Meeting 1 Live

November 26, 2025

[1]:
```
# Hello World
```

[2]:
```
1 + 2
```

[2]:
```
3
```

[3]:
```
x = 1
```

[4]:
```
y = 2
```

[5]:
```
x + y
```

[5]:
```
3
```

[6]:
```
type(x)
```

[6]:
```
int
```

[7]:
```
x = 3.0
```

[8]:
```
type(x)
```

[8]:
```
float
```

[9]:
```
z = x * y
```

[10]:
```
z
```

[10]:
```
6.0
```

[11]:
```
1 / 2
```

[11]:
```
0.5
```

[12]:
```
1 // 2
```

[12]:
```
0
```

[13]:
```
int(1 / 2)
```

[13]: 0

[14]:
```
1 + 2 * 3
```

[14]: 7

[15]:
```
(1 + 2) * 3
```

[15]: 9

[16]:
```
3 ** 2
```

[16]: 9

[17]:
```
2 ^ 3
```

[17]: 1

[18]:
```
1 == 1
```

[18]: True

[19]:
```
1 == 1.0
```

[19]: True

[20]:
```
1 == 1.01
```

[20]: False

[21]:
```
x
```

[21]: 3.0

[22]:
```
x = x + 1
```

[23]:
```
x
```

[23]: 4.0

[24]:
```
x++
```

```
  Cell In[24], line 1
    x++
      ^
```

```
SyntaxError: invalid syntax
```

[25]:
```
x += 1
```

[26]:
```
round(3.9)
```

[26]:
```
4
```

[27]:
```
round(3.141592654, 2)
```

[27]:
```
3.14
```

[28]:
```
round(3.141592654, 3)
```

[28]:
```
3.142
```

[29]:
```
round?
```

```
Signature: round(number, ndigits=None)
Docstring:
Round a number to a given precision in decimal digits.

The return value is an integer if ndigits is omitted or None.  Otherwise
the return value has the same type as the number.  ndigits may be negative.
Type:      builtin_function_or_method
```

[30]:
```
int?
```

```
Init signature: int(self, /, *args, **kwargs)
Docstring:
int([x]) -> integer
int(x, base=10) -> integer

Convert a number or string to an integer, or return 0 if no arguments
are given.  If x is a number, return x.__int__().  For floating point
numbers, this truncates towards zero.

If x is not a number or if base is given, then x must be a string,
bytes, or bytearray instance representing an integer literal in the
given base.  The literal can be preceded by '+' or '-' and be surrounded
by whitespace.  The base defaults to 10.  Valid bases are 0 and 2-36.
Base 0 means to interpret the base from the string as an integer literal.
>>> int('0b100', base=0)
4
Type:           type
Subclasses:     bool, IntEnum, IntFlag, _NamedIntConstant
```

[31]:
```
10 % 3
```

[31]:
```
1
```

[32]:
```
x = 10
```

[35]:
```
x %= 3 # x = x % 3
```

[34]:
```
print(x)
```

```
1
```

[36]:
```
my_str = 'Hello World'
```

[37]:
```
my_str2 = "Hello World"
```

[38]:
```
my_str == my_str2
```

[38]:
```
True
```

[39]:
```
my_str3 = 'Welcome to this year's course'
```

```
  Cell In[39], line 1
    my_str3 = 'Welcome to this year's course'
                                           ^
SyntaxError: unterminated string literal (detected at line 1)
```

[40]:
```
my_str3 = "Welcome to this year's course"
```

[41]:
```
my_str3 = 'Welcome to this year\'s course'
```

[42]:
```
my_str_triple = """Hello World
Welcome to this year's course
This is a double-quote: "
"""
```

[43]:
```
print(my_str_triple)
```

```
Hello World
Welcome to this year's course
This is a double-quote: "
```

[44]:
```
my_str_triple
```

[44]:
```
'Hello World\nWelcome to this year\'s course\nThis is a double-quote: "\n'
```

[45]:
```
my_str = "A" + "B"
```

[46]:
```
my_str
```

[46]:
```
'AB'
```

[47]:
```
my_str = "A" * 10
```

[48]:
```
my_str
```

[48]:
```
'AAAAAAAAAA'
```

[49]:
```
my_str.lower()
```

[49]:
```
'aaaaaaaaaa'
```

[50]:
```
my_str
```

[50]:
```
'AAAAAAAAAA'
```

[51]:
```
my_str.replace("A", "B")
```

[51]:
```
'BBBBBBBBBB'
```

[52]:
```
my_str.replace?
```

```
Signature: my_str.replace(old, new, count=-1, /)
Docstring:
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.
Type:      builtin_function_or_method
```

[53]:
```
dir(my_str)
```

[53]:
```
['__add__',
 '__class__',
 '__contains__',
 '__delattr__',
 '__dir__',
```

```
'__doc__',
'__eq__',
'__format__',
'__ge__',
'__getattribute__',
'__getitem__',
'__getnewargs__',
'__gt__',
'__hash__',
'__init__',
'__init_subclass__',
'__iter__',
'__le__',
'__len__',
'__lt__',
'__mod__',
'__mul__',
'__ne__',
'__new__',
'__reduce__',
'__reduce_ex__',
'__repr__',
'__rmod__',
'__rmul__',
'__setattr__',
'__sizeof__',
'__str__',
'__subclasshook__',
'capitalize',
'casefold',
'center',
'count',
'encode',
'endswith',
'expandtabs',
'find',
'format',
'format_map',
'index',
'isalnum',
'isalpha',
'isascii',
'isdecimal',
'isdigit',
'isidentifier',
'islower',
'isnumeric',
```

```
 'isprintable',
 'isspace',
 'istitle',
 'isupper',
 'join',
 'ljust',
 'lower',
 'lstrip',
 'maketrans',
 'partition',
 'removeprefix',
 'removesuffix',
 'replace',
 'rfind',
 'rindex',
 'rjust',
 'rpartition',
 'rsplit',
 'rstrip',
 'split',
 'splitlines',
 'startswith',
 'strip',
 'swapcase',
 'title',
 'translate',
 'upper',
 'zfill']
```

[54]:
```
my_str.isascii?
```

```
Signature: my_str.isascii()
Docstring:
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.
Type:      builtin_function_or_method
```

[56]:
```
my_str.isascii()
```

[56]:
```
True
```

[57]:
```
my_str.isdigit()
```

[57]:
```
False
```

[58]:
```
name = "Ben"
age = 1337
```

[60]:
```
print("Hello, I am " + name + " and I am " + age + " years old")
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[60], line 1
----> 1 print("Hello, I am " + name + " and I am " + age + " years old")

TypeError: can only concatenate str (not "int") to str
```

[61]:
```
print("Hello, I am " + name + " and I am " + str(age) + " years old")
```

```
Hello, I am Ben and I am 1337 years old
```

[62]:
```
print(f"Hello, I am {name} and I am {age} years old")
```

```
Hello, I am Ben and I am 1337 years old
```

[63]:
```
print(f"Hello, I am {name} and I am {age - 1297} years old")
```

```
Hello, I am Ben and I am 40 years old
```

[64]:
```
print(f"Hello, I am {name.upper()} and I am {age - 1297} years old")
```

```
Hello, I am BEN and I am 40 years old
```

[65]:
```
my_str
```

[65]:
```
'AAAAAAAAAA'
```

[66]:
```
my_str = "Hello World"
```

[68]:
```
my_str[0]
```

[68]:
```
'H'
```

[69]:
```
my_str[10]
```

[69]:
```
'd'
```

[70]:
```
my_str[-1]
```

[70]:
```
'd'
```

[71]:
```
my_str[-2]
```

[71]:
```
'l'
```

[72]:
```
my_str[0:6]
```

[72]:
```
'Hello '
```

[73]:
```
my_str[:6]
```

[73]:
```
'Hello '
```

[74]:
```
my_str[6:]
```

[74]:
```
'World'
```

[75]:
```
my_str[::2]
```

[75]:
```
'HloWrd'
```

[76]:
```
my_str[::3]
```

[76]:
```
'HlWl'
```

[77]:
```
my_str[::-1]
```

[77]:
```
'dlroW olleH'
```

[ ]: