[COM1002]
프로그래밍1

Variables and Functions

#02. 변수와 함수

김현하

한양대학교 ERICA 소프트웨어학부

2021.9.14.
2021년도 2학기

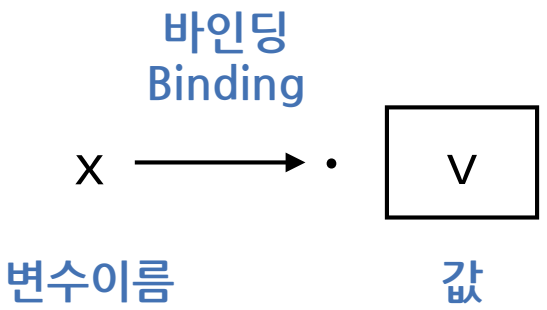
목차

- 변수
  - 변수 지정, 변수 작명 규칙, 지정문 실행, 키보드 입력, 지정문 실행 순서, 예약어, 주석
- 함수
  - 람다 요약, 함수 정의와 호출 : 문법과 의미, 함수 만들기 실전

변수
Variable

프로그램 실행 중 생기는 계산 값을
나중에 다시 사용하기 위해서
지어두는 이름

네임스페이스
Namespace



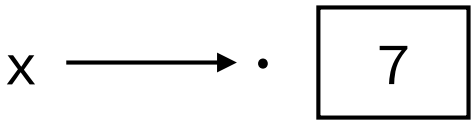
변수variable 지정assignment

지정문 <변수> = <식>

- 변수 : 지어진 이름
- 지정 : <식>을 계산한 값에 이름을 지어주는 과정

- [Python 인터프리터]
  - x = 3 + 4
  - x = x + 2
  - pooh

오른쪽 수식 3 + 4를 계산한 결과 값인 7을 메모리 적당한 장소를 배정받아 저장하고, 그 곳을 변수 x로 지정한다.



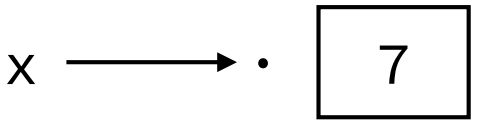
지정문
Assignment

```
x = 3 + 4
```

```
x = x + 2
```

```
x = "Freedom"
```

네임스페이스
Namespace



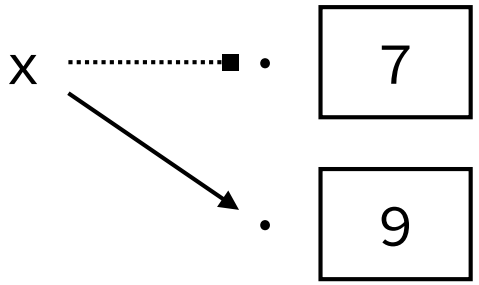
지정문
Assignment

```
x = 3 + 4
```

```
x = x + 2
```

```
x = "Freedom"
```

네임스페이스
Namespace



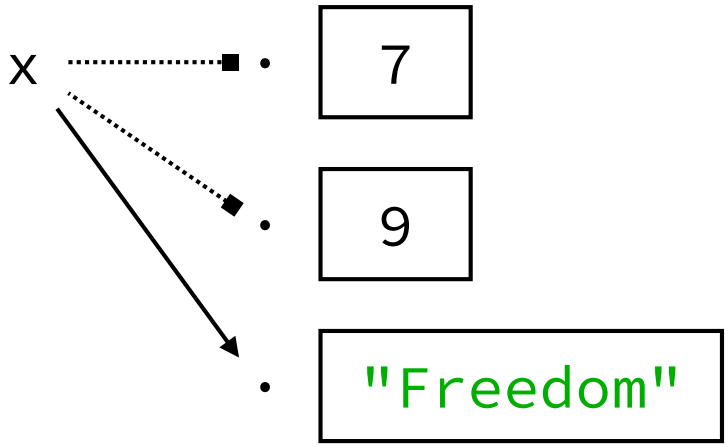
지정문
Assignment

```
x = 3 + 4

x = x + 2

x = "Freedom"
```

네임스페이스
Namespace



변수 작명 규칙

- 규칙
  - 문자 (a-z, A-Z), 숫자(0-9), 밑줄(_)의 조합으로 만들어야 함
  - 숫자로 시작할 수 없음
  - Python coding convention
- [Python 인터프리터]
  - `susieQ = 1958`
  - `python1st = 1991`
  - `pythonProgramming = "easy"`
  - `python_programming = "fast"`
  - `1stpython`

- 값의 특징을 잘 나타내는 명사 또는 명사구를 사용
- 나름의 작명 규칙을 정하고 일관성을 유지
- 관습을 따름 (일반 변수는 소문자로 시작)

원의 면적 구하기

$$\pi \times r^2$$

인터프리터에서 하나씩 실행

지정문 시행

- [Python 인터프리터]
  - `print(3.14 * 5 ** 2)`
- [Python 인터프리터]
  - `pi = 3.14`
  - `radius = 5`
  - `area = pi * radius ** 2`
  - `print(area)`

인터프리터에서 하나씩 실행

지정문 시행

- [Python 인터프리터]
  - math.pi
- [Python 인터프리터]
  - import math
  - radius = 5
  - area = math.pi * radius ** 2
  - print(area)

Python 표준 라이브러리 : math

인터프리터에서 하나씩 실행

지정문 시행

- [Python 인터프리터]
  - `from math import pi`
  - `pi`
- [Python 인터프리터]
  - `radius = 5`
  - `area = pi * radius ** 2`
  - `print(area)`

Python 표준 라이브러리 : math

인터프리터에서 하나씩 실행

지정문 시행

- [Python 인터프리터]
  - `from math import *`
  - `pi`
- [Python 인터프리터]
  - `radius = 5`
  - `area = pi * radius ** 2`
  - `print(area)`

Python 표준 라이브러리 : math

통합개발환경

Integrated Development Environment

IDLE 편집기 활용

프로그램을 파일에 저장하여 한꺼번에 실행

프로그램을 파일에 저장하여 한꺼번에 실행

지정문 시행

- [Python 인터프리터] areacircle.py

```
1  radius = 25.36
2  from math import pi
3  area = pi * radius ** 2
4  print(area)
```

키보드 입력

- [Python 인터프리터]

```
1  x = input()
2  print(x)
```

키보드 입력

- [Python 인터프리터] areacircle.py

```
1  radius = input()
2  from math import pi
3  area = pi * radius ** 2
4  print(area)
```

키보드 입력

- [Python 인터프리터] areacircle.py

```
1  radius = float(input())
2  from math import pi
3  area = pi * radius ** 2
4  print(area)
```

키보드 입력

- [Python 인터프리터] areacircle.py

```
1  radius = float(input("Enter  the radius : "))
2  from math import pi
3  area = pi * radius ** 2
4  print("The area of a circle with radius", radius, "is", area)
```

정밀도 다듬기

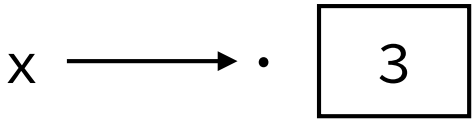
반올림하기

키보드 입력

- [Python 인터프리터]
  - `round(1.49)`
  - `round(1.5)`
  - `round(2.5)`
  - `round(2.51)`
  - `round(2.356, 2)`

지정문 실행 순서

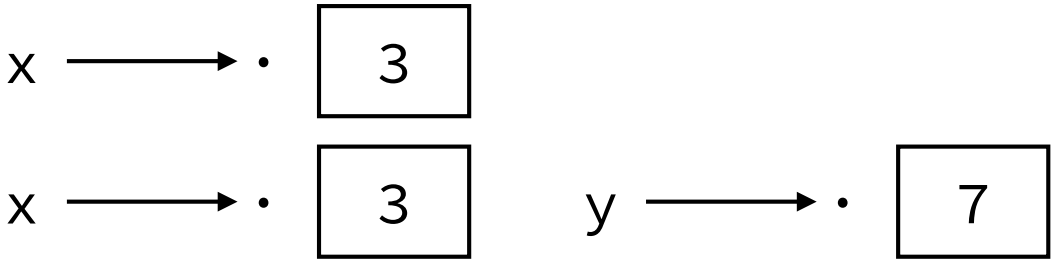
- [Python 인터프리터]
  - x = 3



  - y = 7
  - print(x, y)
  - x = y
  - y = x
  - print(x, y)

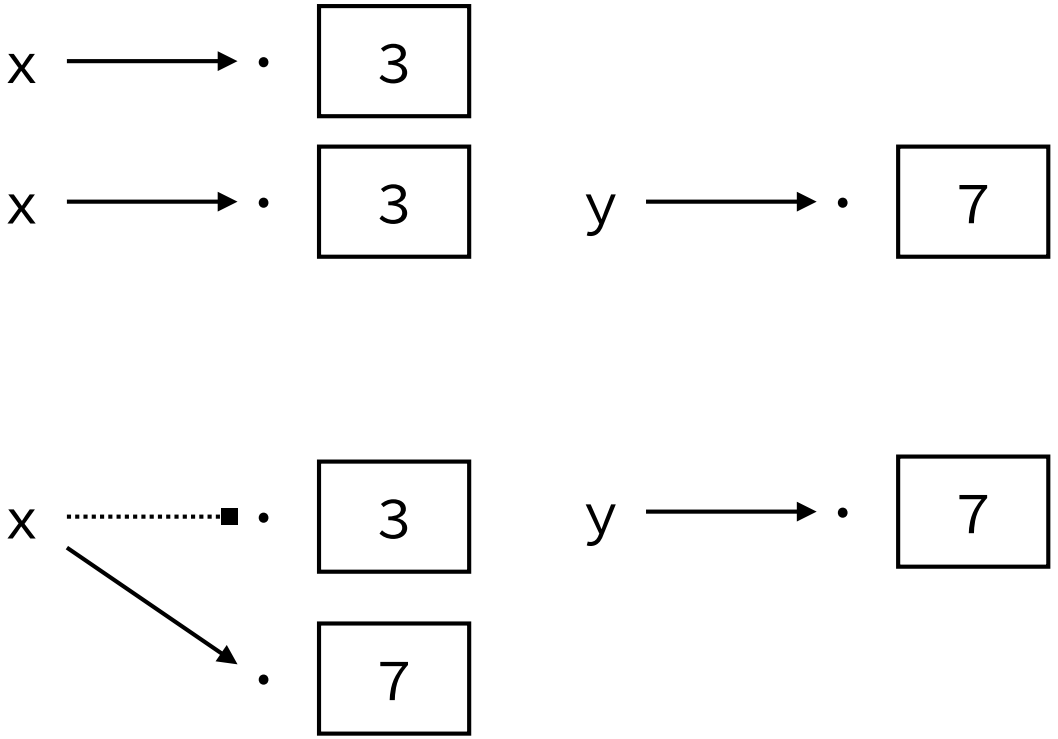
지정문 실행 순서

- [Python 인터프리터]
  - `x = 3`
  - `y = 7`
  - `print(x, y)`
  - `x = y`
  - `y = x`
  - `print(x, y)`



지정문 실행 순서

- [Python 인터프리터]
  - x = 3
  - y = 7
  - print(x, y)
  - x = y
  - y = x
  - print(x, y)

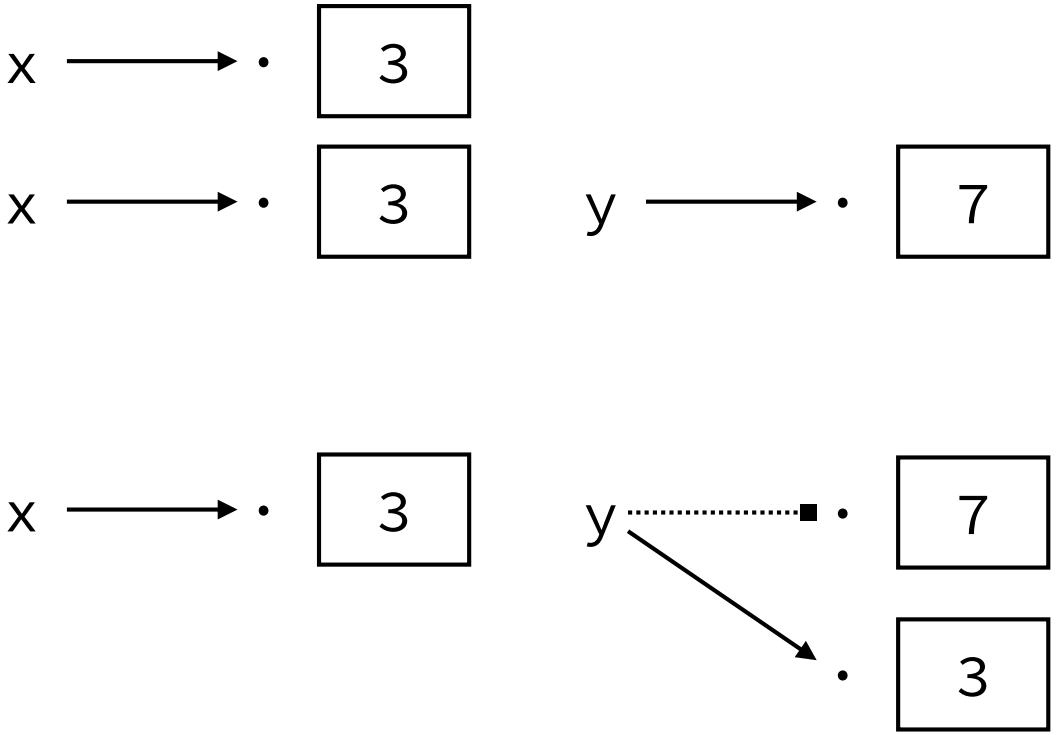


지정문 실행 순서

- [Python 인터프리터]
  - `x = 3`
  - `y = 7`
  - `print(x, y)`
  - **`y = x`**
  - **`x = y`**
  - `print(x, y)`

지정문 실행 순서

- [Python 인터프리터]
  - x = 3
  - y = 7
  - print(x, y)
  - y = x
  - x = y
  - print(x, y)

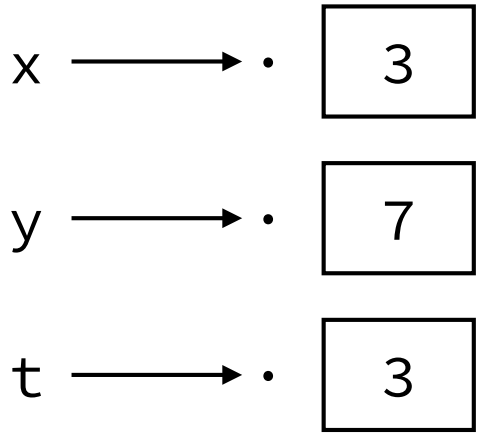


지정문 실행 순서

- [Python 인터프리터]
  - `x = 3`
  - `y = 7`
  - `print(x, y)`
  - **`t`** `= x`
  - `x = y`
  - `y =` **`t`**
  - `print(x, y)`

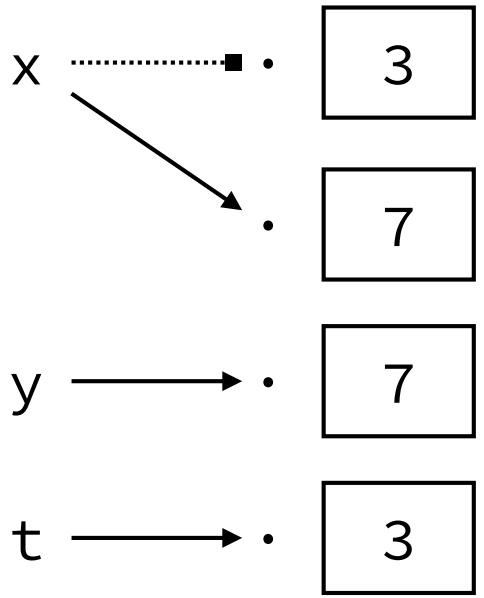
지정문 실행 순서

- [Python 인터프리터]
  - x = 3
  - y = 7
  - print(x, y)
  - t = x
  - x = y
  - y = t
  - print(x, y)



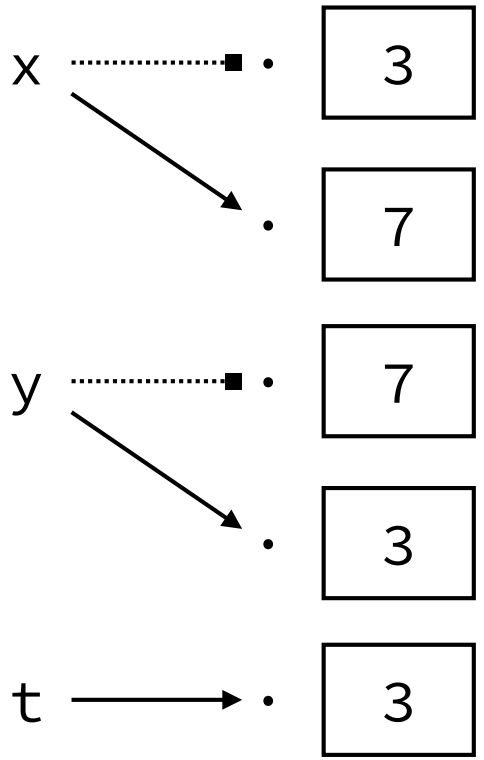
지정문 실행 순서

- [Python 인터프리터]
  - `x = 3`
  - `y = 7`
  - `print(x, y)`
  - `t = x`
  - `x = y`
  - `y = t`
  - `print(x, y)`



지정문 실행 순서

- [Python 인터프리터]
  - `x = 3`
  - `y = 7`
  - `print(x, y)`
  - `t = x`
  - `x = y`
  - `y = t`
  - `print(x, y)`



지정문 실행 순서

- [Python 인터프리터]
  - `x = 3`
  - `y = 7`
  - `print(x, y)`
  - `t = x`
  - `x = y`
  - `y = t`
  - `print(x, y)`

- [Python 인터프리터]
  - `x = 3`
  - `y = 7`
  - `print(x, y)`
  - `x, y = y, x` **동시지정**
  - `print(x, y)`
  - `x = y = z = 0` **복수지정**
  - `print(x, y, z)`

예약어 reserved words

False	await	else	import	pass
None	break	except	in	raise
True	class	finally	is	return
and	continue	for	lambda	try
as	def	from	nonlocal	while
assert	del	global	not	with
async	elif	if	or	yield

주석comments

- [Python 인터프리터] areacircle.py

```
1  # Calculate the area of circle
2  # in: radius from standard input
3  # out: area of circle to standard output
4  radius = float(input("Enter  the radius : "))
5  from math import pi
6  area = pi * radius ** 2 # calculate the area of the circle
7  print("The area of a circle with radius", radius, "is", area)
```

함수

함수function

- 함수
  - 프로그램 코드에서 독립적으로 존재 가치가 있고 재사용이 가능한 부분에 이름을 붙인 것
  - 내장 함수 (built-in function)
    - `print()`, `input()`, `int()`, `float()`, `str()`, `round()`, ...
  - 모듈module : 함수를 종류 별로 모아둔 것

함수function

- [Python 인터프리터]
  - `from math import pi`
  - `radius = 3`
  - `print(pi * radius ** 2)`
  - `radius = 5`
  - `print(pi * radius ** 2)`
  - `radius = 9`
  - `print(pi * radius ** 2)`

람다 요약lambda abstraction

```
lambda <변수> : <식>
```

- <변수> = 파라미터parameter
- [Python 인터프리터]
  - `from math import pi`
  - `lambda radius : pi * radius ** 2`
    파라미터 몸체body
  - `(lambda radius : pi * radius ** 2)`
    클로져closure

람다식에 대입lambda application

(lambda <변수> : <식>) (<인수>)

- <인수>argument = <식>
- [Python 인터프리터]
  - `from math import pi`
  - `(lambda radius : pi * radius ** 2)(3)`

람다 요약lambda abstraction

- <변수> = 파라미터parameter
- <인수>argument
- [Python 인터프리터]
  - `from math import pi`
  - `lambda radius : pi * radius ** 2`
  - `(lambda radius : pi * radius ** 2)(3)`
  - `area_circle = lambda radius : pi * radius ** 2`
  - `area_circle(3)`
  - `area_circle(5)`
  - `area_circle(9)`

`lambda` <변수> : <식>

(`lambda` <변수> : <식>) (<인수>)

함수 정의와 호출 : 문법과 의미

함수 정의

```
def <함수이름>(<변수1>, <변수2>, ..., <변수n>):
____<블록>
```

- 〈함수이름〉 : 일반 변수 이름과 같은 작명 규칙
- 〈변수〉 : (형식formal) 파라미터parameter, 0개 이상 필요한 만큼 나열
- 〈블록〉
  - 한 줄 이상의 코드로 구성된 블록block
  - 같은 간격으로 들여쓰기indentation를 해서 코드의 영역scope을 표시 (Python은 4칸을 권장)
  - 정의할 때는 〈블록〉을 실행하지 않음
- return : 함수에서 만들어 낸 정보를 결과로 내어줄 때 사용
  - `return` 〈식〉
  - 프로시저procedure : `return`이 없는 함수 (예: `print`)

함수 정의와 호출 : 문법과 의미

함수 호출 <함수이름>(<식>$_1$, <식>$_2$, ..., <식>$_n$)

- 이미 정의된 함수만 호출 가능
- <식> : 실제 파라미터actual parameter 혹은 인수argument
  - 함수를 호출하면
    - 모든 <식>을 계산
    - 각각의 <식>을 정의한 함수의 형식 파라미터 변수에 지정
    - 함수의 <블록>을 실행
    - <블록> 실행 중 `return` <식> 을 만나면, 계산한 <식>을 전달
      - `x = int("2021")`

함수 만들기 실전

- [Python 인터프리터]

```
def area_circle(radius):
    return pi * radius ** 2
```

```
print(area_circle(3))
```

```
print(area_circle(5))
```

```
print(area_circle(9))
```

함수 만들기 실전

소수점 둘째자리에서 반올림

- [Python 인터프리터]

```
def area_circle(radius):
    area = pi * radius ** 2
    return round(area, 2)

print(area_circle(3))

print(area_circle(5))

print(area_circle(9))
```

함수 만들기 실전

소수점 허용 자리 수 조정

- [Python 인터프리터]

```
def area_circle(radius, n):
    area = pi * radius ** 2
    return round(area, n)
```

```
print(area_circle(3, 1))
```

```
print(area_circle(5, 2))
```

```
print(area_circle(9, 3))
```

함수 만들기 실전 - 비교

- [Python 인터프리터]
  - ```
    def area_circle(radius, n):
        area = pi * radius ** 2
        return round(area, n)
    ```
  - ```
    area_circle(3, 1)
    ```
  - ```
    area_circle(5, 2)
    ```
  - ```
    area_circle(9, 3)
    ```

- [Python 인터프리터]
  - ```
    radius = 3
    round(pi * radius ** 2, 1)
    ```
  - ```
    radius = 5
    round(pi * radius ** 2, 2)
    ```
  - ```
    radius = 9
    round(pi * radius ** 2, 3)
    ```
    ```