In Python, there are several types of errors, which can be categorized into three main classes:

1. Syntax Errors:

These are also known as parsing errors and occur when the Python interpreter encounters a statement that violates the language's syntax rules. These errors prevent the code from running. Some common syntax errors include:

- Missing or mismatched parentheses, brackets, or curly braces.

- Misspelled keywords.

- Missing colons at the end of statements where they are required.

- Improper indentation.

In [ ]:

In [1]:
```
if x = 5:  # SyntaxError: invalid syntax
    print("Hello, World")
```

```
  Input In [1]
    if x = 5:  # SyntaxError: invalid syntax
       ^
SyntaxError: invalid syntax. Maybe you meant '==' or ':=' instead of '='?
```

In [3]:
```
for = 3
print(for)
```

```
  Input In [3]
    for = 3
        ^
SyntaxError: invalid syntax
```

In [ ]:

In [ ]:

2. Runtime Errors:

These errors occur during the execution of a program, typically when the program tries to perform an operation that is not allowed. Some common runtime errors include:

NameError: Occurs when a variable or function is used before it's defined.

TypeError: Occurs when an operation is performed on incompatible data types.

ValueError: Occurs when a function receives an argument of the correct data type but an invalid value.

ZeroDivisionError: Occurs when you try to divide a number by zero.

In [ ]:

In [9]:
```
# Attempt to use an undefined variable

a = 6
c = b + a
print(c)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Input In [9], in <module>
      1 # Attempt to use an undefined variable
      3 a = 6
----> 4 c = b + a
      5 print(c)

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

In [ ]:

In [10]:
```
x = 5
y = "2"
z = x + y  # TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Input In [10], in <module>
      1 x = 5
      2 y = "2"
----> 3 z = x + y

TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

In [ ]:

In [12]:
```
num_str = "abc"
num = int(num_str)
print(num)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Input In [12], in <module>
      1 num_str = "abc"
----> 2 num = int(num_str)
      3 print(num)

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

In [ ]:

In [13]:
```
a = 3
b = 0
print(a/b)
```

```
---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call last)
Input In [13], in <module>
      1 a = 3
      2 b = 0
----> 3 print(a/b)

ZeroDivisionError: division by zero
```

In [ ]:

In [ ]:

3. Logical Error:

These errors are the most subtle and challenging to detect because the code runs without any errors or exceptions, but it doesn't produce the expected result due to a flaw in the algorithm or logic. These errors are also known as "bugs." Debugging is required to identify and fix them.

In [14]:
```
sub1 = 70
sub2= 80
sub3= 90
avg = sub1+sub2+sub3/3
print(avg)
```

```
180.0
```

In [ ]: