파이썬 프로그램 만들기

(Durable Rules 패키지)

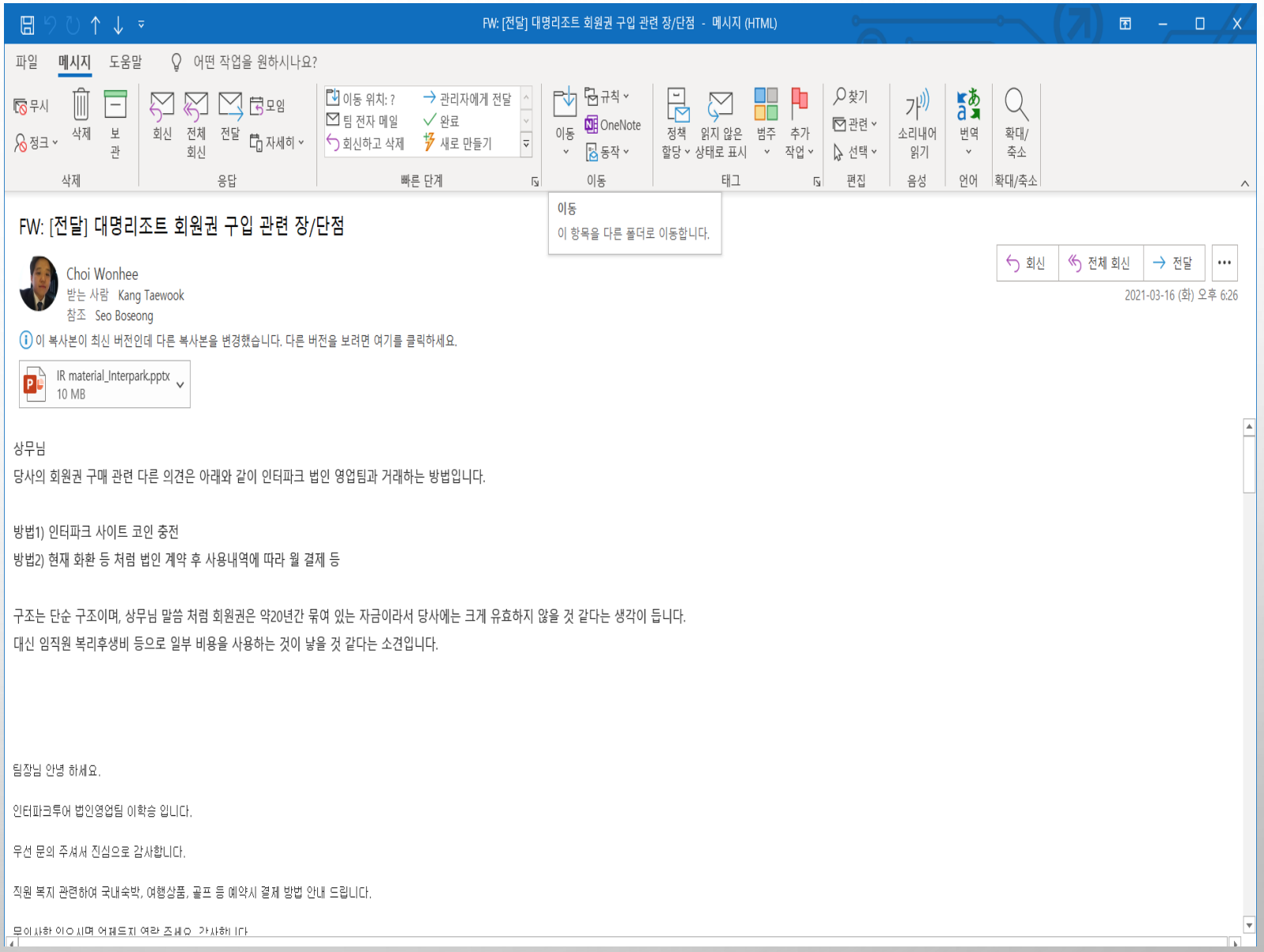
2021. 04. 01

최 원 희

Spring Cloud

회사 내부 임직원 대상 복리후생 검토

- 휴가비 지원
- 출장비 및 식대 등
- 상여금
- 출장비
- 기타 여비 규정 등



Untitled4.ipynb ☆

파일 수정 보기 삽입 런타임 도구 도움말 모든 변경사항이 저장됨

댓글 공유 원희

+ 코드 + 텍스트 RAM 디스크 수정 가능

```
[33]  !pip install durable_rules
      from durable.lang import *

      psum = 0
      dsum = 0

      def paySum(pay):
        psum += pay

      def daySum(day):
        dsum += day

        with ruleset('employee'):
          @when_all(c.first << (m.predicate == '나는') & (m.object == '휴가를 간다'),
                    (m.predicate == '나는') & (m.object == '팀장이다') & (m.subject == c.first.subject))
          def A(c):
              c.assert_fact({ 'subject': c.first.subject, 'predicate': '휴가비', 'object': 'pay100' })
              c.assert_fact({ 'subject': c.first.subject, 'predicate': '휴가일수', 'object': 'day5' })

          @when_all(c.first << (m.predicate == '나는') & (m.object == '휴가를 간다'),
                    (m.predicate == '나는') & (m.object == '팀장이다') & (m.subject == c.first.subject))
          def B(c):
              c.assert_fact({ 'subject': c.first.subject, 'predicate': '휴가비', 'object': 'pay50' })
              c.assert_fact({ 'subject': c.first.subject, 'predicate': '휴가일수', 'object': 'day3' })

          @when_all(c.first << (m.predicate == '친구랑') & (m.object == '휴가를 간다'),
                    (m.predicate == '나는') & (m.object == '팀장이다') & (m.subject == c.first.subject))
          def C(c):
              c.assert_fact({ 'subject': c.first.subject, 'predicate': '휴가비', 'object': 'pay30' })
              c.assert_fact({ 'subject': c.first.subject, 'predicate': '휴가일수', 'object': 'day2' })
```

Untitled4.ipynb

파일 수정 보기 삽입 런타임 도구 도움말 모든 변경사항이 저장됨

댓글 공유 원희

+ 코드 + 텍스트

RAM 디스크 수정 가능

[33]

```
@when_all(c.first << (m.predicate == '부모님과 함께') & (m.object == '휴가를 간다'),
          (m.predicate == '나는') & (m.object == '팀장이다') & (m.subject == c.first.subject))
def D(c):
    c.assert_fact({ 'subject': c.first.subject, 'predicate': '휴가비', 'object': 'pay50' })
    c.assert_fact({ 'subject': c.first.subject, 'predicate': '휴가일수', 'object': 'day3' })

@when_all(c.first << (m.predicate == '직장 동료들과 같이') & (m.object == '휴가를 간다'),
          (m.predicate == '나는') & (m.object == '팀장이다') & (m.subject == c.first.subject))
def E(c):
    c.assert_fact({ 'subject': c.first.subject, 'predicate': '휴가비', 'object': 'pay10' })
    c.assert_fact({ 'subject': c.first.subject, 'predicate': '휴일일수', 'object': 'day1' })

@when_all(c.first << (m.predicate == '가족과 함께') & (m.object == '휴가를 간다'),
          (m.predicate == '나는') & (m.object == '팀장이다') & (m.subject == c.first.subject))
def F(c):
    c.assert_fact({ 'subject': c.first.subject, 'predicate': '휴가비', 'object': 'pay200' })
    c.assert_fact({ 'subject': c.first.subject, 'predicate': '휴일일수', 'object': 'day5' })

@when_all(c.first << (m.predicate == '아내와 함께') & (m.object == '휴가를 간다'),
          (m.predicate == '나는') & (m.object == '팀장이다') & (m.subject == c.first.subject))
def G(c):
    c.assert_fact({ 'subject': c.first.subject, 'predicate': '휴가비', 'object': 'pay30' })
    c.assert_fact({ 'subject': c.first.subject, 'predicate': '휴일일수', 'object': 'day2' })

@when_all(c.first << (m.predicate == '혼자') & (m.object == '출장을 간다'),
          (m.predicate == '나는') & (m.object == '팀장이다') & (m.subject == c.first.subject))
def H(c):
    c.assert_fact({ 'subject': c.first.subject, 'predicate': '출장비', 'object': 'pay10' })
    c.assert_fact({ 'subject': c.first.subject, 'predicate': '출장일수', 'object': 'day2' })
```

Untitled4.ipynb

파일 수정 보기 삽입 런타임 도구 도움말 모든 변경사항이 저장됨

댓글 공유 원희

+ 코드 + 텍스트

RAM 디스크 수정 가능

```
def G(c):
    c.assert_fact({ 'subject': c.first.subject, 'predicate': '휴가비', 'object': 'pay30' })
    c.assert_fact({ 'subject': c.first.subject, 'predicate': '휴일일수', 'object': 'day2' })

@when_all(c.first << (m.predicate == '혼자') & (m.object == '출장을 간다'),
          (m.predicate == '나는') & (m.object == '팀장이다') & (m.subject == c.first.subject))
def H(c):
    c.assert_fact({ 'subject': c.first.subject, 'predicate': '출장비', 'object': 'pay10' })
    c.assert_fact({ 'subject': c.first.subject, 'predicate': '출장일수', 'object': 'day2' })

@when_all(c.first << (m.predicate == '(임원과') & (m.object == '출장을 간다'),
          (m.predicate == '나는') & (m.object == '팀장이다') & (m.subject == c.first.subject))
def I(c):
    c.assert_fact({ 'subject': c.first.subject, 'predicate': '출장비', 'object': 'pay30' })
    c.assert_fact({ 'subject': c.first.subject, 'predicate': '출장일수', 'object': 'day1' })

@when_all(c.first << (m.predicate == '팀 전체') & (m.object == '출장을 간다'),
          (m.predicate == '나는') & (m.object == '팀장이다') & (m.subject == c.first.subject))
def J(c):
    c.assert_fact({ 'subject': c.first.subject, 'predicate': '출장비', 'object': 'pay100' })
    c.assert_fact({ 'subject': c.first.subject, 'predicate': '출장일수', 'object': 'day5' })

@when_all(c.first << (m.predicate == '점심식대를 지급한다') & (m.object == '식대'))
def K(c):
    c.assert_fact({ 'subject': c.first.subject, 'predicate': '식대', 'object': 'pay1' })

@when_all(c.first << (m.predicate == '팀 전체') & (m.object == '회식이다'))
def L(c):
    c.assert_fact({ 'subject': c.first.subject, 'predicate': '회식대', 'object': 'pay10' })

@when_any(c.first << (m.predicate == '나는') & ((m.object == '팀장이다') | (m.object == '매니저다')))
```

Untitled4.ipynb

파일 수정 보기 삽입 런타임 도구 도움말 모든 변경사항이 저장됨

댓글 공유 원희

+ 코드 + 텍스트

RAM 디스크 수정 가능

[33]

```
def pay15(c):
    print('Fact: {0}에게 {1}은 15만원을 지급한다'.format(c.m.subject, c.m.predicate))
    psum += 15

@when_any((m.object == 'day10'))
def day6(c):
    print('Fact: {0}에게 {1}는 5일 및 휴가비를 지급합니다'.format(c.m.subject, c.m.predicate))
    dsum += 10

@when_any((m.object == 'day6'))
def day6(c):
    print('Fact: {0}에게 {1}는 6일 입니다'.format(c.m.subject, c.m.predicate))
    dsum += 6

@when_any((m.object == 'day2'))
def day2(c):
    print('Fact: {0}에게 {1}는 2일 입니다'.format(c.m.subject, c.m.predicate))
    daySum(2)

@when_any((m.object == 'day1'))
def day1(c):
    print('Fact: {0}에게 {1}는 1일 입니다'.format(c.m.subject, c.m.predicate))
    dsum += 1

@when_any((m.object == 'point'))
def point(c):
    print('Fact: {0}에게 {1}에는 복지포인트 30만원을 지급한다'.format(c.m.subject, c.m.predicate))
    psum += 30

@when_any((m.object == 'coupon'))
def coupon(c):
    print('Fact: {0}에게 {1}에는 법인 차량을 지급한다'.format(c.m.subject, c.m.predicate))
```

0초 오후 6:40에 완료됨

Untitled4.ipynb

파일 수정 보기 삽입 런타임 도구 도움말 모든 변경사항이 저장됨

댓글 공유 원희

+ 코드 + 텍스트 RAM 디스크 수정 가능

[33]
```
            print('Fact: {0}에게 {1}은 1일 입니다'.format(c.m.subject, c.m.predicate))
            dsum += 1

        @when_any((m.object == 'point'))
        def point(c):
            print('Fact: {0}에게 {1}에는 복지포인트 30만원을 지급한다'.format(c.m.subject, c.m.predicate))
            psum += 30

        @when_any((m.object == 'coupon'))
        def coupon(c):
            print('Fact: {0}에게 {1}에는 법인 차량을 지급한다'.format(c.m.subject, c.m.predicate))
            psum += 2

        #@when_all(+m.subject)
        #def output(c):
        #    print('assert_fact 등록내용: {0} {1} {2}'.format(c.m.subject, c.m.predicate, c.m.object))
```

```
Requirement already satisfied: durable_rules in /usr/local/lib/python3.7/dist-packages (2.0.28)
```

```
assert_fact('employee', { 'subject': '최원희', 'predicate': '팀장이다', 'object': '휴가를 간다' })
assert_fact('employee', { 'subject': '최원희', 'predicate': '나는', 'object': '팀장이다' })
assert_fact('employee', { 'subject': '최원희', 'predicate': '팀전체', 'object': '회식이다' })
assert_fact('employee', { 'subject': '최원희', 'predicate': '상여금', 'object': '연말 성과 보너스' })
assert_fact('employee', { 'subject': '최원희', 'predicate': '가족과 함께', 'object': '휴가를 간다' })
assert_fact('employee', { 'subject': '최원희', 'predicate': '임원과', 'object': '출장을 간다' })

print("팀장이 받는 총액은 {0}만원이며, 연차외의 총 경조휴무일은 {1}일 입니다".format(psum, dsum))
```

Untitled4.ipynb

파일 수정 보기 삽입 런타임 도구 도움말 모든 변경사항이 저장됨

댓글 공유 원희

+ 코드 + 텍스트

RAM 디스크 수정 가능

```
Requirement already satisfied: durable_rules in /usr/local/lib/python3.7/dist-packages (2.0.28)
```

```
assert_fact('employee', { 'subject': '최원희', 'predicate': '팀장이다', 'object': '휴가를 간다' })
assert_fact('employee', { 'subject': '최원희', 'predicate': '나는', 'object': '팀장이다' })
assert_fact('employee', { 'subject': '최원희', 'predicate': '팀전체', 'object': '회식이다' })
assert_fact('employee', { 'subject': '최원희', 'predicate': '상여금', 'object': '연말 성과 보너스' })
assert_fact('employee', { 'subject': '최원희', 'predicate': '가족과 함께', 'object': '휴가를 간다' })
assert_fact('employee', { 'subject': '최원희', 'predicate': '임원과', 'object': '출장을 간다' })

print("팀장이 받는 총액은 {0}만원이며, 연차외의 총 경조휴무일은 {1}일 입니다".format(psum, dsum))
```

```
---------------------------------------------------------------------------
MessageNotHandledException                Traceback (most recent call last)
<ipython-input-47-69f5cafa4db3> in <module>()
----> 1 assert_fact('employee', { 'subject': '최원희', 'predicate': '팀장이다', 'object': '휴가를 간다' })
      2 assert_fact('employee', { 'subject': '최원희', 'predicate': '나는', 'object': '팀장이다' })
      3 assert_fact('employee', { 'subject': '최원희', 'predicate': '팀전체', 'object': '회식이다' })
      4 assert_fact('employee', { 'subject': '최원희', 'predicate': '상여금', 'object': '연말 성과 보너스' })
      5 assert_fact('employee', { 'subject': '최원희', 'predicate': '가족과 함께', 'object': '휴가를 간다' })

                                  4 frames
/usr/local/lib/python3.7/dist-packages/durable/engine.py in _handle_result(self, result, message)
    326     def _handle_result(self, result, message):
    327         if result[0] == 1:
--> 328             raise MessageNotHandledException(message)
    329         elif result[0] == 2:
    330             raise MessageObservedException(message)

MessageNotHandledException: {'subject': '최원희', 'predicate': '팀장이다', 'object': '휴가를 간다'}
```

SEARCH STACK OVERFLOW

소스코드

```
 !pip install durable_rules
 from durable.lang import *

psum = 0
dsum = 0

def paySum(pay):
  psum += pay

def daySum(day):
  dsum += day

  with ruleset('employee'):
    @when_all(c.first << (m.predicate == '나는') & (m.object == '휴가를 간다'),
              (m.predicate == '나는') & (m.object == '팀장이다') & (m.subject == c.first.subject))
    def A(c):
        c.assert_fact({ 'subject': c.first.subject, 'predicate': '휴가비', 'object': 'pay100' })
        c.assert_fact({ 'subject': c.first.subject, 'predicate': '휴가일수', 'object': 'day5' })

    @when_all(c.first << (m.predicate == '나는') & (m.object == '휴가를 간다'),
              (m.predicate == '나는') & (m.object == '팀장이다') & (m.subject == c.first.subject))
    def B(c):
        c.assert_fact({ 'subject': c.first.subject, 'predicate': '휴가비', 'object': 'pay50' })
        c.assert_fact({ 'subject': c.first.subject, 'predicate': '휴가일수', 'object': 'day3' })

    @when_all(c.first << (m.predicate == '친구랑') & (m.object == '휴가를 간다'),
              (m.predicate == '나는') & (m.object == '팀장이다') & (m.subject == c.first.subject))
    def C(c):
        c.assert_fact({ 'subject': c.first.subject, 'predicate': '휴가비', 'object': 'pay30' })
        c.assert_fact({ 'subject': c.first.subject, 'predicate': '휴가일수', 'object': 'day2' })

    @when_all(c.first << (m.predicate == '부모님과 함께') & (m.object == '휴가를 간다'),
              (m.predicate == '나는') & (m.object == '팀장이다') & (m.subject == c.first.subject))
    def D(c):
        c.assert_fact({ 'subject': c.first.subject, 'predicate': '휴가비', 'object': 'pay50' })
        c.assert_fact({ 'subject': c.first.subject, 'predicate': '휴가일수', 'object': 'day3' })

    @when_all(c.first << (m.predicate == '직장 동료들과 같이') & (m.object == '휴가를 간다'),
              (m.predicate == '나는') & (m.object == '팀장이다') & (m.subject == c.first.subject))
    def E(c):
        c.assert_fact({ 'subject': c.first.subject, 'predicate': '휴가비', 'object': 'pay10' })
        c.assert_fact({ 'subject': c.first.subject, 'predicate': '휴일일수', 'object': 'day1' })

    @when_all(c.first << (m.predicate == '가족과 함께') & (m.object == '휴가를 간다'),
              (m.predicate == '나는') & (m.object == '팀장이다') & (m.subject == c.first.subject))
    def F(c):
        c.assert_fact({ 'subject': c.first.subject, 'predicate': '휴가비', 'object': 'pay200' })
        c.assert_fact({ 'subject': c.first.subject, 'predicate': '휴일일수', 'object': 'day5' })
```

소스코드

```
@when_all(c.first << (m.predicate == '아내와 함께') & (m.object == '휴가를 간다'),
          (m.predicate == '나는') & (m.object == '팀장이다') & (m.subject == c.first.subject))
def G(c):
    c.assert_fact({ 'subject': c.first.subject, 'predicate': '휴가비', 'object': 'pay30' })
    c.assert_fact({ 'subject': c.first.subject, 'predicate': '휴일일수', 'object': 'day2' })

@when_all(c.first << (m.predicate == '혼자') & (m.object == '출장을 간다'),
          (m.predicate == '나는') & (m.object == '팀장이다') & (m.subject == c.first.subject))
def H(c):
    c.assert_fact({ 'subject': c.first.subject, 'predicate': '출장비', 'object': 'pay10' })
    c.assert_fact({ 'subject': c.first.subject, 'predicate': '출장일수', 'object': 'day2' })

@when_all(c.first << (m.predicate == '(임원과') & (m.object == '출장을 간다'),
          (m.predicate == '나는') & (m.object == '팀장이다') & (m.subject == c.first.subject))
def I(c):
    c.assert_fact({ 'subject': c.first.subject, 'predicate': '출장비', 'object': 'pay30' })
    c.assert_fact({ 'subject': c.first.subject, 'predicate': '출장일수', 'object': 'day1' })

@when_all(c.first << (m.predicate == '팀 전체') & (m.object == '출장을 간다'),
          (m.predicate == '나는') & (m.object == '팀장이다') & (m.subject == c.first.subject))
def J(c):
    c.assert_fact({ 'subject': c.first.subject, 'predicate': '출장비', 'object': 'pay100' })
    c.assert_fact({ 'subject': c.first.subject, 'predicate': '출장일수', 'object': 'day5' })

@when_all(c.first << (m.predicate == '점심식대를 지급한다') & (m.object == '식대'))
def K(c):
    c.assert_fact({ 'subject': c.first.subject, 'predicate': '식대', 'object': 'pay1' })

@when_all(c.first << (m.predicate == '팀 전체') & (m.object == '회식이다'))
def L(c):
    c.assert_fact({ 'subject': c.first.subject, 'predicate': '회식대', 'object': 'pay10' })

@when_any(c.first << (m.predicate == '나는') & ((m.object == '팀장이다') | (m.object == '매니저다')))
def M(c):
    c.assert_fact({ 'subject': c.first.subject, 'predicate': '휴가비', 'object': 'point' })
    c.assert_fact({ 'subject': c.first.subject, 'predicate': '출장을 간다', 'object': 'coupon' })

@when_any((m.object == 'pay150') & ((m.predicate == '상여금')))
def pay150(c):
    print('Fact: {0}에게 {1}은 150만원을 지급한다{2}'.format(c.m.subject, c.m.predicate, psum))
    psum += 150
    paySum(150)

@when_any((m.object == 'pay100') & ((m.predicate == '상여금') | (m.predicate == '연말 성과 보너스')))
def pay100(c):
    print('Fact: {0}에게 {1}은 100만원을 지급한다'.format(c.m.subject, c.m.predicate))
    psum += 100
    paySum(100)
```

소스코드

```
@when_any((m.object == 'pay75') & ((m.predicate == '상여금') | (m.predicate == '연말 성과 보너스')))
  def pay75(c):
      print('Fact: {0}에게 {1}은 75만원을 지급한다'.format(c.m.subject, c.m.predicate))
      psum += 75

  @when_any((m.object == 'pay50') & ((m.predicate == '상여금') | (m.predicate == '연말 성과 보너스')))
  def pay50(c):
      print('Fact: {0}에게 {1}은 50만원을 지급한다'.format(c.m.subject, c.m.predicate))
      psum += 50

  @when_any((m.object == 'pay30') & ((m.predicate == '상여금') | (m.predicate == '연말 성과 보너스')))
  def pay30(c):
      print('Fact: {0}에게 {1}은 30만원을 지급한다'.format(c.m.subject, c.m.predicate))
      psum += 30

  @when_any((m.object == 'pay20') & ((m.predicate == '상여금') | (m.predicate == '연말 성과 보너스') | (m.predicate == '팀장')))
  def pay20(c):
      print('Fact: {0}에게 {1}은 20만원을 지급한다'.format(c.m.subject, c.m.predicate))
      psum += 20

  @when_any((m.object == 'pay15') & ((m.predicate == '상여금') | (m.predicate == '연말 성과 보너스') | (m.predicate == '팀장')))
  def pay15(c):
      print('Fact: {0}에게 {1}은 15만원을 지급한다'.format(c.m.subject, c.m.predicate))
      psum += 15

  @when_any((m.object == 'day10'))
  def day6(c):
      print('Fact: {0}에게 {1}는 5일 및 휴가비를 지급합니다'.format(c.m.subject, c.m.predicate))
      dsum += 10

  @when_any((m.object == 'day6'))
  def day6(c):
      print('Fact: {0}에게 {1}는 6일 입니다'.format(c.m.subject, c.m.predicate))
      dsum += 6

  @when_any((m.object == 'day2'))
  def day2(c):
      print('Fact: {0}에게 {1}는 2일 입니다'.format(c.m.subject, c.m.predicate))
      daySum(2)

  @when_any((m.object == 'day1'))
  def day1(c):
      print('Fact: {0}에게 {1}는 1일 입니다'.format(c.m.subject, c.m.predicate))
      dsum += 1
```

```
@when_any((m.object == 'point'))
  def point(c):
      print('Fact: {0}에게 {1}에는 복지포인트 30만원을 지급한다'.format(c.m.subject, c.m.predicate))

      psum += 30

  @when_any((m.object == 'coupon'))
  def coupon(c):
      print('Fact: {0}에게 {1}에는 법인 차량을 지급한다'.format(c.m.subject, c.m.predicate))
      psum += 2

  #@when_all(+m.subject)
  #def output(c):
  #    print('assert_fact 등록내용: {0} {1} {2}'.format(c.m.subject, c.m.predicate, c.m.object))
```

소스코드