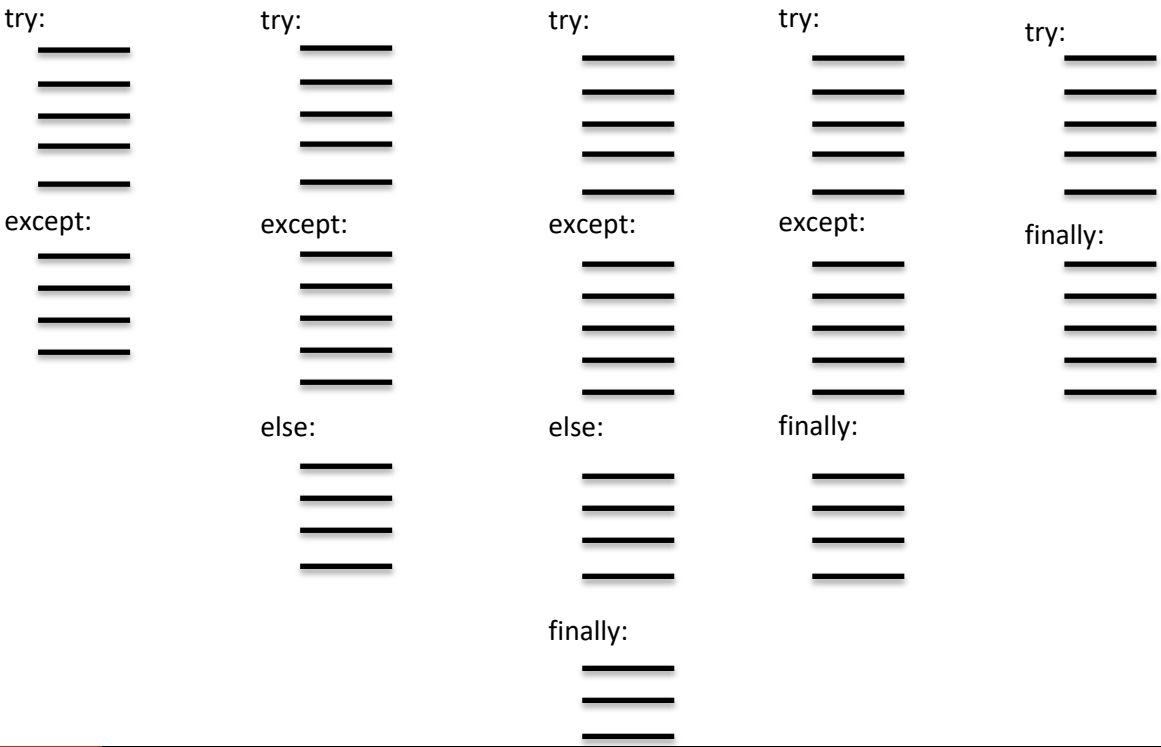
Python Fundamentals day 65

Today's agenda

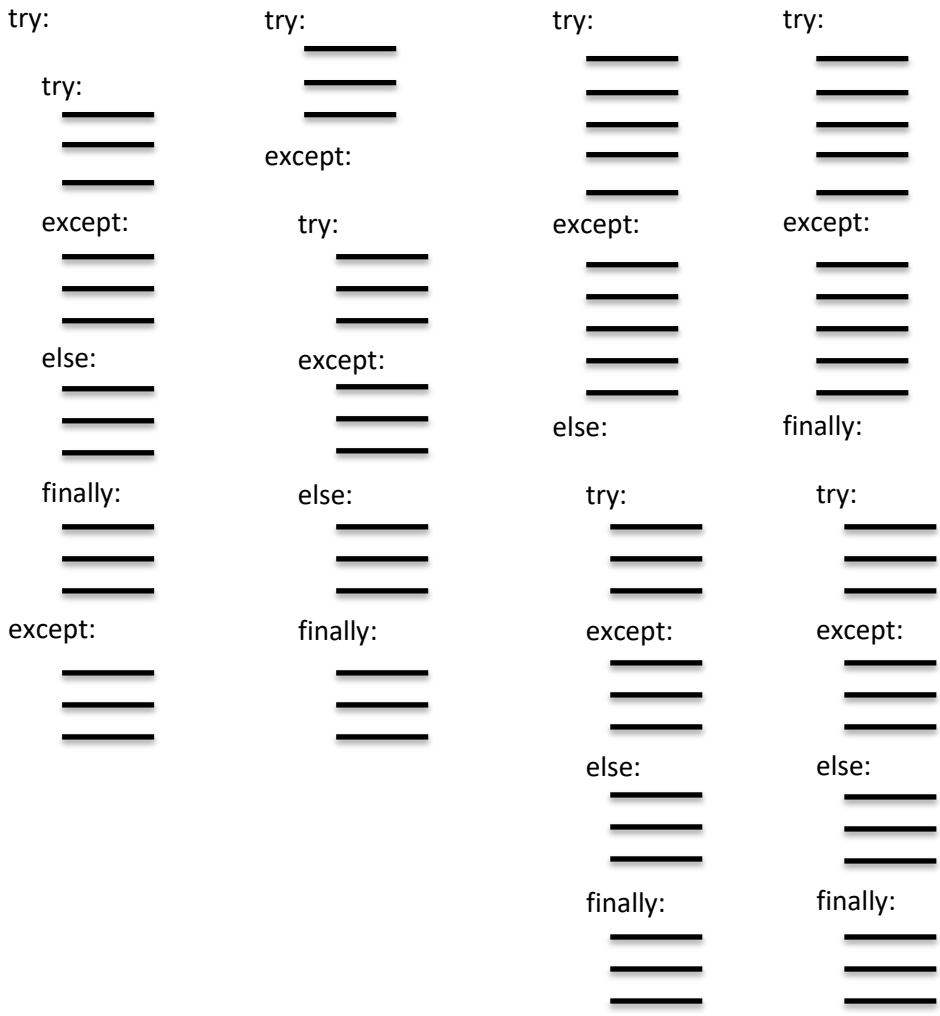
- Valid combinations
- Exception hierarchy
- Custom Exceptions

Valid Combinations

We have seen several blocks used in exception handling. Let us see what are these valid combinations of blocks in below segment

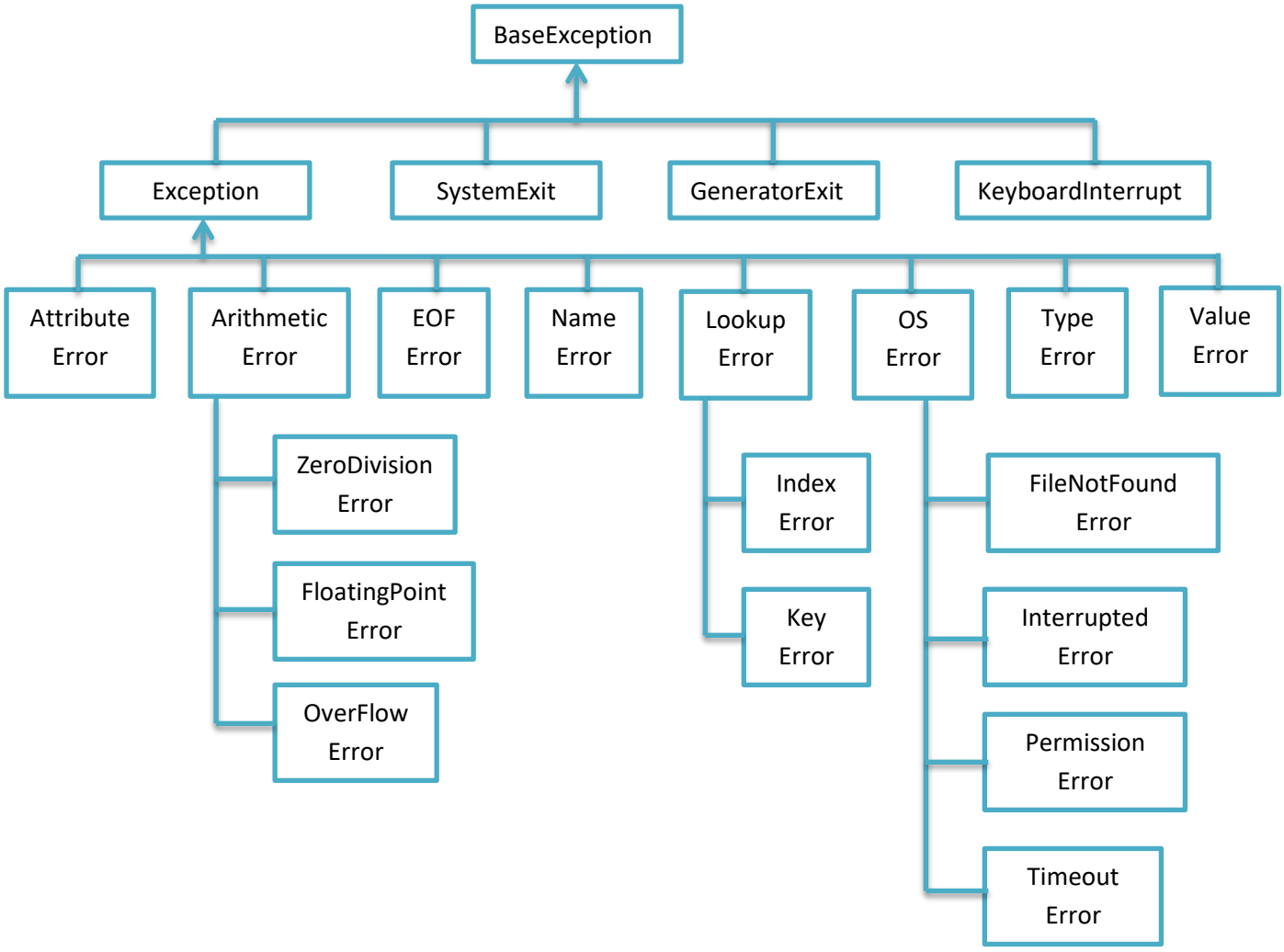


We can also have these nested in one another



Exception Hierarchy

The Python exception class hierarchy consists of a few dozen different exceptions spread across a handful of important base class types. In below chart let us see hierarchy in detail



Custom Exceptions

Let us previous example

```
def validate(mob):
    if len(mob)==10:
        print('Valid mobile number')
    else:
        raise ValueError

def main():
    mob=input()
    validate(mob)

main()
```

Output:

```
  File "C:/Users/rooman/Downloads/test.py", line 11, in <module>
    main()

  File "C:/Users/rooman/Downloads/test.py", line 9, in main
    validate(mob)

  File "C:/Users/rooman/Downloads/test.py", line 5, in validate
    raise ValueError

ValueError
```

We can see we have got the value error. But it is not conveying proper message. The user might not understand why value error is generated. So to resolve this issue, in python users can create custom exceptions. Let us see how

```
class InvalidMobileNumberError(Exception):
    pass

def validate(mob):
    if len(mob)==10:
        print('Valid Mobile Number')
    else:
        raise InvalidMobileNumberError('Enter 10 digit mobile number')

def main():
    mob=input()
    validate(mob)

main()
```

Output:

```
  File "C:/Users/rooman/Downloads/test.py", line 14, in <module>
    main()

  File "C:/Users/rooman/Downloads/test.py", line 12, in main
    validate(mob)

  File "C:/Users/rooman/Downloads/test.py", line 8, in validate
    raise InvalidMobileNumberError('Enter 10 digit mobile number')

InvalidMobileNumberError: Enter 10 digit mobile number
```

Now we can see exception generated clearly conveys the message. Let us take another example and get more clarity

```
def menu(item):
    if item=='pizza':
        print('Enjoy your pizza')
    elif item=='idli':
        print('Enjoy your idli')
    elif item=='burger':
        print('Enjoy your burger')
    else:
        raise NameError

def main():
    item=input()
    menu(item)

main()
```

Output:

```
  File "C:/Users/rooman/Downloads/test.py", line 15, in <module>
    main()

  File "C:/Users/rooman/Downloads/test.py", line 13, in main
    menu(item)

  File "C:/Users/rooman/Downloads/test.py", line 9, in menu
    raise NameError

NameError
```

Given pasta as the input we got NameError. But user may not know why is error generated, therefore lets customise the exception

```
class ItemNotInMenuError(Exception):
    pass

def menu(item):
    if item=='pizza':
        print('Enjoy your pizza')
    elif item=='idli':
        print('Enjoy your idli')
    elif item=='burger':
        print('Enjoy your burger')
    else:
        raise ItemNotInMenuError('Item not present in menu')

def main():
    item=input()
    menu(item)

main()
```

Output:

```
  File "C:/Users/rooman/Downloads/test.py", line 18, in <module>
    main()

  File "C:/Users/rooman/Downloads/test.py", line 16, in main
    menu(item)

  File "C:/Users/rooman/Downloads/test.py", line 12, in menu
    raise ItemNotInMenuError('Item not present in menu')

ItemNotInMenuError: Item not present in menu
```

Now we can see the proper message and resolve it by giving try and except blocks

```
class ItemNotInMenuError(Exception):
    pass

def menu(item):
    if item=='pizza':
        print('Enjoy your pizza')
    elif item=='idli':
        print('Enjoy your idli')
    elif item=='burger':
        print('Enjoy your burger')
    else:
        raise ItemNotInMenuError('Item not present in menu')

def main():
    item=input()
    try:
        menu(item)
    except ItemNotInMenuError as e:
        print(e)

main()
```

Output:

```
In [17]: runfile('C:/Users/rooman/Downloads/test.py', wdir='C:/Users/
rooman/Downloads')

pasta
Item not present in menu
```

Now let us take an example of creating an account. Which will check if the username is unique and password satisfying the conditions

```
class DuplicateUserError(Exception):
    pass

class WeakPasswordError(Exception):
    pass

class User:

    user_name=set()

    def __init__(self,un,mob,pwd):
        self.un=un
        self.mob=mob
        self.pwd=pwd
        self.add_user()
        self.validate()

    def add_user(self):
        if self.un in User.user_name:
            raise DuplicateUserError('Username already exists')
        else:
            User.user_name.add(self.un)

    def validate(self):
        uc=lc=num=sp=0
        for i in self.pwd:
            if i.isupper():
                uc+=1
            elif i.islower():
                lc+=1
            elif i.isdigit():
                num+=1
            else:
                sp=+1


        if len(self.pwd)<6 or uc==0 or\
        lc==0 or num==0 or sp==0:
            raise WeakPasswordError('Password not strong enough')

def main():
    un=input('Enter Username: ')
    mob=int(input('Enter Mobile: '))
    pwd=input('Enter the password: ')
    try:
        u1=User(un,mob,pwd)
        u2=User(un,mob,pwd)
    except DuplicateUserError as e:
        print(e)
    except WeakPasswordError as e:
        print(e)
    except:
        print('Hey some issue occurred')
    else:
        print('Account created successfully')

main()
```

Output:

```
In [18]: runfile('C:/Users/rooman/Downloads/test.py', wdir='C:/Users/
rooman/Downloads')

Enter Username: pythonfan

Enter Mobile: 8899562325

Enter the password: #rooman123
Password not strong enough
```

Let us give existing username and check the output

```
In [19]: runfile('C:/Users/rooman/Downloads/test.py', wdir='C:/Users/
rooman/Downloads')

Enter Username: pythonfan

Enter Mobile: 8899562325

Enter the password: #Rooman123
Username already exists
```