Table of contents


```
<a id=""></a>
```

1. Exception / Error handling

- A try statement has the general form:

```
try:
    <body>
except <ErrorType>:
    <handler>
```

(go to top)

In [24]:
```
def sqrt(x):
    """Returns the sq root of a number"""
    try:
        return x ** 0.5
    except:
        print('x must be a float or int')

print(sqrt('hello'))
```

```
x must be a float or int
None
```

In [26]:
```
print(sqrt(4))
```

```
2.0
```

2. Catch Type Errors only

- When an operation or function is applied to an object of innapropriate type etc.....

(go to top)

In [28]:
```
def sqrt(x):
    """Returns the sq root of a number"""
    try:
        return x ** 0.5
    except TypeError:
        print('x must be a float or int')

print(sqrt('hello'))
```

```
x must be a float or int
None
```

In [22]:
```
print(sqrt(4))
```

```
2.0
```

3. Raise an Error

- Sometimes an input might run perfectly in python
- But if we dont want to allow that kind of input we use raise

(go to top)

In [25]:
```
def sqrt(x):
    """Returns the sq root of a number"""
    if type(x) != type(2):
        raise TypeError('x must be a float of int')
    elif x < 0:
        raise ValueError('x must be non-negative')
    return x ** 0.5
```

In [28]:
```
print(sqrt('x'))
```

```
---------------------------------------------------------------------
------
TypeError                                 Traceback (most recent call
last)
<ipython-input-28-9344b316444c> in <module>
----> 1 print(sqrt('x'))

<ipython-input-25-d836447f87e7> in sqrt(x)
      2     """Returns the sq root of a number"""
      3     if type(x) != type(2):
----> 4         raise TypeError('x must be a float of int')
      5     elif x < 0:
      6         raise ValueError('x must be non-negative')

TypeError: x must be a float of int
```

In [30]:
```
print(sqrt(-2))
```

```
---------------------------------------------------------------------
------
ValueError                                Traceback (most recent call
last)
<ipython-input-30-4f4356609c54> in <module>
----> 1 print(sqrt(-2))

<ipython-input-25-d836447f87e7> in sqrt(x)
      4         raise TypeError('x must be a float of int')
      5     elif x < 0:
----> 6         raise ValueError('x must be non-negative')
      7     return x ** 0.5

ValueError: x must be non-negative
```

3.1. Example

- The multiple excepts are similar to elifs. If an error occurs, Python will try each except in tum looking for one that matches the type of error. The bare except at the bottom in this example acts like an else and will be used as the default if no previous except error type matches. If there is no default at the bottom and none of the except types match the error, then the program crashes and Python reports the error.
- If you follow the error type with an `as <variable>` in an except clause, Python will assign that variable the actual exception object. In this case, I turned the exception into a string and looked at the message to see what caused the ValueError. Notice that this text is exactly what Python prints out if the error is not caught (e.g., `ValueError: math domain error`).

(go to top)

In [47]:
```
def main():
    print("This program finds the real solutions to a quadratic\n")

    a = float(input("Enter coefficient a: "))
    b = float(input("Enter coefficient b: "))
    c = float(input("Enter coefficient c: "))
    discRoot = math.sqrt(b * b - 4 * a * c)
    root1 = (-b + discRoot) / (2 * a)
    root2 = (-b - discRoot) / (2 * a)
    print("\nThe solutions are:", root1, root2 )

main()
#enter 2,2,6 for a,b,c
# note ValueError: math domain error
```

```
Enter coefficient b: 2
Enter coefficient c: 6

---------------------------------------------------------------------------
--------
ValueError                                Traceback (most recent ca
ll last)
<ipython-input-47-392f6f16faa2> in <module>
     10     print("\nThe solutions are:", root1, root2 )
     11
---> 12 main()
     13 #enter 2,2,6 for a,b,c

<ipython-input-47-392f6f16faa2> in main()
      5     b = float(input("Enter coefficient b: "))
      6     c = float(input("Enter coefficient c: "))
----> 7     discRoot = math.sqrt(b * b - 4 * a * c)
      8     root1 = (-b + discRoot) / (2 * a)
      9     root2 = (-b - discRoot) / (2 * a)

ValueError: math domain error
```

In [48]:
```
def main():
    try:
        print("This program finds the real solutions to a quadratic\n"

        a = float(input("Enter coefficient a: "))
        b = float(input("Enter coefficient b: "))
        c = float(input("Enter coefficient c: "))
        discRoot = math.sqrt(b * b - 4 * a * c)
        root1 = (-b + discRoot) / (2 * a)
        root2 = (-b - discRoot) / (2 * a)
        print("\nThe solutions are:", root1, root2 )
    except ValueError as v_err:
        if str(v_err) =="math domain error":
            print("No Real Roots")
        else:
            print("Invalid coefficient given")
    except :
        print("\nSomething went wrong, sorry!")

main()
#enter 2,2,6 for a,b,c
```

```
This program finds the real solutions to a quadratic

Enter coefficient a: 2
Enter coefficient b: k
Invalid coefficient given
```

In [2]:
```
import random as r
import string

# import characters
upper = string.ascii_uppercase
lower = string.ascii_lowercase
nums = string.digits
sym = string.punctuation

while True:
    passLen = input('Enter a number between 8 and 15: ')
    try:
        if (8 <= int(passLen) <= 15):
            passLen = int(passLen)
            break
        else:
            raise ValueError
    except ValueError:
        print('Input must be a number btw 8 and 15: ')
        passLen = input('Enter a number between 8 and 15: ')

# create temporary password to hold all possible characters
```

```
# create temporary password to hold all possible characters
tempPass = r.sample(upper, passLen//4) + r.sample(lower, passLen//4) +

#shuffle password
r.shuffle(tempPass)

password = ''.join(tempPass)
password
```

```
Enter a number between 8 and 15: u
Input must be a number btw 8 and 15:

Enter a number between 8 and 15:
 ii
```

```
---------------------------------------------------------------------
------
KeyboardInterrupt                         Traceback (most recent call
last)
<ipython-input-2-2adbab3c86ea> in <module>
     18     except ValueError:
     19         print('Input must be a number btw 8 and 15: ')
---> 20         passLen = input('Enter a number between 8 and 15: ')
     21
     22 # create temporary password to hold all possible characters

~/opt/anaconda3/lib/python3.8/site-packages/ipykernel/kernelbase.py
in raw_input(self, prompt)
    858                 "raw_input was called, but this frontend does
not support input requests."
    859             )
--> 860         return self._input_request(str(prompt),
    861             self._parent_ident,
    862             self._parent_header,

~/opt/anaconda3/lib/python3.8/site-packages/ipykernel/kernelbase.py
in _input_request(self, prompt, ident, parent, password)
    902             except KeyboardInterrupt:
    903                 # re-raise KeyboardInterrupt, to truncate tra
ceback
--> 904                 raise KeyboardInterrupt("Interrupted by user"
) from None
    905             except Exception as e:
    906                 self.log.warning("Invalid Message:", exc_info
=True)

KeyboardInterrupt: Interrupted by user
```

4. Title

(go to top)

5. Title

(go to top)

6. Title

(go to top)

7. Title

(go to top)

8. Title

(go to top)

9. Title

(go to top)

10. Title

(go to top)

11. Title

(go to top)

12. Title

(go to top)

13. Title

(go to top)

14. Title

(go to top)

15. Title

(go to top)

16. Title

(go to top)

17. Title

(go to top)

18. Title

(go to top)

19. Title

(go to top)

20. Title

(go to top)

In [ ]:

In [ ]: