



Lektionstillfälle 12

"Felsökning, felhantering och debuggning"

Utbildare: Robert Westin

NACKADEMIN

Kort summering av föregående lektion/ev. lektioner

Föregående lektion:

- Objektorienterad utveckling
- Klasser, instanser
- Metoder
- Instansvariabler
- Klassvariabler

Lektionstillfällets mål och metod

Mål med lektionen:

- Grundläggande problemlösning och felsökning för programmering
Felsökning av kod

Lektionens arbetsmetod/er:

- Exempel, föreläsning och labbar


Begreppsgenomgång

- Felsökning
- Felhantering
- Debugger
- "Exception" - undantag
- "try" och "except"
- "raise"
- Feltyper
- Exception hierarchy

Olika typer av fel

- Syntaxfel
 - "syntax error"
 - programmet kan inte fortsätta
- Körningsfel
 - "runtime error"
 - Det saknas t.ex en fil, som gör att programmet misslyckas
- Logikfel
 - Lösningen är felaktig men startar
 - t.ex. Utdata till en fil är inte vad som förväntades

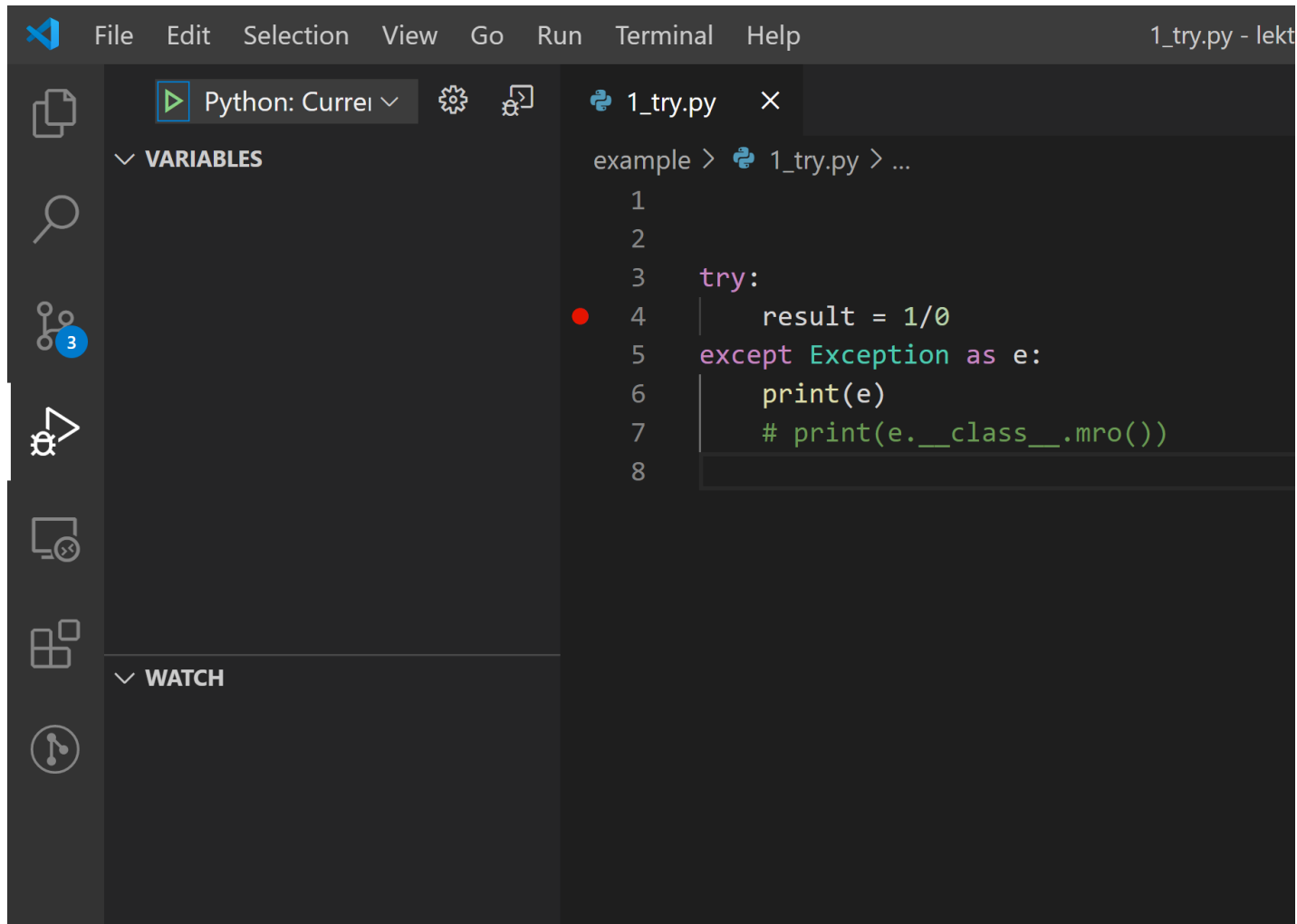
Exception exempel

```
example >  1_try.py > ...  
1  
2 def division(x, y):  
3     try:  
4         result = x / y  
5         return result  
6     except ZeroDivisionError as b:  
7         print(b)  
8     except Exception as e:  
9         print(e)  
10        print(e.__class__.mro())  
11
```

<https://docs.python.org/3/tutorial/errors.html#errors-and-exceptions>

NACKADEMIN

Debugger i VS Code

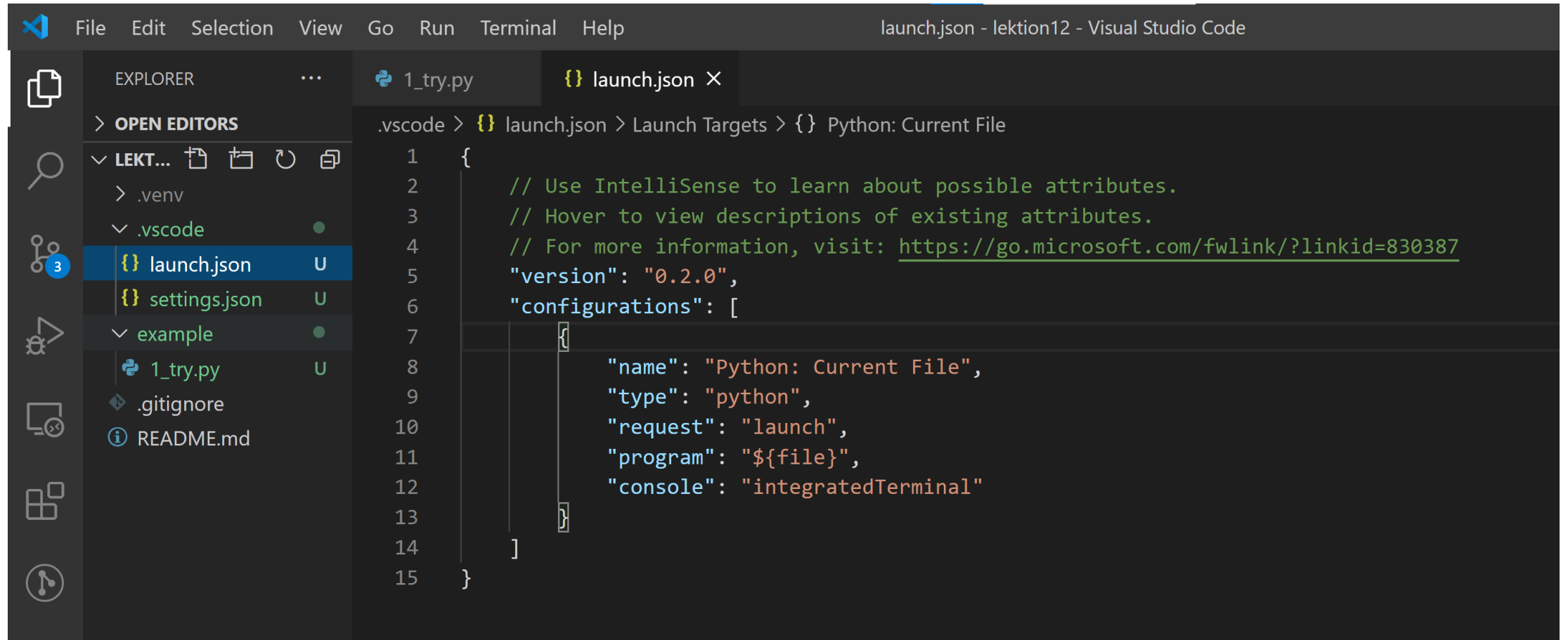


Röda pricken är en
"breakpoint"

.

.

Debugger i VS Code



Try statement

8.4. The try statement

The `try` statement specifies exception handlers and/or cleanup code for a group of statements:

```
try_stmt ::= try1_stmt | try2_stmt
try1_stmt ::= "try" ":" suite
              ("except" [expression ["as" identifier]] ":" suite)+
              ["else" ":" suite]
              ["finally" ":" suite]
try2_stmt ::= "try" ":" suite
              "finally" ":" suite
```

Källa: https://docs.python.org/3/reference/compound_stmts.html#the-try-statement

Raise

7.8. The raise statement

```
raise_stmt ::= "raise" [expression ["from" expression]]
```

If no expressions are present, `raise` re-raises the last exception that was active in the current scope. If no exception is active in the current scope, a `RuntimeError` exception is raised indicating that this is an error.

```
example > 2_raise.py > ...
1  def something_went_wrong():
2      try:
3          raise Exception("something went wrong")
4      except Exception as e:
5          print(e)
6
7
8  something_went_wrong()
9
```

https://docs.python.org/3/reference/simple_stmts.html#the-raise-statement

NACKADEMIN

Exception hierarchy

Exception hierarchy

The class hierarchy for built-in exceptions is:

```
BaseException
+-- SystemExit
+-- KeyboardInterrupt
+-- GeneratorExit
+-- Exception
    +-- StopIteration
    +-- StopAsyncIteration
    +-- ArithmeticError
        |   +-- FloatingPointError
        |   +-- OverflowError
        |   +-- ZeroDivisionError
    +-- AssertionError
    +-- AttributeError
    +-- BufferError
    +-- EOFError
    +-- ImportError
        |   +-- ModuleNotFoundError
    +-- LookupError
```

```
try:
    result = 1/0
except Exception as e:
    print(e)
```

<https://docs.python.org/3/library/exceptions.html#exception-hierarchy>

NACKADEMIN

Felsökningsstrategi

- Övning Diskutera:
 - 1. Vilka verktyg finns i IDE:n för att hitta och förstå fel?
 - 2. Vart brukar ni leta svar utanför IDE:n?
 - 3. Hitta ett exempel på fel som tagit lång tid att felsöka, utvärdera om något från steg 1 och 2 skulle ha underlättat felsökningen

Felsökning av logikfel

- Använd debuggern och kolla värden steg för steg
- Skriv "pseudo code" vad programmet gör
- Testa olika indata
- Om koden är komplex, försök skriv den i mindre delar

Generell felsökningsstrategi

- Använd linters, formatters, extensions osv.
- Läs alltid felmeddelandet i terminalen noga
- Använd stackoverflow, google och exempelvis docs
- Kolla alltid upp dokumentationen när du använder moduler, funktioner och datatyper du är osäker på.
- Använd debuggern för att förstå programflödet
- Skriv ut på lämpliga ställen
- Skriv bra felhantering

Summering av dagens lektion

- Felsökning
 - IDE
 - Debugger
-
- Reflektioner?

Framåtblick inför nästa lektion

- Inledning av slutuppgift
- Repetera det ni har gjort hittils
- Fokusera på ren enkel kod
- Objektorientering