Assignment 9 - Python

Math Function

```
In [1]: x = sqrt(25) #sqrt is inbuild function
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[1], line 1
----> 1 x = sqrt(25) #sqrt is inbuild function

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

```
In [2]: import math              # math is module
        x = math.sqrt(25)
        print(x)
```

```
5.0
```

```
In [3]: x1 = math.sqrt(15)
        print(x1)
```

```
3.872983346207417
```

```
In [4]: print(math.floor(3.87))      # floor - minimal or least value
```

```
3
```

```
In [5]: print(math.ceil(3.87))        # ceil - max or highest value
```

```
4
```

```
In [6]: print(math.pow(3,2))
```

```
9.0
```

```
In [7]: print(math.e)   # these are constant
```

```
2.718281828459045
```

```
In [8]: print(math.pi)   # these are constant
```

```
3.141592653589793
```

```
In [9]: import math as m
        m.sqrt(10)
```

```
Out[9]: 3.1622776601683795
```

```
In [10]: from math import sqrt, pow   #  math has many function if you want to call specific
         pow(2,3)
```

```
Out[10]: 8.0
```

```
In [11]: round(pow(2,3))
```

```
Out[11]: 8
```

Help Function

```
In [12]: help(math)
```

```
Help on built-in module math:

NAME
    math

DESCRIPTION
    This module provides access to the mathematical functions
    defined by the C standard.

FUNCTIONS
    acos(x, /)
        Return the arc cosine (measured in radians) of x.

        The result is between 0 and pi.

    acosh(x, /)
        Return the inverse hyperbolic cosine of x.

    asin(x, /)
        Return the arc sine (measured in radians) of x.

        The result is between -pi/2 and pi/2.

    asinh(x, /)
        Return the inverse hyperbolic sine of x.

    atan(x, /)
        Return the arc tangent (measured in radians) of x.

        The result is between -pi/2 and pi/2.

    atan2(y, x, /)
        Return the arc tangent (measured in radians) of y/x.

        Unlike atan(y/x), the signs of both x and y are considered.

    atanh(x, /)
        Return the inverse hyperbolic tangent of x.

    cbrt(x, /)
        Return the cube root of x.

    ceil(x, /)
        Return the ceiling of x as an Integral.

        This is the smallest integer >= x.

    comb(n, k, /)
        Number of ways to choose k items from n items without repetition and without
order.

        Evaluates to n! / (k! * (n - k)!) when k <= n and evaluates
        to zero when k > n.

        Also called the binomial coefficient because it is equivalent
        to the coefficient of k-th term in polynomial expansion of the
```

```
            expression (1 + x)**n.

            Raises TypeError if either of the arguments are not integers.
            Raises ValueError if either of the arguments are negative.

        copysign(x, y, /)
            Return a float with the magnitude (absolute value) of x but the sign of y.

            On platforms that support signed zeros, copysign(1.0, -0.0)
            returns -1.0.

        cos(x, /)
            Return the cosine of x (measured in radians).

        cosh(x, /)
            Return the hyperbolic cosine of x.

        degrees(x, /)
            Convert angle x from radians to degrees.

        dist(p, q, /)
            Return the Euclidean distance between two points p and q.

            The points should be specified as sequences (or iterables) of
            coordinates.  Both inputs must have the same dimension.

            Roughly equivalent to:
                sqrt(sum((px - qx) ** 2.0 for px, qx in zip(p, q)))

        erf(x, /)
            Error function at x.

        erfc(x, /)
            Complementary error function at x.

        exp(x, /)
            Return e raised to the power of x.

        exp2(x, /)
            Return 2 raised to the power of x.

        expm1(x, /)
            Return exp(x)-1.

            This function avoids the loss of precision involved in the direct evaluation
    of exp(x)-1 for small x.

        fabs(x, /)
            Return the absolute value of the float x.

        factorial(n, /)
            Find n!.

            Raise a ValueError if x is negative or non-integral.

        floor(x, /)
            Return the floor of x as an Integral.

            This is the largest integer <= x.

        fmod(x, y, /)
            Return fmod(x, y), according to platform C.

            x % y may differ.

        frexp(x, /)
            Return the mantissa and exponent of x, as pair (m, e).

            m is a float and e is an int, such that x = m * 2.**e.
            If x is 0, m and e are both 0.  Else 0.5 <= abs(m) < 1.0.

        fsum(seq, /)
            Return an accurate floating point sum of values in the iterable seq.

            Assumes IEEE-754 floating point arithmetic.

        gamma(x, /)
            Gamma function at x.

        gcd(*integers)
            Greatest Common Divisor.

        hypot(...)
            hypot(*coordinates) -> value

            Multidimensional Euclidean distance from the origin to a point.

            Roughly equivalent to:
                sqrt(sum(x**2 for x in coordinates))

            For a two dimensional point (x, y), gives the hypotenuse
            using the Pythagorean theorem:  sqrt(x*x + y*y).

            For example, the hypotenuse of a 3/4/5 right triangle is:

                >>> hypot(3.0, 4.0)
                5.0

        isclose(a, b, *, rel_tol=1e-09, abs_tol=0.0)
            Determine whether two floating point numbers are close in value.

              rel_tol
                maximum difference for being considered "close", relative to the
                magnitude of the input values
              abs_tol
                maximum difference for being considered "close", regardless of the
                magnitude of the input values

            Return True if a is close in value to b, and False otherwise.

            For the values to be considered close, the difference between them
            must be smaller than at least one of the tolerances.
            -inf, inf and NaN behave similarly to the IEEE 754 Standard.  That
            is, NaN is not close to anything, even itself.  inf and -inf are
            only close to themselves.

        isfinite(x, /)
            Return True if x is neither an infinity nor a NaN, and False otherwise.

        isinf(x, /)
            Return True if x is a positive or negative infinity, and False otherwise.

        isnan(x, /)
            Return True if x is a NaN (not a number), and False otherwise.

        isqrt(n, /)
            Return the integer part of the square root of the input.

        lcm(*integers)
            Least Common Multiple.

        ldexp(x, i, /)
            Return x * (2**i).

            This is essentially the inverse of frexp().

        lgamma(x, /)
            Natural logarithm of absolute value of Gamma function at x.

        log(...)
            log(x, [base=math.e])
            Return the logarithm of x to the given base.

            If the base is not specified, returns the natural logarithm (base e) of x.

        log10(x, /)
            Return the base 10 logarithm of x.

        log1p(x, /)
            Return the natural logarithm of 1+x (base e).

            The result is computed in a way which is accurate for x near zero.

        log2(x, /)
            Return the base 2 logarithm of x.

        modf(x, /)
            Return the fractional and integer parts of x.

            Both results carry the sign of x and are floats.

        nextafter(x, y, /, *, steps=None)
            Return the floating-point value the given number of steps after x towards y.

            If steps is not specified or is None, it defaults to 1.

            Raises a TypeError, if x or y is not a double, or if steps is not an intege
```

```
r.
        Raises ValueError if steps is negative.

    perm(n, k=None, /)
        Number of ways to choose k items from n items without repetition and with or
der.

        Evaluates to n! / (n - k)! when k <= n and evaluates
        to zero when k > n.

        If k is not specified or is None, then k defaults to n
        and the function returns n!.

        Raises TypeError if either of the arguments are not integers.
        Raises ValueError if either of the arguments are negative.

    pow(x, y, /)
        Return x**y (x to the power of y).

    prod(iterable, /, *, start=1)
        Calculate the product of all the elements in the input iterable.

        The default start value for the product is 1.

        When the iterable is empty, return the start value.  This function is
        intended specifically for use with numeric values and may reject
        non-numeric types.

    radians(x, /)
        Convert angle x from degrees to radians.

    remainder(x, y, /)
        Difference between x and the closest integer multiple of y.

        Return x - n*y where n*y is the closest integer multiple of y.
        In the case where x is exactly halfway between two multiples of
        y, the nearest even value of n is used. The result is always exact.

    sin(x, /)
        Return the sine of x (measured in radians).

    sinh(x, /)
        Return the hyperbolic sine of x.

    sqrt(x, /)
        Return the square root of x.

    sumprod(p, q, /)
        Return the sum of products of values from two iterables p and q.

        Roughly equivalent to:

            sum(itertools.starmap(operator.mul, zip(p, q, strict=True)))

        For float and mixed int/float inputs, the intermediate products
        and sums are computed with extended precision.
    tan(x, /)
        Return the tangent of x (measured in radians).

    tanh(x, /)
        Return the hyperbolic tangent of x.

    trunc(x, /)
        Truncates the Real x to the nearest Integral toward 0.

        Uses the __trunc__ magic method.

    ulp(x, /)
        Return the value of the least significant bit of the float x.

DATA
    e = 2.718281828459045
    inf = inf
    nan = nan
    pi = 3.141592653589793
    tau = 6.283185307179586

FILE
    (built-in)
```

In [13]:
```
help(list)
```

```
Help on class list in module builtins:

class list(object)
 |  list(iterable=(), /)
 |
 |  Built-in mutable sequence.
 |
 |  If no argument is given, the constructor creates a new empty list.
 |  The argument must be an iterable if specified.
 |
 |  Methods defined here:
 |
 |  __add__(self, value, /)
 |      Return self+value.
 |
 |  __contains__(self, key, /)
 |      Return bool(key in self).
 |
 |  __delitem__(self, key, /)
 |      Delete self[key].
 |
 |  __eq__(self, value, /)
 |      Return self==value.
 |
 |  __ge__(self, value, /)
 |      Return self>=value.
 |
 |  __getattribute__(self, name, /)
 |      Return getattr(self, name).
 |
 |  __getitem__(self, index, /)
 |      Return self[index].
 |
 |  __gt__(self, value, /)
 |      Return self>value.
 |
 |  __iadd__(self, value, /)
 |      Implement self+=value.
 |
 |  __imul__(self, value, /)
 |      Implement self*=value.
 |
 |  __init__(self, /, *args, **kwargs)
 |      Initialize self.  See help(type(self)) for accurate signature.
 |
 |  __iter__(self, /)
 |      Implement iter(self).
 |
 |  __le__(self, value, /)
 |      Return self<=value.
 |
 |  __len__(self, /)
 |      Return len(self).
 |
 |  __lt__(self, value, /)
 |      Return self<value.
 |
 |  __mul__(self, value, /)
 |      Return self*value.
 |
 |  __ne__(self, value, /)
 |      Return self!=value.
 |
 |  __repr__(self, /)
 |      Return repr(self).
 |
 |  __reversed__(self, /)
 |      Return a reverse iterator over the list.
 |
 |  __rmul__(self, value, /)
 |      Return value*self.
 |
 |  __setitem__(self, key, value, /)
 |      Set self[key] to value.
 |
 |  __sizeof__(self, /)
 |      Return the size of the list in memory, in bytes.
 |
 |  append(self, object, /)
 |      Append object to the end of the list.
 |
 |  clear(self, /)
 |      Remove all items from list.
 |
 |  copy(self, /)
 |      Return a shallow copy of the list.
 |
 |  count(self, value, /)
 |      Return number of occurrences of value.
 |
 |  extend(self, iterable, /)
 |      Extend list by appending elements from the iterable.
 |
 |  index(self, value, start=0, stop=9223372036854775807, /)
 |      Return first index of value.
 |
 |      Raises ValueError if the value is not present.
 |
 |  insert(self, index, object, /)
 |      Insert object before index.
 |
 |  pop(self, index=-1, /)
 |      Remove and return item at index (default last).
 |
 |      Raises IndexError if list is empty or index is out of range.
 |
 |  remove(self, value, /)
 |      Remove first occurrence of value.
 |
 |      Raises ValueError if the value is not present.
 |
 |  reverse(self, /)
 |      Reverse *IN PLACE*.
 |
 |  sort(self, /, *, key=None, reverse=False)
 |      Sort the list in ascending order and return None.
 |
 |      The sort is in-place (i.e. the list itself is modified) and stable (i.e. the
 |      order of two equal elements is maintained).
 |
 |      If a key function is given, apply it once to each list item and sort them,
 |      ascending or descending, according to their function values.
 |
 |      The reverse flag can be set to sort in descending order.
 |
 |  ----------------------------------------------------------------------
 |  Class methods defined here:
 |
 |  __class_getitem__(...)
 |      See PEP 585
 |
 |  ----------------------------------------------------------------------
 |  Static methods defined here:
 |
 |  __new__(*args, **kwargs)
 |      Create and return a new object.  See help(type) for accurate signature.
 |
 |  ----------------------------------------------------------------------
 |  Data and other attributes defined here:
 |
 |  __hash__ = None
```

In [14]:
```
help(tuple)
```

```
Help on class tuple in module builtins:

class tuple(object)
 |  tuple(iterable=(), /)
 |
 |  Built-in immutable sequence.
 |
 |  If no argument is given, the constructor returns an empty tuple.
 |  If iterable is specified the tuple is initialized from iterable's items.
 |
 |  If the argument is a tuple, the return value is the same object.
 |
 |  Built-in subclasses:
 |      asyncgen_hooks
 |      UnraisableHookArgs
 |
 |  Methods defined here:
 |
 |  __add__(self, value, /)
 |      Return self+value.
 |
 |  __contains__(self, key, /)
 |      Return bool(key in self).
 |
 |  __eq__(self, value, /)
 |      Return self==value.
 |
 |  __ge__(self, value, /)
 |      Return self>=value.
 |
 |  __getattribute__(self, name, /)
 |      Return getattr(self, name).
 |
 |  __getitem__(self, key, /)
 |      Return self[key].
 |
 |  __getnewargs__(self, /)
 |
 |  __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.
 |
 |  __mul__(self, value, /)
 |      Return self*value.
 |
 |  __ne__(self, value, /)
 |      Return self!=value.
 |
 |  __repr__(self, /)
 |      Return repr(self).
 |
 |  __rmul__(self, value, /)
 |      Return value*self.
 |
 |  count(self, value, /)
 |      Return number of occurrences of value.
 |
 |  index(self, value, start=0, stop=9223372036854775807, /)
 |      Return first index of value.
 |
 |      Raises ValueError if the value is not present.
 |
 |  ----------------------------------------------------------------------
 |  Class methods defined here:
 |
 |  __class_getitem__(...)
 |      See PEP 585
 |
 |  ----------------------------------------------------------------------
 |  Static methods defined here:
 |
 |  __new__(*args, **kwargs)
 |      Create and return a new object.  See help(type) for accurate signature.
```

In [15]:
```
help(set)
```

```
Help on class set in module builtins:

class set(object)
 |  set() -> new empty set object
 |  set(iterable) -> new set object
 |
 |  Build an unordered collection of unique elements.
 |
 |  Methods defined here:
 |
 |  __and__(self, value, /)
 |      Return self&value.
 |
 |  __contains__(...)
 |      x.__contains__(y) <==> y in x.
 |
 |  __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.
 |
 |  __iand__(self, value, /)
 |      Return self&=value.
 |
 |  __init__(self, /, *args, **kwargs)
 |      Initialize self.  See help(type(self)) for accurate signature.
 |
 |  __ior__(self, value, /)
 |      Return self|=value.
 |
 |  __isub__(self, value, /)
 |      Return self-=value.
 |
 |  __iter__(self, /)
 |      Implement iter(self).
 |
 |  __ixor__(self, value, /)
 |      Return self^=value.
 |
 |  __le__(self, value, /)
 |      Return self<=value.
 |
 |  __len__(self, /)
 |      Return len(self).
 |
 |  __lt__(self, value, /)
 |      Return self<value.
 |
 |  __ne__(self, value, /)
 |      Return self!=value.
 |
 |  __or__(self, value, /)
 |      Return self|value.
 |
 |  __rand__(self, value, /)
 |      Return value&self.
 |
 |  __reduce__(...)
 |      Return state information for pickling.
 |
 |  __repr__(self, /)
 |      Return repr(self).
 |
 |  __ror__(self, value, /)
 |      Return value|self.
 |
 |  __rsub__(self, value, /)
 |      Return value-self.
 |
 |  __rxor__(self, value, /)
 |      Return value^self.
 |
 |  __sizeof__(...)
 |      S.__sizeof__() -> size of S in memory, in bytes
 |
 |  __sub__(self, value, /)
 |      Return self-value.
 |
 |  __xor__(self, value, /)
 |      Return self^value.
 |
 |  add(...)
 |      Add an element to a set.
 |
 |      This has no effect if the element is already present.
 |
 |  clear(...)
 |      Remove all elements from this set.
 |
 |  copy(...)
 |      Return a shallow copy of a set.
 |
 |  difference(...)
 |      Return the difference of two or more sets as a new set.
 |
 |      (i.e. all elements that are in this set but not the others.)
 |
 |  difference_update(...)
 |      Remove all elements of another set from this set.
 |
 |  discard(...)
 |      Remove an element from a set if it is a member.
 |
 |      Unlike set.remove(), the discard() method does not raise
 |      an exception when an element is missing from the set.
 |
 |  intersection(...)
 |      Return the intersection of two sets as a new set.
 |
 |      (i.e. all elements that are in both sets.)
 |
 |  intersection_update(...)
 |      Update a set with the intersection of itself and another.
 |
 |  isdisjoint(...)
 |      Return True if two sets have a null intersection.
 |
 |  issubset(self, other, /)
 |      Test whether every element in the set is in other.
 |
 |  issuperset(self, other, /)
 |      Test whether every element in other is in the set.
 |
 |  pop(...)
 |      Remove and return an arbitrary set element.
 |      Raises KeyError if the set is empty.
 |
 |  remove(...)
 |      Remove an element from a set; it must be a member.
 |
 |      If the element is not a member, raise a KeyError.
 |
 |  symmetric_difference(...)
 |      Return the symmetric difference of two sets as a new set.
 |
 |      (i.e. all elements that are in exactly one of the sets.)
 |
 |  symmetric_difference_update(...)
 |      Update a set with the symmetric difference of itself and another.
 |
 |  union(...)
 |      Return the union of sets as a new set.
 |
 |      (i.e. all elements that are in either set.)
 |
 |  update(...)
 |      Update a set with the union of itself and others.
 |
 |  ----------------------------------------------------------------------
 |  Class methods defined here:
 |
 |  __class_getitem__(...)
 |      See PEP 585
 |
 |  ----------------------------------------------------------------------
 |  Static methods defined here:
 |
 |  __new__(*args, **kwargs)
 |      Create and return a new object.  See help(type) for accurate signature.
 |
 |  ----------------------------------------------------------------------
 |  Data and other attributes defined here:
 |
 |  __hash__ = None
```

user input function in python || command line input

In [17]:
```
x = input()
y = input()
z = x + y
print(z)      # console is waiting for user to enter input  # default value in 'str
```

```
35
```

In [19]:
```
type(x1)
type(y1)
```

Out[19]:
```
str
```

In [20]:
```
x1 = input('Enter the 1st number') #whenevery you works in input function it always
a1 = int(x1)
y1 = input('Enter the 2nd number') # it wont understand as arithmetic operator
b1 = int(y1)
z1 = a1 + b1
print(z1)
```

```
27
```

for the above code notice we are using many lines because fo that wasting some memory spaces as well

In [21]:
```
x2 = int(input('Enter the 1st number'))
y2 = int(input('Enter the 2nd number'))
z2 = x2 + y2
z2
```

Out[21]:
```
6
```

lets take input from the user in char format, but we dont have char format in python

In [26]:
```
ch = input('enter a char')
print(ch)
```

```
Vihari Nandan
```

In [23]:
```
print(ch[0])
```

```
V
```

In [24]:
```
print(ch[1])
```

```
i
```

In [25]:
```
print(ch[-1])
```
```
n
```

In [27]:
```
ch = input('enter a char')[0]
print(ch)
```
```
V
```

In [29]:
```
ch = input('enter a char')[1:3]
print(ch)
```
```
ih
```

In [30]:
```
ch = input('enter a char')
print(ch) # if you enter as 2 + 6 -1 we get output as 2 + 6-1 only
```
```
Vihari Nandan
```

EVAL function using input

In [31]:
```
result = eval(input('enter an expr'))
print(result)
```
```
11
```

if you want to pass the value in cmd can we pass the value like this

when we run the above code in cmd then we get the value to 11 only but how to add them then we need to use very important concept called (argv) -- (argument values) what it does if you pass 1 value then 1 value it will display but if you pass 2 value then it will display 2 values argv -- it will understand based on index number & bydefault index number 0 means that is file name