



7



# Python Quick Tips

## Try/Except Statements



# Python Quick Tips #7

## Try/Except Statements

# Errors



Errors must be  
**anticipated and handled**

# Example

OK

Error

```
save.py x
1  var1 = 5.6
2  var2 = 8
3
4  sum = var1 + var2
5
6  print(sum)

Shell x
>>> %Run save.py
13.6
>>> |
```

```
save.py x
1  var1 = 5.6
2  var2 = 'test'
3
4  sum = var1 + var2
5
6  print(sum)

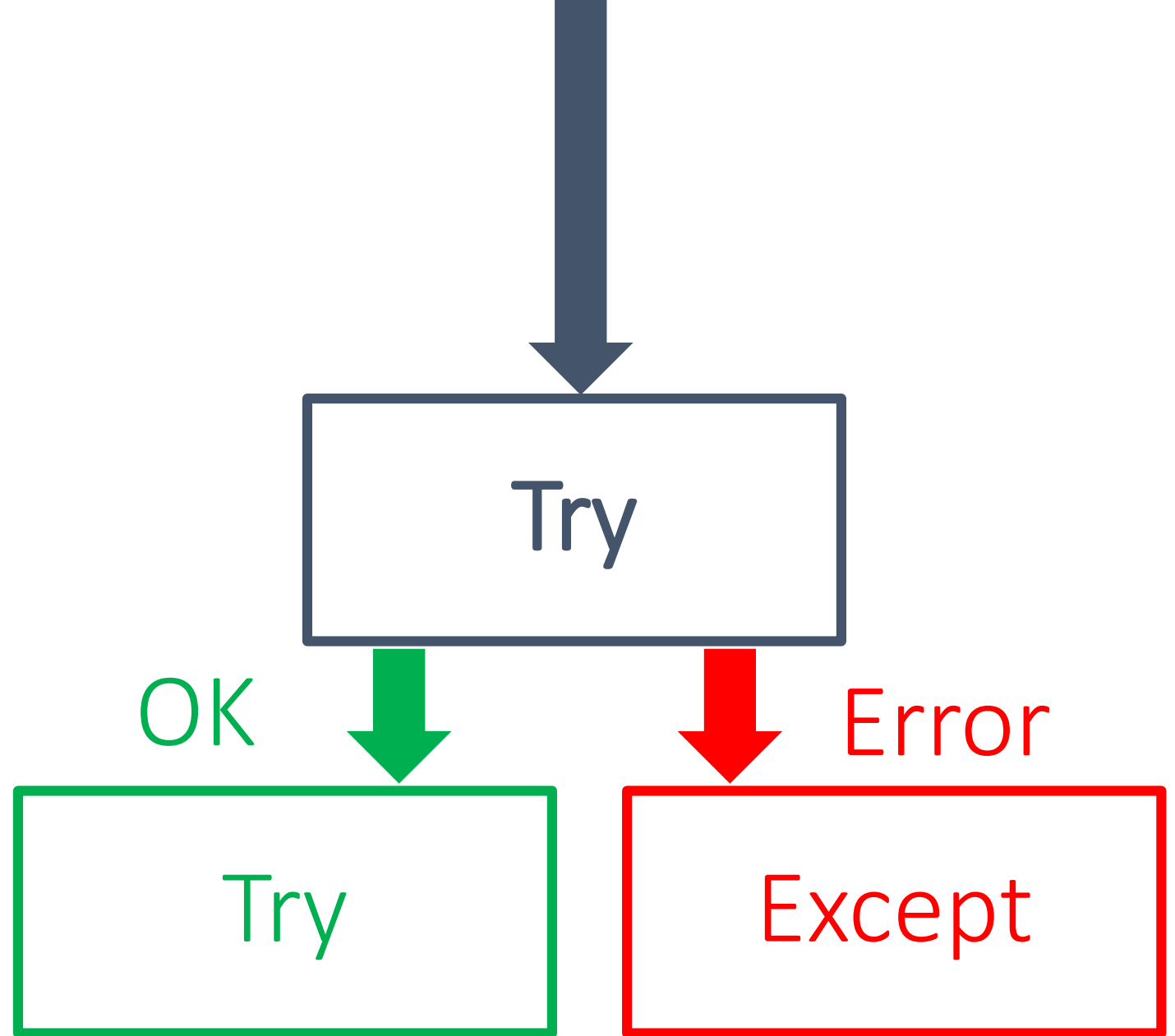
Shell x
>>> %Run save.py
Traceback (most recent call last):
  File "C:\Users\Gavin\Desktop\save.py", line 4, in <module>
    sum = var1 + var2
TypeError: unsupported operand type(s) for +
: 'float' and 'str'
>>> |
```

Using If  
= Yuck!

```
save.py x
1 var1 = 5.6
2 var2 = '4'
3
4 check1 = type(var1)==int or type(var1)==float
5 check2 = type(var2)==int or type(var2)==float
6
7 check3 = type(var1)==str and type(var2)==str
8 check4 = type(var1)!=str and type(var2)!=str
9
10 check5 = check3 or check4
11
12 check = check1 and check2 and check5
13
14 if check:
15     sum = var1 + var2
16     print(sum)
17 else:
18     print('Inputs must be real/int or string')
```

```
Shell x
>>> %Run save.py
    Inputs must be real/int or string
>>> |
```

# Try/Except



# Try/Except Statements

Syntax

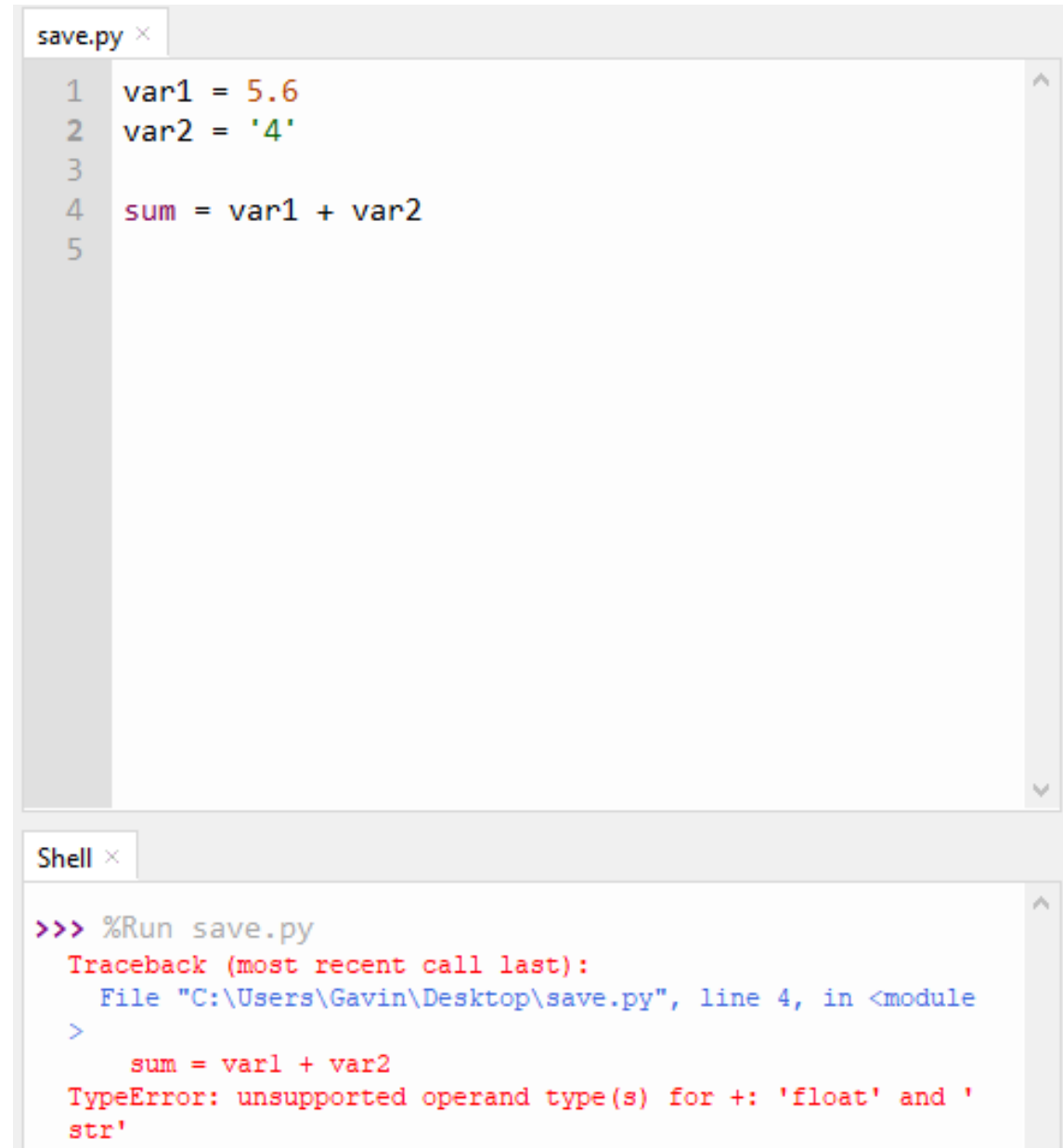
**try:**

(tab)test (no error)

**except:**

(tab)unless (error)

# Unhandled



The image shows a screenshot of a Python IDE with two panels. The top panel, titled 'save.py', contains the following code:

```
1 var1 = 5.6
2 var2 = '4'
3
4 sum = var1 + var2
5
```

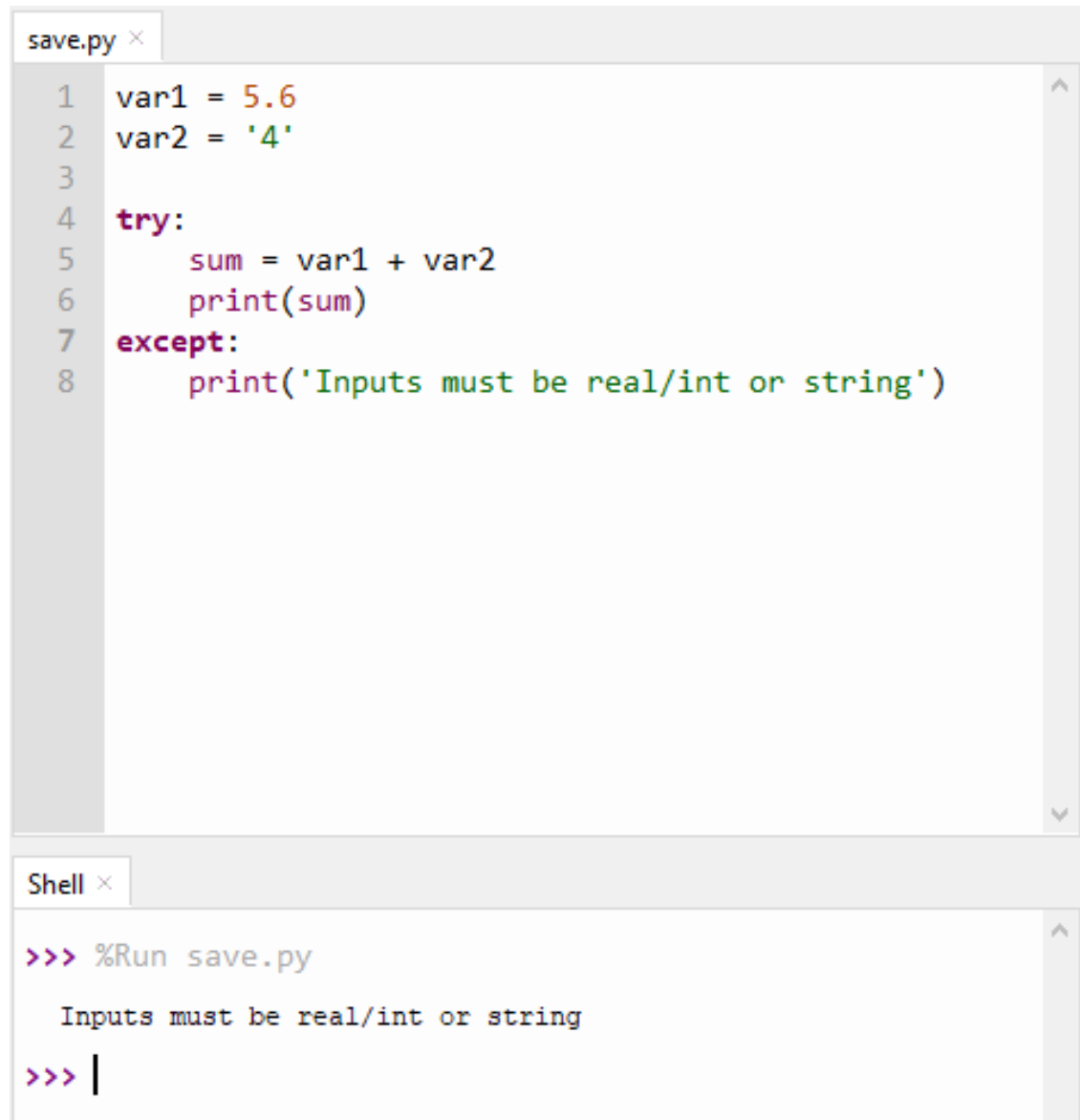
The bottom panel, titled 'Shell', shows the execution of the script and the resulting error:

```
>>> %Run save.py
Traceback (most recent call last):
  File "C:\Users\Gavin\Desktop\save.py", line 4, in <module>
    >
      sum = var1 + var2
TypeError: unsupported operand type(s) for +: 'float' and 'str'
```

The error message indicates a `TypeError` because the code attempts to add a float (`5.6`) and a string (`'4'`), which are incompatible types for the `+` operator.



# Handled!



The image shows a screenshot of a Python IDE with two panels. The top panel, titled 'save.py', contains a Python script with 8 lines of code. The script defines two variables, 'var1' (5.6) and 'var2' ('4'), and attempts to add them. A try-except block catches the resulting TypeError and prints a custom error message. The bottom panel, titled 'Shell', shows the command prompt where the script was executed, resulting in the custom error message being displayed.

```
save.py x
1 var1 = 5.6
2 var2 = '4'
3
4 try:
5     sum = var1 + var2
6     print(sum)
7 except:
8     print('Inputs must be real/int or string')
```

```
Shell x
>>> %Run save.py
Inputs must be real/int or string
>>> |
```

## Else/Finally

We can also add

**else:**

(tab)this (no error)

**finally:**

(tab)that (either way)

# Example

OK

```
save.py x
1 var1 = 5.6
2 var2 = 4
3
4 try:
5     sum = var1 + var2
6     print(sum)
7 except:
8     print('Inputs must be real/int or string')
9 else:
10    print('Script was successful')
11 finally:
12    print('Script was run')
```

```
Shell x
>>> %Run save.py
9.6
Script was successful
Script was run
```

Error

```
save.py x
1 var1 = 5.6
2 var2 = '4'
3
4 try:
5     sum = var1 + var2
6     print(sum)
7 except:
8     print('Inputs must be real/int or string')
9 else:
10    print('Script was successful')
11 finally:
12    print('Script was run')
```

```
Shell x
>>> %Run save.py
Inputs must be real/int or string
Script was run
```



Next on #8  
Loops (for/while)



7



# Python Quick Tips

## Try/Except Statements