Di hari 32 saya belajar mengenai exception (try dan except) dan match statement. Sebelum mendalami lebih lanjut pertama ada dua error, yaitu syntax error dan error runtime. Jadi:

- **Syntax Error** = **Salah tulis code** → Program tidak bisa mulai
- **Runtime Error** = **Salah logic/data** → Program jalan tapi crash di tengah

Contoh penyebab syntax error:

```python
# ❌ SALAH SYNTAX
print("Hello world"  # Kurang tutup kurung
if x = 5:            # Seharusnya ==
for i in range(10    # Kurang tutup kurung
```

Contoh penyebab runtime error:

```
# ✅ SYNTAX BENAR, TAPI RUNTIME ERROR
print(10 / 0)               # ZeroDivisionError
print(int("bukan_angka"))   # ValueError
print(list[10])             # IndexError (jika list pendek)
```

- Error karena 10 gk bisa dibagi nol
- Error karena Data string huruf dipaksa rubah jadi integer
- Error karena bikin list tapi cuman satu item aja

Nah mengatasi runtime error ini menggunakan exception, contohnya:

```
input_user = int(input("masukan angka: "))
hasil = “nan”

try:
   hasil = 10/input_user
except:
   print("input tidak boleh 0")

print(f"hasil = {hasil}")
```

Nah kalo semisal si user menginput nilai Nol hasilnya maka hasilnya

Input tidak boleh 0

Hasil = nan

Kenapa begitu karena logika exception ini pertama dia menjalankan yang method try kemudian karena hasilnya akan error maka dia otomatis akan menjalankan yang didalam method except.

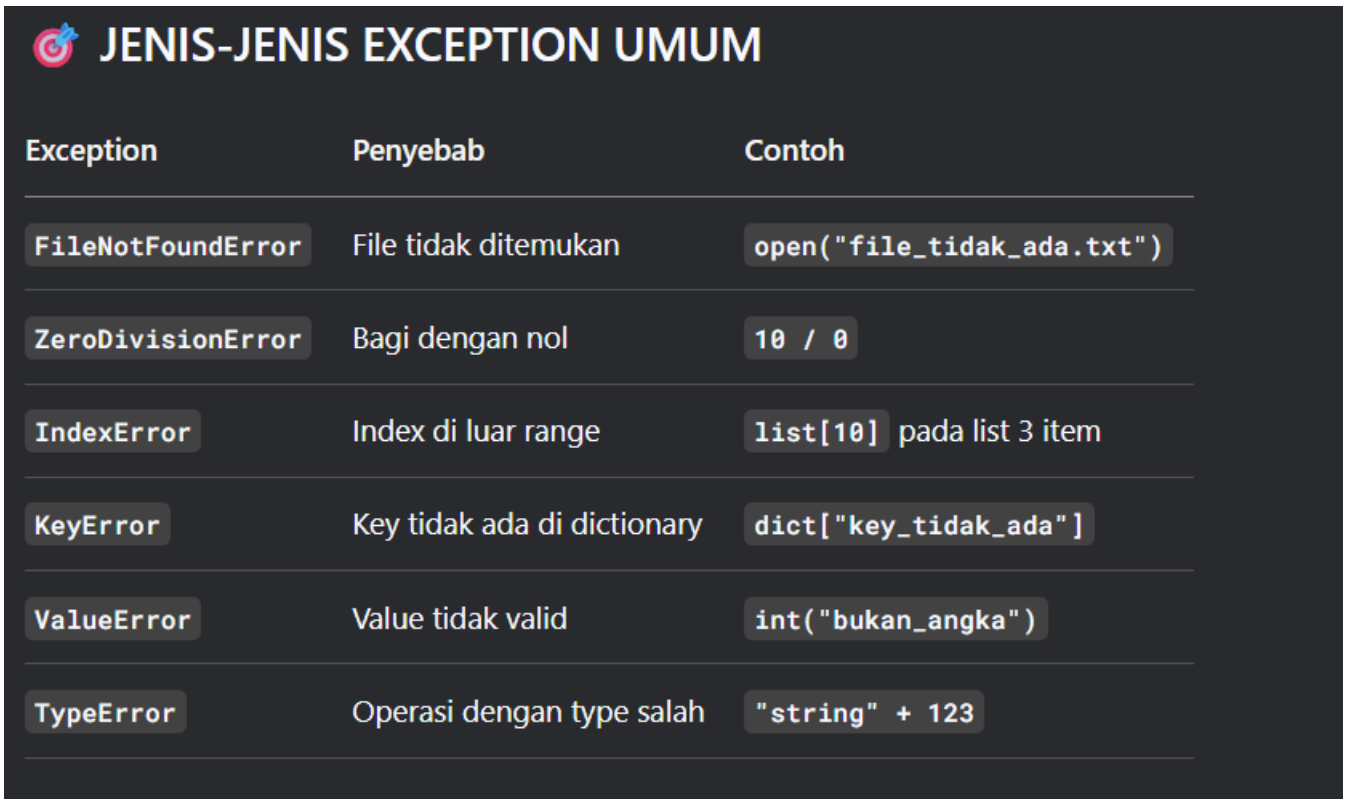
Nah di exception ini bisa juga mendefinisikan specific error yang dijalankan, contoh:

```
def bagi_angka(pembilang, penyebut):
  try:
    hasil = pembilang / penyebut
    print(f"Hasil: {hasil}")
    return hasil
  except ZeroDivisionError:
    print("Error: Tidak bisa bagi dengan nol!")
```

```
    return None
  except TypeError:
    print("Error: Input harus angka!")
    return None

# Test cases
bagi_angka(10, 2)   # ✅ Hasil: 5.0
bagi_angka(10, 0)   # ✅ "Error: Tidak bisa bagi dengan nol!"
bagi_angka(10, "a") # ✅ "Error: Input harus angka!"
```

Si ZeroDivisionError dan TypeError ini Adalah specific error, kalo gk disertakan specific error, Python akan menangkap **semua exception** yang terjadi dalam try block. Ada berbagai macam jenis exception, berikut Adalah yang umum:



Ada pun specific error lainya yaitu Exception, Exception ini dia dipake buat nerima segala macam penyebab error, contoh pemakaiannya untuk mengetahui error apa yang muncul:

```
try:
  file = open("tidak_ada.txt", "r")
except Exception as e:
  print(f"Error type: {type(e).__name__}")
  print(f"Error message: {e}")
  print(f"Error details: {e.args}")
```

```
#output
Error type: FileNotFoundError
Error message: [Errno 2] No such file or directory: 'tidak_ada.txt'
Error details: (2, 'No such file or directory')
```

Ada pun penerapan lainnya dalam exception, contohnya:

```
from numbers import Number

def tambah(a,b): # buat fungsi tambah
  if not isinstance(a,Number) or not isinstance(b,Number): #kalau a dan b bukan angka akan muncul error dalam raise ini
    raise "input harus angka"
  return a+b

#semisal tanpa if dan langsung return
print(tambah(“b”,”v”)) #outputnya bv

#kalo pake raise dia akan langsung error
```

saya juga belajar mengenai match statement fitur baru di python 3.12 yaitu match statement strukturnya seperti di bawah

Analoginya:

Seperti **kunci dan gembok** - Anda mencocokkan pola (pattern) dengan data yang ada.

Basic Syntax:

python Copy Download

```python
match value:
    case pattern1:
        # action1
    case pattern2:
        # action2
    case _:
        # default action
```

Contoh:

```
status = 404

match status:
  case 200:
    print("Success")
  case 404:
    print("Not Found")
  case 500:
    print("Server Error")
  case _:
    print("Unknown status")


outputnya
Not Found
```

Mirip if else tapi

💡 KAPAN MENGGUNAKAN `match`?

Gunakan `match` ketika:

- Memiliki banyak kondisi complex
- Bekerja dengan nested data structures
- Butuh pattern matching yang powerful

Gunakan `if-else` ketika:

- Kondisi sederhana
- Hanya 2-3 kondisi
- Compatibility dengan Python versi lama