In [ ]:
```
class,object,Method,Constructor,special methods
```

In [2]:
```
class prodA:
    pid=101
    pcost=1222.22

class myclass:
    pcount=10

obj=myclass()
obj.pcount
obj.pid
obj.pcost
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-2-45814b3784e3> in <module>
      8 obj=myclass()
      9 obj.pcount
---> 10 obj.pid
     11 obj.pcost

AttributeError: 'myclass' object has no attribute 'pid'
```

In [ ]:
```
Inheritance---> reusuability

class parentClass:
    <attr1>
    <attr2>

class Childclass(<ParentClass>):
    <attr3>
    <attr4>
```

In [3]:
```
help(str)
```

```
Help on class str in module builtins:

class str(object)
 |  str(object='') -> str
 |  str(bytes_or_buffer[, encoding[, errors]]) -> str
 |
 |  Create a new string object from the given object. If encoding or
 |  errors is specified, then the object must expose a data buffer
 |  that will be decoded using the given encoding and error handler.
 |  Otherwise, returns the result of object.__str__() (if defined)
 |  or repr(object).
 |  encoding defaults to sys.getdefaultencoding().
 |  errors defaults to 'strict'.
 |
 |  Methods defined here:
 |
 |  __add__(self, value, /)
 |      Return self+value.
 |
 |  __contains__(self, key, /)
 |      Return key in self.
 |
 |  __eq__(self, value, /)
 |      Return self==value.
 |
```

```
|  __format__(self, format_spec, /)
|      Return a formatted version of the string as described by format_spec.
|
|  __ge__(self, value, /)
|      Return self>=value.
|
|  __getattribute__(self, name, /)
|      Return getattr(self, name).
|
|  __getitem__(self, key, /)
|      Return self[key].
|
|  __getnewargs__(...)
|
|  __gt__(self, value, /)
|      Return self>value.
|
|  __hash__(self, /)
|      Return hash(self).
|
|  __iter__(self, /)
|      Implement iter(self).
|
|  __le__(self, value, /)
|      Return self<=value.
|
|  __len__(self, /)
|      Return len(self).
|
|  __lt__(self, value, /)
|      Return self<value.
|
|  __mod__(self, value, /)
|      Return self%value.
|
|  __mul__(self, value, /)
|      Return self*value.
|
|  __ne__(self, value, /)
|      Return self!=value.
|
|  __repr__(self, /)
|      Return repr(self).
|
|  __rmod__(self, value, /)
|      Return value%self.
|
|  __rmul__(self, value, /)
|      Return value*self.
|
|  __sizeof__(self, /)
|      Return the size of the string in memory, in bytes.
|
|  __str__(self, /)
|      Return str(self).
|
|  capitalize(self, /)
|      Return a capitalized version of the string.
|
|      More specifically, make the first character have upper case and the rest lower
|      case.
|
|  casefold(self, /)
|      Return a version of the string suitable for caseless comparisons.
|
```

```
center(self, width, fillchar=' ', /)
    Return a centered string of length width.

    Padding is done using the specified fill character (default is a space).

count(...)
    S.count(sub[, start[, end]]) -> int

    Return the number of non-overlapping occurrences of substring sub in
    string S[start:end].  Optional arguments start and end are
    interpreted as in slice notation.

encode(self, /, encoding='utf-8', errors='strict')
    Encode the string using the codec registered for encoding.

    encoding
      The encoding in which to encode the string.
    errors
      The error handling scheme to use for encoding errors.
      The default is 'strict' meaning that encoding errors raise a
      UnicodeEncodeError.  Other possible values are 'ignore', 'replace' and
      'xmlcharrefreplace' as well as any other name registered with
      codecs.register_error that can handle UnicodeEncodeErrors.

endswith(...)
    S.endswith(suffix[, start[, end]]) -> bool

    Return True if S ends with the specified suffix, False otherwise.
    With optional start, test S beginning at that position.
    With optional end, stop comparing S at that position.
    suffix can also be a tuple of strings to try.

expandtabs(self, /, tabsize=8)
    Return a copy where all tab characters are expanded using spaces.

    If tabsize is not given, a tab size of 8 characters is assumed.

find(...)
    S.find(sub[, start[, end]]) -> int

    Return the lowest index in S where substring sub is found,
    such that sub is contained within S[start:end].  Optional
    arguments start and end are interpreted as in slice notation.

    Return -1 on failure.

format(...)
    S.format(*args, **kwargs) -> str

    Return a formatted version of S, using substitutions from args and kwargs.
    The substitutions are identified by braces ('{' and '}').

format_map(...)
    S.format_map(mapping) -> str

    Return a formatted version of S, using substitutions from mapping.
    The substitutions are identified by braces ('{' and '}').

index(...)
    S.index(sub[, start[, end]]) -> int

    Return the lowest index in S where substring sub is found,
    such that sub is contained within S[start:end].  Optional
    arguments start and end are interpreted as in slice notation.
```

```
|      Raises ValueError when the substring is not found.
|
|  isalnum(self, /)
|      Return True if the string is an alpha-numeric string, False otherwise.
|
|      A string is alpha-numeric if all characters in the string are alpha-numeric and
|      there is at least one character in the string.
|
|  isalpha(self, /)
|      Return True if the string is an alphabetic string, False otherwise.
|
|      A string is alphabetic if all characters in the string are alphabetic and there
|      is at least one character in the string.
|
|  isascii(self, /)
|      Return True if all characters in the string are ASCII, False otherwise.
|
|      ASCII characters have code points in the range U+0000-U+007F.
|      Empty string is ASCII too.
|
|  isdecimal(self, /)
|      Return True if the string is a decimal string, False otherwise.
|
|      A string is a decimal string if all characters in the string are decimal and
|      there is at least one character in the string.
|
|  isdigit(self, /)
|      Return True if the string is a digit string, False otherwise.
|
|      A string is a digit string if all characters in the string are digits and there
|      is at least one character in the string.
|
|  isidentifier(self, /)
|      Return True if the string is a valid Python identifier, False otherwise.
|
|      Call keyword.iskeyword(s) to test whether string s is a reserved identifier,
|      such as "def" or "class".
|
|  islower(self, /)
|      Return True if the string is a lowercase string, False otherwise.
|
|      A string is lowercase if all cased characters in the string are lowercase and
|      there is at least one cased character in the string.
|
|  isnumeric(self, /)
|      Return True if the string is a numeric string, False otherwise.
|
|      A string is numeric if all characters in the string are numeric and there is at
|      least one character in the string.
|
|  isprintable(self, /)
|      Return True if the string is printable, False otherwise.
|
|      A string is printable if all of its characters are considered printable in
|      repr() or if it is empty.
|
|  isspace(self, /)
|      Return True if the string is a whitespace string, False otherwise.
|
|      A string is whitespace if all characters in the string are whitespace and there
|      is at least one character in the string.
|
|  istitle(self, /)
|      Return True if the string is a title-cased string, False otherwise.
|
```

```
|      In a title-cased string, upper- and title-case characters may only
|      follow uncased characters and lowercase characters only cased ones.
|
|  isupper(self, /)
|      Return True if the string is an uppercase string, False otherwise.
|
|      A string is uppercase if all cased characters in the string are uppercase and
|      there is at least one cased character in the string.
|
|  join(self, iterable, /)
|      Concatenate any number of strings.
|
|      The string whose method is called is inserted in between each given string.
|      The result is returned as a new string.
|
|      Example: '.'.join(['ab', 'pq', 'rs']) -> 'ab.pq.rs'
|
|  ljust(self, width, fillchar=' ', /)
|      Return a left-justified string of length width.
|
|      Padding is done using the specified fill character (default is a space).
|
|  lower(self, /)
|      Return a copy of the string converted to lowercase.
|
|  lstrip(self, chars=None, /)
|      Return a copy of the string with leading whitespace removed.
|
|      If chars is given and not None, remove characters in chars instead.
|
|  partition(self, sep, /)
|      Partition the string into three parts using the given separator.
|
|      This will search for the separator in the string.  If the separator is found,
|      returns a 3-tuple containing the part before the separator, the separator
|      itself, and the part after it.
|
|      If the separator is not found, returns a 3-tuple containing the original string
|      and two empty strings.
|
|  replace(self, old, new, count=-1, /)
|      Return a copy with all occurrences of substring old replaced by new.
|
|        count
|          Maximum number of occurrences to replace.
|          -1 (the default value) means replace all occurrences.
|
|      If the optional argument count is given, only the first count occurrences are
|      replaced.
|
|  rfind(...)
|      S.rfind(sub[, start[, end]]) -> int
|
|      Return the highest index in S where substring sub is found,
|      such that sub is contained within S[start:end].  Optional
|      arguments start and end are interpreted as in slice notation.
|
|      Return -1 on failure.
|
|  rindex(...)
|      S.rindex(sub[, start[, end]]) -> int
|
|      Return the highest index in S where substring sub is found,
|      such that sub is contained within S[start:end].  Optional
|      arguments start and end are interpreted as in slice notation.
```

```
|
|      Raises ValueError when the substring is not found.
|
|  rjust(self, width, fillchar=' ', /)
|      Return a right-justified string of length width.
|
|      Padding is done using the specified fill character (default is a space).
|
|  rpartition(self, sep, /)
|      Partition the string into three parts using the given separator.
|
|      This will search for the separator in the string, starting at the end. If
|      the separator is found, returns a 3-tuple containing the part before the
|      separator, the separator itself, and the part after it.
|
|      If the separator is not found, returns a 3-tuple containing two empty strings
|      and the original string.
|
|  rsplit(self, /, sep=None, maxsplit=-1)
|      Return a list of the words in the string, using sep as the delimiter string.
|
|        sep
|          The delimiter according which to split the string.
|          None (the default value) means split according to any whitespace,
|          and discard empty strings from the result.
|        maxsplit
|          Maximum number of splits to do.
|          -1 (the default value) means no limit.
|
|      Splits are done starting at the end of the string and working to the front.
|
|  rstrip(self, chars=None, /)
|      Return a copy of the string with trailing whitespace removed.
|
|      If chars is given and not None, remove characters in chars instead.
|
|  split(self, /, sep=None, maxsplit=-1)
|      Return a list of the words in the string, using sep as the delimiter string.
|
|      sep
|        The delimiter according which to split the string.
|        None (the default value) means split according to any whitespace,
|        and discard empty strings from the result.
|      maxsplit
|        Maximum number of splits to do.
|        -1 (the default value) means no limit.
|
|  splitlines(self, /, keepends=False)
|      Return a list of the lines in the string, breaking at line boundaries.
|
|      Line breaks are not included in the resulting list unless keepends is given and
|      true.
|
|  startswith(...)
|      S.startswith(prefix[, start[, end]]) -> bool
|
|      Return True if S starts with the specified prefix, False otherwise.
|      With optional start, test S beginning at that position.
|      With optional end, stop comparing S at that position.
|      prefix can also be a tuple of strings to try.
|
|  strip(self, chars=None, /)
|      Return a copy of the string with leading and trailing whitespace removed.
|
|      If chars is given and not None, remove characters in chars instead.
```

```
|
|  swapcase(self, /)
|      Convert uppercase characters to lowercase and lowercase characters to uppercase.
|
|  title(self, /)
|      Return a version of the string where each word is titlecased.
|
|      More specifically, words start with uppercased characters and all remaining
|      cased characters have lower case.
|
|  translate(self, table, /)
|      Replace each character in the string using the given translation table.
|
|        table
|          Translation table, which must be a mapping of Unicode ordinals to
|          Unicode ordinals, strings, or None.
|
|      The table must implement lookup/indexing via __getitem__, for instance a
|      dictionary or list.  If this operation raises LookupError, the character is
|      left untouched.  Characters mapped to None are deleted.
|
|  upper(self, /)
|      Return a copy of the string converted to uppercase.
|
|  zfill(self, width, /)
|      Pad a numeric string with zeros on the left, to fill a field of the given width.
|
|      The string is never truncated.
|
|  ----------------------------------------------------------------------
|  Static methods defined here:
|
|  __new__(*args, **kwargs) from builtins.type
|      Create and return a new object.  See help(type) for accurate signature.
|
|  maketrans(...)
|      Return a translation table usable for str.translate().
|
|      If there is only one argument, it must be a dictionary mapping Unicode
|      ordinals (integers) or characters to Unicode ordinals, strings or None.
|      Character keys will be then converted to ordinals.
|      If there are two arguments, they must be strings of equal length, and
|      in the resulting dictionary, each character in x will be mapped to the
|      character at the same position in y. If there is a third argument, it
|      must be a string, whose characters will be mapped to None in the result.
```

In [ ]:
```
obj=Childclass()
obj.attr1-----|
obj.attr2-----|-------parentClass
obj.attr3
obj.attr4
```

In [4]:
```
class prodA:
    pid=111
    pcost=2232.11

class myclass(prodA):#Inheritance
    pcount=10

obj=myclass()  # chid class Object
print(obj.pcount)
```

```
print(obj.pid)
```

```
10
111
2232.11
```

In [5]:

```
class A:
    var=10
    def f1(self):
        print("Inside f1")


class B(A): # B inherits A
    name="Some"
    def f2(self):
        print("Inside f2")

obj = B()
print(obj.var)# A Attribute
print(obj.name) # B Attribute
obj.f1()# A class method
obj.f2()#B Class Method
```

```
10
Some
Inside f1
Inside f2
```

In [ ]:

```
A
|
B
Single Inheritance
====================================
p1 p2  p3
|__|___|
   |
   C
Multiple Inheritance
class p1:
    pass
class p2:
    pass
class p3:
    pass
class c(p1,p2,p3):
    pass

obj=c()
=============================================
p1
|
p2
|
p3
|
C
object for can access p1 attr, p2 attr and p3 attr
```

In [13]:

```
import datetime
print(type(datetime.datetime))
obj=datetime.datetime.now()
```

```
print(obj)
print(obj.year)
help(datetime.datetime)
```

```
<class 'type'>
2023-01-20 14:41:46.090675
2023
Help on class datetime in module datetime:

class datetime(date)
 |  datetime(year, month, day[, hour[, minute[, second[, microsecond[,tzinfo]]]]])
 |
 |  The year, month and day arguments are required. tzinfo may be None, or an
 |  instance of a tzinfo subclass. The remaining arguments may be ints.
 |
 |  Method resolution order:
 |      datetime
 |      date
 |      builtins.object
 |
 |  Methods defined here:
 |
 |  __add__(self, value, /)
 |      Return self+value.
 |
 |  __eq__(self, value, /)
 |      Return self==value.
 |
 |  __ge__(self, value, /)
 |      Return self>=value.
 |
 |  __getattribute__(self, name, /)
 |      Return getattr(self, name).
 |
 |  __gt__(self, value, /)
 |      Return self>value.
 |
 |  __hash__(self, /)
 |      Return hash(self).
 |
 |  __le__(self, value, /)
 |      Return self<=value.
 |
 |  __lt__(self, value, /)
 |      Return self<value.
 |
 |  __ne__(self, value, /)
 |      Return self!=value.
 |
 |  __radd__(self, value, /)
 |      Return value+self.
 |
 |  __reduce__(...)
 |      __reduce__() -> (cls, state)
 |
 |  __reduce_ex__(...)
 |      __reduce_ex__(proto) -> (cls, state)
 |
 |  __repr__(self, /)
 |      Return repr(self).
 |
 |  __rsub__(self, value, /)
 |      Return value-self.
 |
 |  __str__(self, /)
```

```
 |      Return str(self).
 |
 |  __sub__(self, value, /)
 |      Return self-value.
 |
 |  astimezone(...)
 |      tz -> convert to local time in new timezone tz
 |
 |  ctime(...)
 |      Return ctime() style string.
 |
 |  date(...)
 |      Return date object with same year, month and day.
 |
 |  dst(...)
 |      Return self.tzinfo.dst(self).
 |
 |  isoformat(...)
 |      [sep] -> string in ISO 8601 format, YYYY-MM-DDT[HH[:MM[:SS[.mmm[uuu]]]]][+HH:M
M].
 |      sep is used to separate the year from the time, and defaults to 'T'.
 |      timespec specifies what components of the time to include (allowed values are 'a
uto', 'hours', 'minutes', 'seconds', 'milliseconds', and 'microseconds').
 |
 |  replace(...)
 |      Return datetime with new specified fields.
 |
 |  time(...)
 |      Return time object with same time but with tzinfo=None.
 |
 |  timestamp(...)
 |      Return POSIX timestamp as float.
 |
 |  timetuple(...)
 |      Return time tuple, compatible with time.localtime().
 |
 |  timetz(...)
 |      Return time object with same time and tzinfo.
 |
 |  tzname(...)
 |      Return self.tzinfo.tzname(self).
 |
 |  utcoffset(...)
 |      Return self.tzinfo.utcoffset(self).
 |
 |  utctimetuple(...)
 |      Return UTC time tuple, compatible with time.localtime().
 |
 |  ----------------------------------------------------------------------
 |  Class methods defined here:
 |
 |  combine(...) from builtins.type
 |      date, time -> datetime with same date and time fields
 |
 |  fromisoformat(...) from builtins.type
 |      string -> datetime from datetime.isoformat() output
 |
 |  fromtimestamp(...) from builtins.type
 |      timestamp[, tz] -> tz's local time from POSIX timestamp.
 |
 |  now(tz=None) from builtins.type
 |      Returns new datetime object representing current time local to tz.
 |
 |        tz
 |          Timezone object.
```

```
|
|       If no tz is specified, uses local timezone.
|
|   strptime(...) from builtins.type
|       string, format -> new datetime parsed from a string (like time.strptime()).
|
|   utcfromtimestamp(...) from builtins.type
|       Construct a naive UTC datetime from a POSIX timestamp.
|
|   utcnow(...) from builtins.type
|       Return a new datetime representing UTC day and time.
|
|   ----------------------------------------------------------------------
|   Static methods defined here:
|
|   __new__(*args, **kwargs) from builtins.type
|       Create and return a new object.  See help(type) for accurate signature.
|
|   ----------------------------------------------------------------------
|   Data descriptors defined here:
|
|   fold
|
|   hour
|
|   microsecond
|
|   minute
|
|   second
|
|   tzinfo
|
|   ----------------------------------------------------------------------
|   Data and other attributes defined here:
|
|   max = datetime.datetime(9999, 12, 31, 23, 59, 59, 999999)
|
|   min = datetime.datetime(1, 1, 1, 0, 0)
|
|   resolution = datetime.timedelta(microseconds=1)
|
|   ----------------------------------------------------------------------
|   Methods inherited from date:
|
|   __format__(...)
|       Formats self with strftime.
|
|   isocalendar(...)
|       Return a 3-tuple containing ISO year, week number, and weekday.
|
|   isoweekday(...)
|       Return the day of the week represented by the date.
|       Monday == 1 ... Sunday == 7
|
|   strftime(...)
|       format -> strftime() style string.
|
|   toordinal(...)
|       Return proleptic Gregorian ordinal.  January 1 of year 1 is day 1.
|
|   weekday(...)
|       Return the day of the week represented by the date.
|       Monday == 0 ... Sunday == 6
|
```

```
|  ----------------------------------------------------------------------
|  Class methods inherited from date:
|
|  fromisocalendar(...) from builtins.type
|      int, int, int -> Construct a date from the ISO year, week number and weekday.
|
|      This is the inverse of the date.isocalendar() function
|
|  fromordinal(...) from builtins.type
|      int -> date corresponding to a proleptic Gregorian ordinal.
|
|  today(...) from builtins.type
|      Current date or datetime:  same as self.__class__.fromtimestamp(time.time()).
|
|  ----------------------------------------------------------------------
|  Data descriptors inherited from date:
|
|  day
|
|  month
|
|  year
```

In [14]:
```
obj.strftime("%D")

#date+%D=>MM/DD/YYYY
```

Out[14]:
```
'01/20/23'
```

In [17]:
```
import time
print(time.ctime())

print(datetime.datetime.today())
```

```
Fri Jan 20 14:46:11 2023
2023-01-20 14:46:11.258842
```

In [19]:
```
import sqlite3
dbh=sqlite3.connect("D:\\test.db")
```

In [21]:
```
sth=dbh.cursor()
sth.execute("create table employee(id INT, name TEXT)")
```

Out[21]:
```
<sqlite3.Cursor at 0x85ae8f0>
```

In [23]:
```
sth.execute("insert into employee values(101,'Akshay')")
```

Out[23]:
```
<sqlite3.Cursor at 0x85ae8f0>
```

In [24]:
```
sth.execute("insert into employee values(155,'Karun')")
```

Out[24]:
```
<sqlite3.Cursor at 0x85ae8f0>
```

In [25]:
```
sth.execute("select * from employee")
```

Out[25]:
```
<sqlite3.Cursor at 0x85ae8f0>
```

In [26]:
```
sth.fetchone()
```

Out[26]:
```
(101, 'Akshay')
```

In [27]:
```
sth.fetchone()#tuple
```

Out[27]:
```
(155, 'Karun')
```

In [29]:
```
sth.execute("select * from employee")
sth.fetchall()#List of Tuple
```

Out[29]:
```
[(101, 'Akshay'), (155, 'Karun')]
```

In [30]:
```
sth.execute("select * from employee")
for i in sth:
    print(i)
```

```
(101, 'Akshay')
(155, 'Karun')
```

In [ ]: