In [6]:
```
a=10
b=20
c=a+b
print(c)
```

```
30
```

Today is Tuesday

Tuesday

Wednesday

Identifiers

-A name in python program is called identifier

-class name,function name,module name,variable name

eg.a=10

a<- Identifier

def test

class car

1.The only allowed characters in python are

-alphabet symbol(a-z)

-digits(0-9)

-Underscore symbol(_)

cash=100

ca$h=100 : this will give syntax error

2.Identifier should not start with digits

total123=100 : this is valid

123total=100 : this will give syntax error

3.Identifiers are case sensitive

eg. total=100 Total=100 TOTAL=100

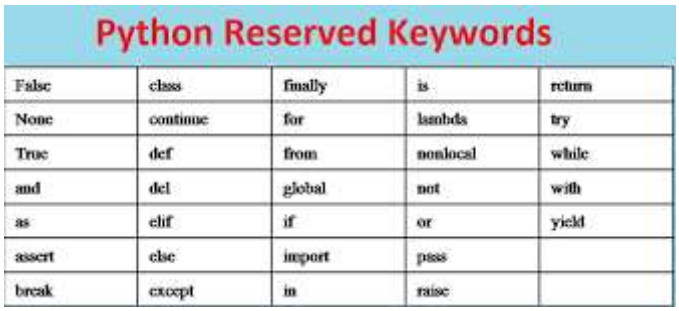
The all variables above will be considered different

4.We can not use reserved keyword as identifiers

def=10 : This will give syntax error as def is reserved keyword

DEF=10 : This is valid

Following are some keyword



Python Reserved Keywords				
False	class	finally	is	return
None	continue	for	lambda	try
True	def	from	nonlocal	while
and	del	global	not	with
as	elif	if	or	yield
assert	else	import	pass	
break	except	in	raise	

5.There is no length limit for python identifier but not recommended to use too lengthy identifiers

6.if identifiers starts with Underscore(_) then it is private

eg.__a=10

_a=10

7.if identifiers starts with double underscore(__) then it is strongly private identifier

eg.__a=10

8.if identifier starts and ends with double underscore symbols then the identifier is language defined special name ,which is also known as magic methods.

eg.**add**(+)

Datatypes

Text type- Str

Numeric type-int,float

sequence type-list,tuple,range

mapping type-dict

set types- set,frozenset

boolean type-bool

In [22]:
```
# *str
s1='Arman'

s1="Arman"
```

```
s1="""Armaan python"""

a="python"

print (a)
print(type(a))
```

```
python
<class 'str'>
```

In [23]:
```
# 1.)int
x=1
y=3456789
z=-3230584
print(type(x))
```

```
<class 'int'>
```

In [24]:
```
# *binary
a=0b1111 # or 0B1111
print(a)
```

```
15
```

b=0b123

print(b):This gives syntax error as it is not a binary number

In [26]:
```
# *octal
x=0o112
print(x)
```

```
74
```

In [27]:
```
x=0o1111
print(x)
```

```
585
```

In [28]:
```
# *hex
p=0xABCD #or 0XABCD
print(p)
```

```
43981
```

Base conversion

1.bin()

In [37]:
```
print(bin(15))
print(bin(0o11))
print(bin(0x12))
bin(15)
```

```
0b1111
0b1001
0b10010
```

Out[37]:
```
'0b1111'
```

In [33]:
```
print(bin(15))
```

```
0b1111
```

2.oct()

3.hex()

In [39]:
```
print(oct(15))
```

```
0o17
```

In [ ]:
```

```