24]:
```
a < 8 and b< 5
```

Out[24]: True

In [25]:
```
a < 8 and b< 2
```

Out[25]: False

OR

In [26]:
```
a< 8 or b< 2
```

Out[26]: True

In [27]:
```
a > 8 or b>5
```

Out[27]: False

NOT

In [29]:
```
x = False
x
```

Out[29]: False

In [30]:
```
not x
```

Out[30]: True

Number Systems

- Binary Number Systems (0b) -> base 2 (0, 1)
- Octal Number Systems (0o) -> base 8 (0, 1, 2, 3, 4, 5, 6, 7)
- Decimal Number System (0x) -> base 10 (0, 1, 2, 3, 4, 5, 6, 7, 8, 9)
- Hexadecimal Number System(0xa, b, c, d, e, f) -> base 16 (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, a(10), b(11), c(12), d(13), e(14), f(15))

Binary Number System

In [33]:
```
25
```

Out[33]: 25

In [2]:
```
bin(25) # binary
```

Out[2]: '0b11001'

In [36]:
```
0b11001 # Here 0b means it implies Binary number system (0b) and other half is binary number convert
```

Out[36]: 25

In [3]:
```
bin(30)
```

Out[3]: '0b11110'

In [7]:
```
int(0b11101)
```

Out[7]: 29

Octal

In [8]:
```
0o31
```

Out[8]: 25

In [9]:
```
oct(25)
```

Out[9]: '0o31'

Hexagonal

In [10]:
```
0xa
```

Out[10]: 10

In [11]:
```
hex(10)
```

Out[11]: '0xa'

BITWISE OPERATOR

- Complement Operator
- AND Operator
- OR Operator
- XOR operator
- LEFT Shift Operator
- Right Shift Operator

Complement Operator (~)

In [31]:
```
~12 # Complement means it stores -ve values
```

Out[31]: -13

In [2]:
```
~1007
```

Out[2]: -1008

In [1]:
```
~ 78
```

Out[1]: -79

In [32]:
```
~46
```

Out[32]: -47

AND Operator (&)

In [13]:
```
12 & 13
```

Out[13]: 12

In [16]:
```
12 | 13
```

Out[16]: 13

In [14]:
```
35 & 69
```

Out[14]: 1

In [17]:
```
print(bin(35), bin(69))
```

```
0b100011 0b1000101
```

XOR (^)

In [18]:
```
12^13
```

Out[18]: 1

In [19]:
```
print(bin(25))
print(bin(30))
```

```
0b11001
0b11110
```

In [20]:
```
25^30
```

Out[20]: 7

In [21]:
```
print(bin(25), bin(30))
```

```
0b11001 0b11110
```

In [23]:
```
print(int(0b00111))
```

```
7
```

Left Shift (<<)

```
In [25]: 10 << 1
```

```
Out[25]: 20
```

```
In [27]: print(0b10100)
```

```
20
```

```
In [28]: 10 << 2
```

```
Out[28]: 40
```

```
In [30]: print(0b101000)
```

```
40
```

Right Shift (>>)

```
In [31]: 10 >> 1
```

```
Out[31]: 5
```

```
In [32]: print(0b101)
```

```
5
```

```
In [33]: 10 >> 2
```

```
Out[33]: 2
```

```
In [34]: 10 >> 3
```

```
Out[34]: 1
```

Print() in Python

In print function we can add 1 or more argument

In [36]:
```
print(10)
print(10, 20)
print('python')
print(10, 20, 'python', 1+2j, True, [1, 2])
```

```
10
10 20
python
10 20 python (1+2j) True [1, 2]
```

Print() result with string

In [37]:
```
num1 = 20
num2 = 30
add = num1+num2
print(add)
```

```
50
```

In [39]:
```
num1 = 20
num2 = 30
add =  num1+ num2
print('The addition of', num1, 'and', num2, 'is=', add)
```

```
The addition of 20 and 30 is= 50
```

In [42]:
```
name = 'python'
age = 20
city = 'hyd'
print("My name is",name, "and i am", age, "years old from", "city")
```

```
My name is python and i am 20 years old from city
```

Print format method

1. Use .format() method

2. Number of format == Number of {}

In [41]:
```
num1, num2 = 20, 30
add = num1 + num2

print('The addition of {} and {} is = {}'.format(num1, num2, add))
```

```
The addition of 20 and 30 is = 50
```

f string method

- Variable should be in urly braces
- and write inside quotes ''
- add f at the start

In [43]:
```
num1, num2 = 20, 30
add = num1 + num2

print(f"The addition of {num1} and {num2} is = {add}")
```

```
The addition of 20 and 30 is = 50
```

End Statement

In [44]:
```
print('hello')
print('good morning')
```

```
hello
good morning
```

In [45]:
```
print('hello', end = ' ')
print("good morning", end='^^^')
```

```
hello good morning^^^
```

`separator (sep=)`

In [46]:
```
print('hello', 'hai', 'how are you', sep = ' *** ')
```

```
hello *** hai *** how are you
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: