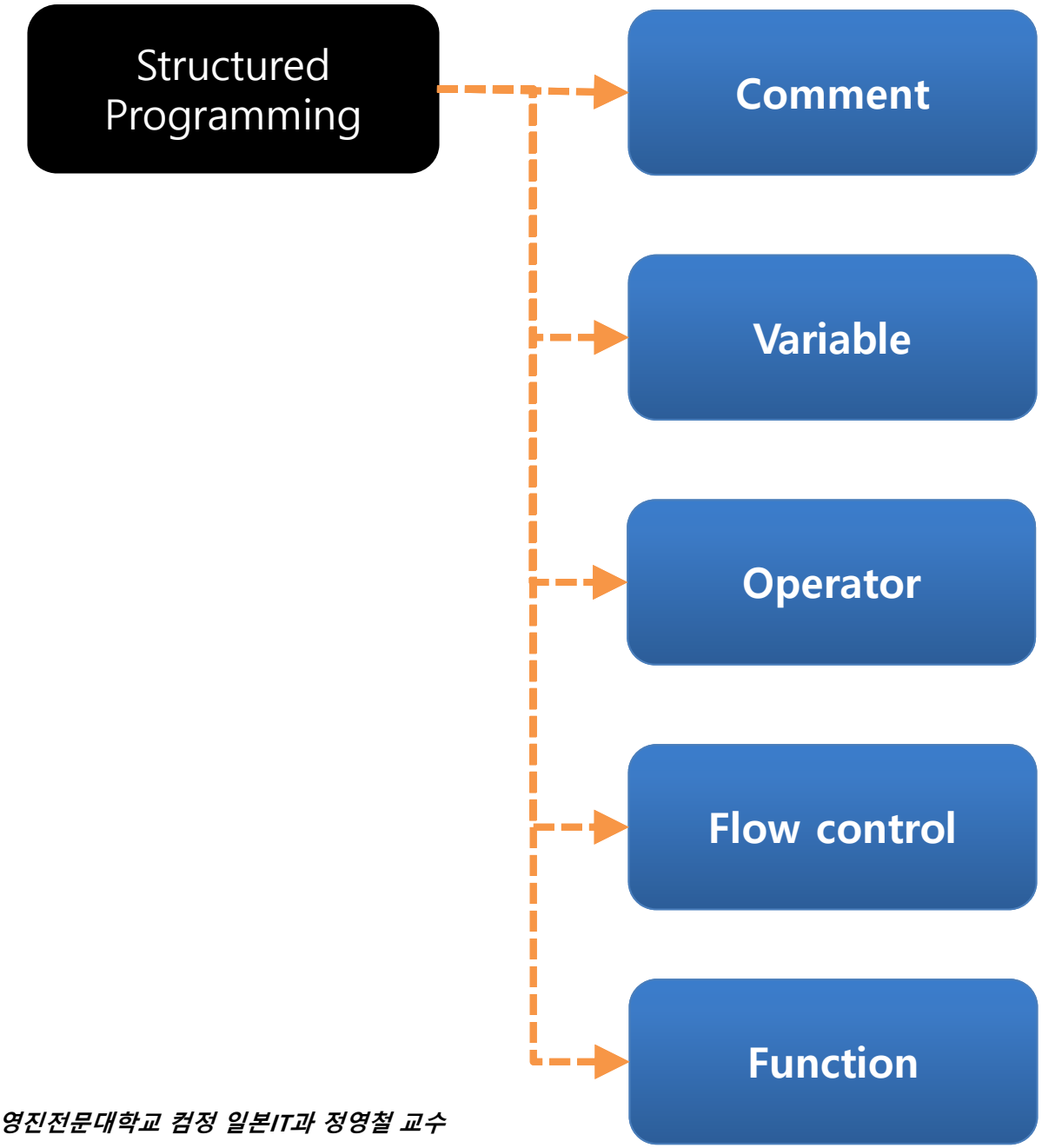
주문식교육의 산실
영진전문대학교

구조적 프로그래밍 구성 요소

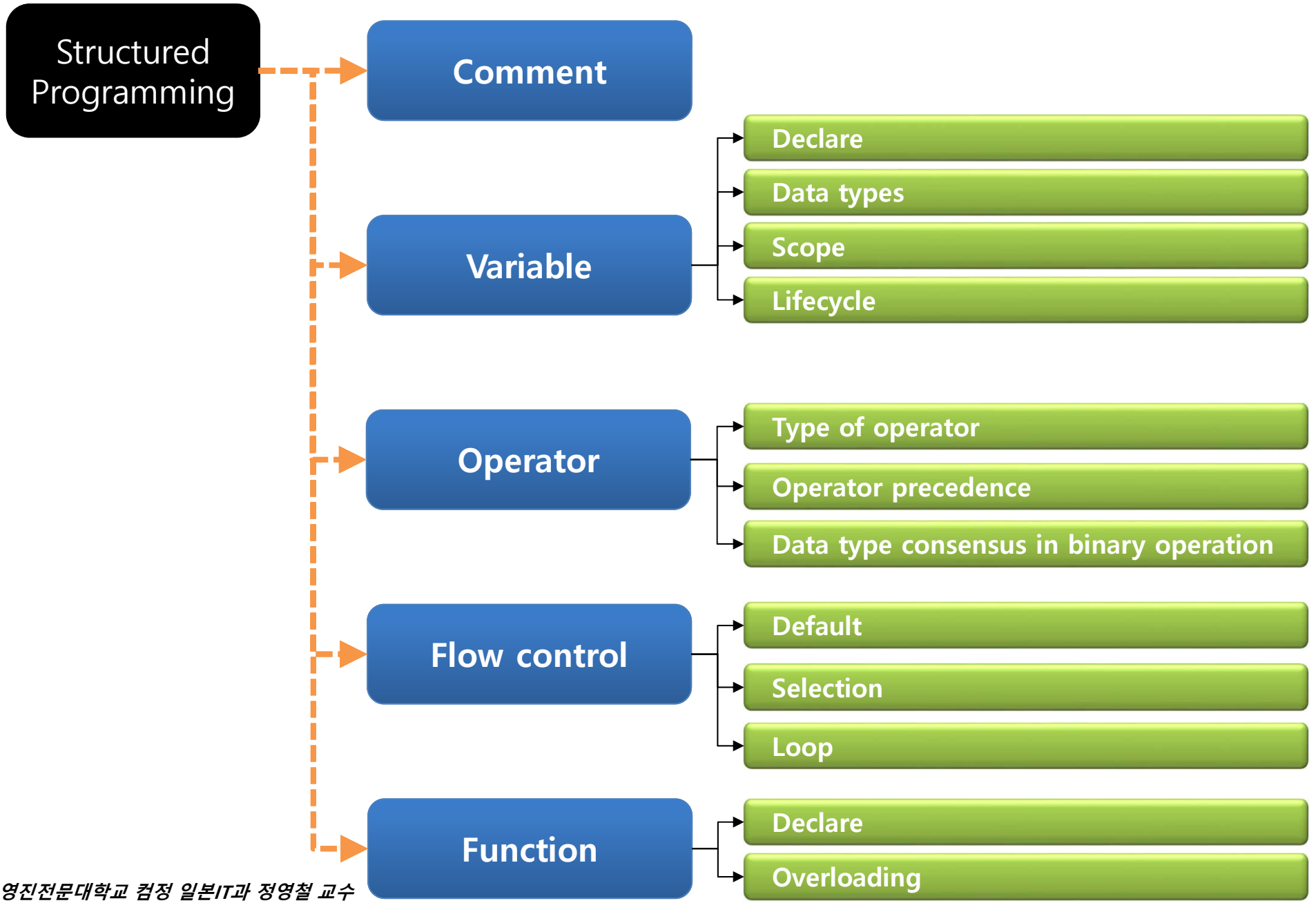
일본IT과

정영철 교수

구조적 프로그래밍 구성 요소 (1)



구조적 프로그래밍 구성 요소 (2)



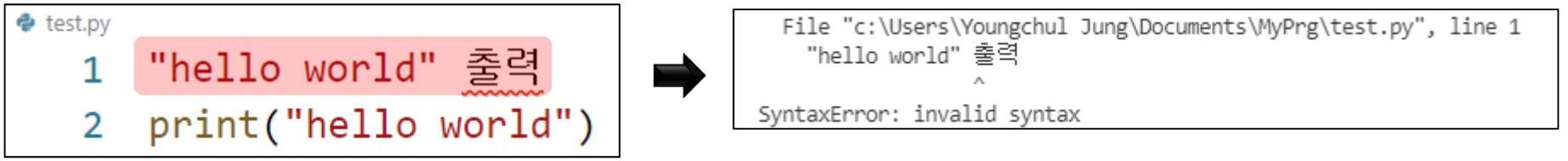
Comment (주석) : 주석이란?

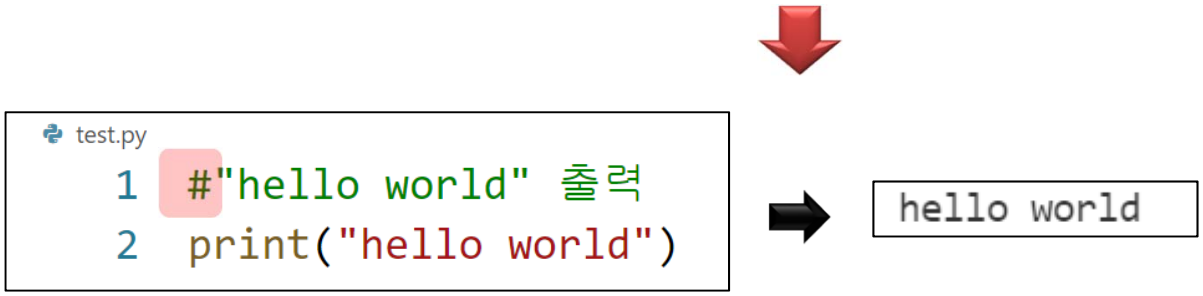
- 프로그램 소소코드의 설명을 적기 위해 사용
  - 즉 프로그래머를 위해 사용
  - 다른 사람들이 자신이 작성한 코드를 쉽게 이해 할 수 있도록, 명료하고 친절하게 작성하는 습관 필요

- 모든 컴퓨터 프로그래밍 언어는 주석 기능을 제공한다.

- 컴파일 또는 인터프리팅 시 코드와 주석을 구분하기 위한 구분자 필요

Interpreting





Comment (주석) : 파이썬 주석 사용 방법

- 두 가지 방법 제공
  - # 기호
    - # 이후 한 줄은 주석 처리
  - ''' ''' 기호
    - 여러 줄 주석 처리

test.py

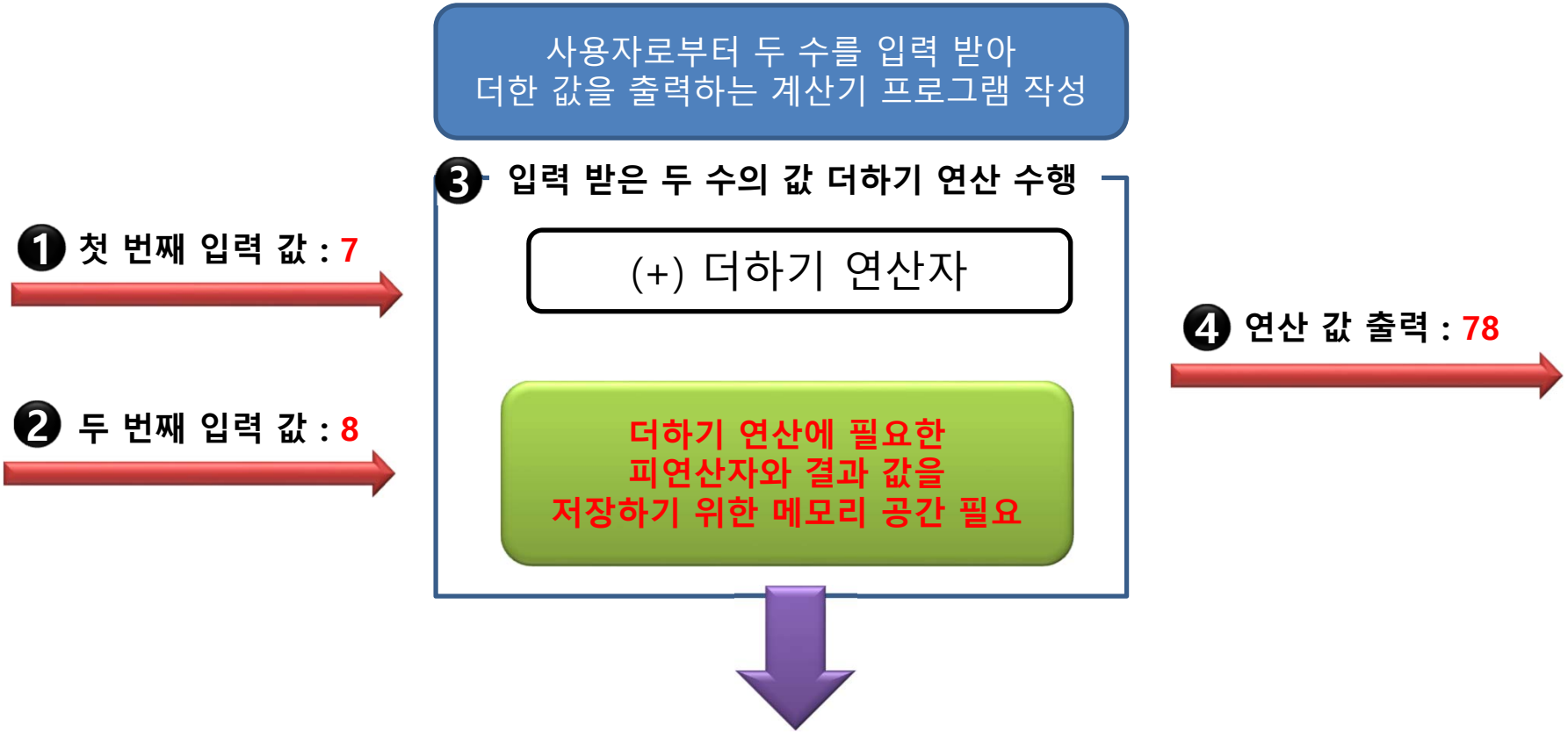
```
1  print("hello world") # "hello world" 출력
2
3  '''
4  파이썬에서 여러 줄
5  주석 처리 할 경우
6  위 Notation을 사용합니다.
7  '''
```

Result after interpreting

```
hello world
```

Variable : 변수란? (1)

- 프로그램에서 데이터를 메모리 상에 저장하기 위해 사용



변수 (Variable)를 사용하여 메모리 공간을 **선언**하고

선언 된 변수는 변수 이름을 이용하여 **사용 (GET, SET)**

Variable : 변수란? (2)

- 프로그램 작성 전 주석을 이용하여 전체 로직에 대한 검증 필요

test.py > ...

```
1  # 키보드로부터 첫 번째 입력 값 저장
2
3  # 키보드로부터 두 번째 입력 값 저장
4
5  # 입력 받은 두 개의 값에 대해 더하기 연산 실시
6
7  # 연산 결과 화면 출력
```

Variable : 변수란? (3)

- 두 수를 키보드로부터 입력 받아 더한 값 출력 프로그램
- 아래 프로그램에서 사용된 변수를 찾아 보고 변수의 사용 절차에 대해 고민

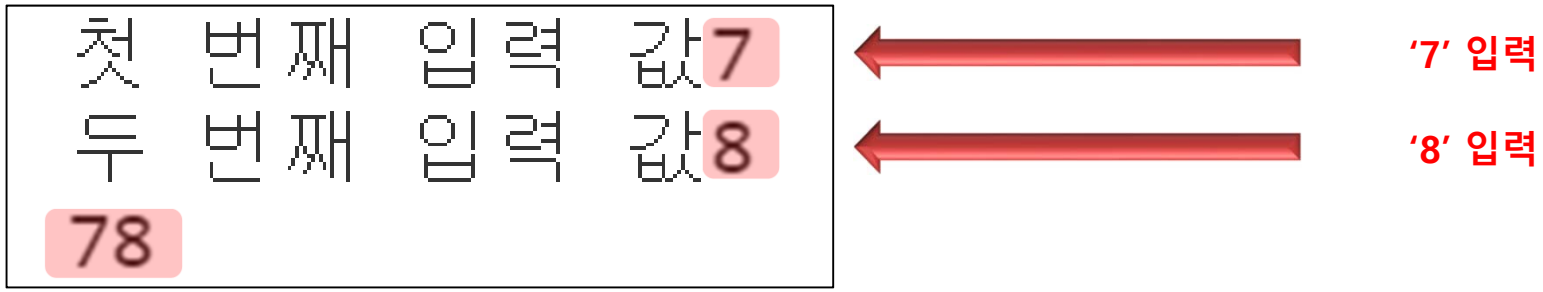
test.py > ...

```
1  # 키보드로부터 첫 번째 입력 값 저장
2  inpuValueA = input("첫 번째 입력 값")
3
4  # 키보드로부터 두 번째 입력 값 저장
5  inpuValueB = input("두 번째 입력 값")
6
7  # 입력 받은 두 개의 값에 대해 더하기 연산 실시
8  result = inpuValueA + inpuValueB
9
10 # 연산 결과 화면 출력
11 print(result)
```

Variable : 변수란? (4)

- 두 수를 키보드로부터 입력 받아 더한 값 출력 프로그램 실행 예제

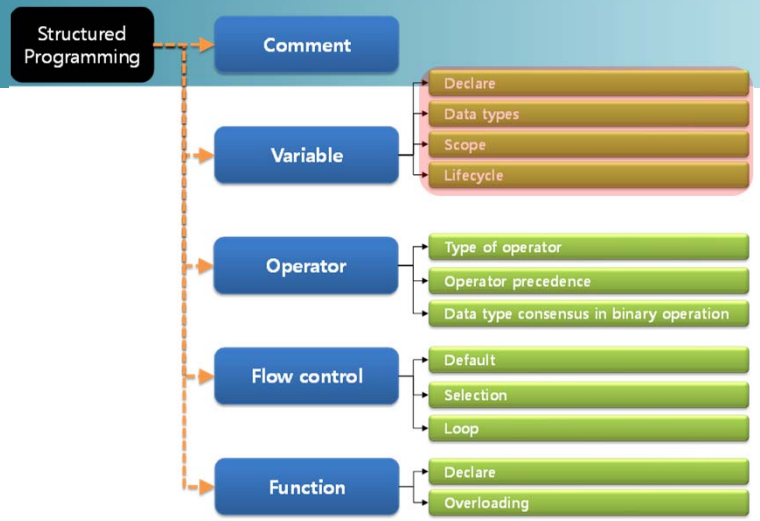
✓ **프로그램 실행 화면**



'7' + '8' 결과 값 78 출력

고찰!!
왜 15가 아니고 '78'일까?

변수 학습 시 고려 사항



- Declare (선언)
- Data types (자료형)
- Scope (범위)
- Lifecycle (생명주기)

변수 학습 시 고려 사항 : 선언 (1)

- Declare (선언)
  - 변수를 사용하기 위해서는 반드시 선언을 해야 된다.

- 변수 선언 시 반드시 필요한 것은?
  - 변수의 이름

- 이름이 왜 필요할까?
  - 변수를 왜 사용하는지 생각 해볼 것
    - 변수는 프로그램에 데이터를 저장하기 위한 공간을 마련 하는 것
    - 프로그램 내 다수개의 저장 공간이 필요할 것
    - 즉 여러 개의 변수가 선언된다는 것
    - 따라서 다수개의 변수 중 특정 변수를 접근하기 위해서는 각 변수마다 이름을 부여

- Compiler 언어들의 경우 자료형도 변수 선언 시 같이 사용

변수 학습 시 고려 사항 : 선언 (2)

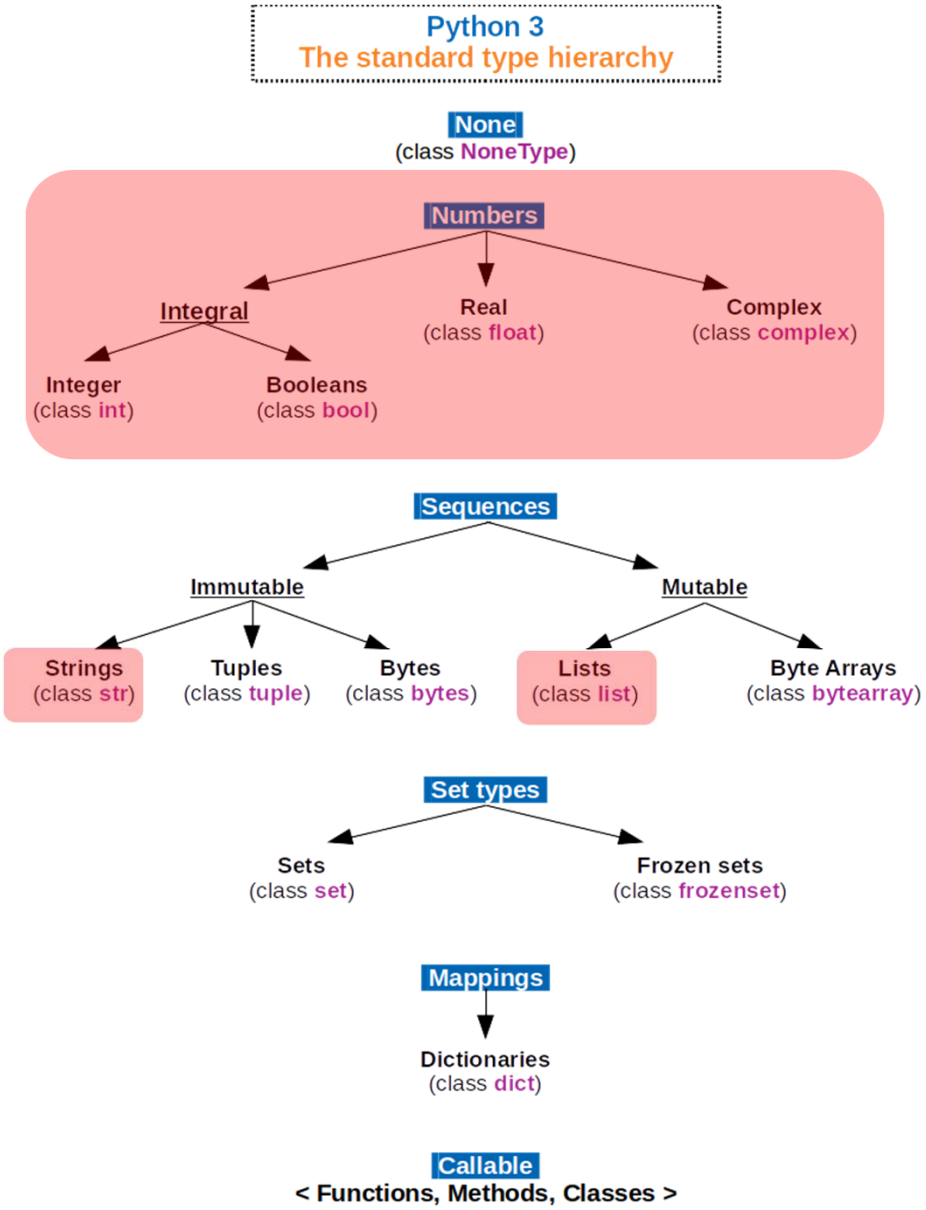
- Declare (선언)
  - 변수를 사용하기 위해서는 반드시 선언을 해야 된다.

- 변수 선언 시 반드시 필요한 것은?
  - 변수의 이름

- 이름이 왜 필요할까?
  - 변수를 왜 사용하는지 생각 해볼 것
    - 변수는 프로그램에 데이터를 저장하기 위한 공간을 마련 하는 것
    - 프로그램 내 다수개의 저장 공간이 필요할 것
    - 즉 여러 개의 변수가 선언된다는 것
    - 따라서 다수개의 변수 중 특정 변수를 접근하기 위해서는 각 변수마다 이름을 부여

- Compiler 언어들의 경우 자료형도 변수 선언 시 같이 명시
  - 예) Java 언어
    - int value; -> 정수형 변수 value 선언
    - float value; -> 실수형 변수 value 선언

변수 학습 시 고려 사항 : 파이썬에서 지원되는 변수 자료형 (1)



Text Type: `str`

Numeric Types: `int`, `float`, `complex`

Sequence Types: `list`, `tuple`, `range`

Mapping Type: `dict`

Set Types: `set`, `frozenset`

Boolean Type: `bool`

Binary Types: `bytes`, `bytearray`, `memoryview`

파이썬에서 지원되는 변수 자료형 : int, float, str

```
1  ################
2  # 정수   자료형 #
3  ################
4  temp_int = 2
5
6  print(type(temp_int))
7  # 출력 값 : <class 'int'>
8
9
10 ################
11 # 실수   자료형 #
12 ################
13 temp_float = 1.5
14
15 print(type(temp_float))
16 # 출력 값 : <class 'float'>
17
18
19 ################
20 # 문자   자료형 #
21 ################
22 temp_string_1 = "bar"
23 temp_string_2 = 'foo'
24
25 print(type(temp_string_1), type(temp_string_2))
26 # 출력 값 : <class 'str'>, <class 'str'>
```

파이썬 변수의 자료형은 입력되는 값에 따라 동적 변화
- Type juggling 이라고 명칭
- 즉 변수의 자료형은 변수의 입력 값에 의해 결정!!

type(변수) 함수
- 입력된 변수의 자료형을 문자열로 반환
- 파이썬에서 변수의 자료형은 입력 값에 따라 변화
- 현 변수의 자료형 확인 필요

파이썬에서 지원되는 변수 자료형 : bool (1)

- Boolean 값 : 논리 값, 참과 거짓으로 구성
- "True" or "False"로 표기, Case sensitive!!

```
40  ##################
41  # Bool  자료형 #
42  ##################
43  temp_bool_1 = True
44  temp_bool_2 = False
45
46  print(type(temp_bool_1), type(temp_bool_2))
47  # 출력 값 : <class 'bool'>, <class 'bool'>
48
49  if temp_bool_1 :
50      print("temp_bool_1 : 참")
51  else :
52      print("temp_bool_1 : 거짓")
53
54  if temp_bool_2 :
55      print("temp_bool_2 : 참")
56  else :
57      print("temp_bool_2 : 거짓")
```

중요!
if, else문 다음에 실행 문장은 반드시 한 칸 이상 공백 유지

미리 맛보기 코너 – 흐름제어 if else

- if else : 흐름제어 – 선택 문
- 사용방법

```
if 조건식 :
    #"참" 일 경우 실행될 문장
else:
    #"거짓" 일 경우 실행될 문장
```

파이썬에서 지원되는 변수 자료형 : bool (2)

- **Truthy : Falsy 값**
- Bool 자료형에서 참 또는 거짓으로 처리 되는 값
- Falsy 이외 값은 모두 Truthy로 처리

```
60  # Truthy, Falsy 예제
61  temp_1 = 1  # 정수형 변수
62  temp_2 = -1 # 정수형 변수
63  temp_3 = 0  # 정수형 변수
64  temp_4 = -0 # 정수형 변수
65
66  if temp_1:          # 1 -> Truthy 값
67      print("참")     # 출력값
68  else:
69      print("거짓")
70
71  if temp_2:          # -1 -> Truthy 값
72      print("참")     # 출력값
73  else:
74      print("거짓")
75
76  if temp_3:          # 0 -> Falsy 값
77      print("참")
78  else:
79      print("거짓")   # 출력값
80
81  if temp_4:          # -0 -> Falsy 값
82      print("참")
83  else:
84      print("거짓")   # 출력값
```

파이썬에서 지원되는 변수 자료형 : bool (3)

```python
# 자료형 별 Falsy 값
temp_1 = 0      # 정수형 변수
temp_2 = 0.0    # 실수형 변수
temp_3 = ""     # 문자열 변수
temp_4 = None   # 값이 없음을 의미.

if temp_1:          # 0 -> Falsy 값
    print("참")
else:
    print("거짓")   # 출력 값

if temp_2:          # 0.0 -> Falsy 값
    print("참")
else:
    print("거짓")   # 출력 값

if temp_3:          # "" -> Falsy 값
    print("참")
else:
    print("거짓")   # 출력 값

if temp_4:          # None -> Falsy 값
    print("참")
else:
    print("거짓")   # 출력 값
```

- 아래 값 Falsy 값으로 사용
  - Boolean : False
  - int : 0
  - float : 0.0
  - None
  - str : “” or ‘’
  - collection
    - [ ]
    - { }
    - ( )
    - range(0)

파이썬에서 지원되는 변수 자료형 : list (1)

List : 여러 개의 변수를 하나의 꾸러미로 관리
- [ ] 연산자를 이용하여 리스트 선언
- 각 원소를 접근하기 위해서는 index 번호 이용
- 원소는 0부터 시작 총 개수 n이면 제일 끝 index는 n-1

```
49  ##################
50  # List  자료형 #
51  ##################
52
53  temp_list = [10, 2, 2.5, "test"]      # 4개의 원소를 가지는 리스트 선언
54
55  print(temp_list[0], temp_list[1], temp_list[2], temp_list[3])
56  #10 2 2.5 test
57                                        # 각 원소를 접근하기 위해 [ index ] 표기법 사용
58  temp_list[3] = 9                        - 첫 번째  원소는 0부터 시작
59  temp_list[2] = 8                        - 마지막 원소는 n – 1, n : 전체 원소 갯수
60  temp_list[1] = 7
61  temp_list[0] = 6
62
63  print(temp_list[0], temp_list[1], temp_list[2], temp_list[3])
64  # 6 7 8 9
65
66  temp_list[4] = 20 # error     # 에러가 발생한 이유는?
```

실습 (1)

- 지구에서 프록시마 케타우리 별까지 빛의 속도록 간다면 몇 년이 걸릴까?
  - 지구-> 프록시마 케타우리 별까지 거리 : 40 X 10^{12} km
  - 빛의 속도 : 300,000 km/sec

실습 (2)

- 아래와 같이 동작하는 프로그램을 작성하라.
  - 키보드로부터 문자 입력
    - 입력 값이 “남자” 이면 “MAN” 출력
    - 입력 값이 “여자” 이면 “WOMAN” 출력

실습 (3)

- 아래와 같이 동작하는 프로그램을 작성하라.
  - 키보드로부터 문자 입력
    - 입력 값이 < 0 이면 "음수" 출력
    - 입력 값이 > 0 이면 "양수" 출력

변수 학습 시 고려 사항 : 변수의 범위 & 생명주기 (1)

```
1  # 변수의 범위(Scope)
2  #  -> 변수를 접근 할 수 있는 영역
3  #  -> 변수의 생명주기와 연관
4
5
6  print(msg) # 에러
7  # Interpreting시 msg 변수를 찾지만
8  # msg 변수가 6번 라인 이전에 선언 되어 있지 않음
9  # 이에 msg 변수를 찾지 못해 프로그램 번역 중단
10
11 # msg 변수를 선언
12 msg = "hello"
```

```
1  # msg 변수를 선언
2  msg = "hello"
3
4  print(msg) # 문자열 hello 출력
```

변수는 선언 된 이후부터 사용이 가능하며
접근 범위는 선언 이후부터 소스 파일의 끝까지

변수 학습 시 고려 사항 : 변수의 범위 & 생명주기 (2)

```
2  # 두 개의 값을 입력 받아 더한 값을 출력하는 함수 정의
3  def Add(argValueA, argValueB):
4      # 입력 받은 두 개의 수를 더한 후 문자열로 변환
5      result = str( (argValueA + argValueB) )
6      # 출력 메시지 작성
7      msg = "합계 : " + result
8      # 함수 반환 값
9      return msg
10
11
12 # 3번 라인에서 작성한 함수 호출
13 print(Add(2,3)) # 합계 : 5
14
15 # 3번 라인에서 작성한 함수 호출
16 print(Add(4,5)) # 합계 : 9
```

중요!
함수 내 뛰어 쓰기로 함수 내 실행문장 블록화 처리, 따라서
특정 함수에 포함되는 실행 문은 반드시 동일한 띄어쓰기 적용

미리 맛보기 코너 – 함수 (Function)

- **함수란?**
  - 자주 사용되는 알고리즘을 함수라는 단위로 작성하고, 이를 프로그램 내에서 필요 시 호출하여 프로그램 작성 및 관리에 효율성을 증진하기 위해 사용
- **사용방법**

```
def 함수명 (매개 변수) :
    # 함수 호출 시 실행 문장
    # 함수 호출 시 실행 문장
    # return 값 <- Option
```

변수 학습 시 고려 사항 : 변수의 범위 & 생명주기 (3)

```
1  name = "YC Jung"
2
3  def printHelloMsg():
4      print("Hello")
5      print(name) # 함수 내에서 함수 밖에 선언되어 있는 변수(전역변수) 접근 가능
6      # name = "Richard Jung" -> 주석 해제 전/후를 실행하고 결과 값 분석!
7      position = "professor" # position 변수의 생명주기?
8
9
10 printHelloMsg()
11
12 print(name)
13
14 print(position)
```

- 변수의 접근 범위는 변수의 생명 주기와 연관 되어 있다.
- 생명주기란 특정 객체가 태어나서 죽을 때 까지의 기간을 의미
- 따라서 변수의 생명주기는 변수가 메모리에 할당된 후 해제 될 때 까지를 의미
- 함수 내 선언된 변수의 생명 주기
  - Birth(출생) : 함수가 호출될 때
  - Death(죽음) : 함수 호출 후 호출된 함수가 종료 될 때
- 이에 함수 내 선언 된 변수들은 함수 내에서만 사용 가능하고, 외부에서는 사용 될 수 없다
- 함수 내 선언 된 변수들을 지역변수(Local variable)라 한다.
- 함수 밖 선언된 변수들을 전역변수 (Global variable)라 하며, 함수 내에서 사용가능

변수 학습 시 고려 사항 : 변수의 범위 & 생명주기 (4)

```
1  value = 10
2
3  def multiply10():
4      # global value  주석 처리 전/후 결과 값 비교
5
6      value = value * 10
7
8      return value
9
10 print(multiply10())
```

```
1  value = 10
2
3  def multiply10():
4      global value  #주석 처리 전/후 결과 값 비교
5
6      value = value * 10
7
8      return value
9
10 print(multiply10())
```

- 함수 내에서 전역 변수를 접근하기 위해서 Global 키워드를 사용한다
- 변수의 GET 동작 모드 시 Global 키워드 생략 가능
- SET 모드의 경우 반드시 필요

실습 (5)

- 화면에 “hello”를 출력하는 printHello 함수를 작성하라

학생 작성 부분!
printHello 함수 작성
함수 호출 시 화면에 “hello”출력

```
2
3
4
5  printHello() # hello 출력
```

실습 (5)

- 세개의 정수 값을 입력 받아 곱한 값을 반환하는 함수를 작성하라.

```
2  def multiply(argA, argB, argC):
3          학생 작성 부분!
4
5  value = multiply(1, 2, 3)
6
7  print(value) # 출력값 6
```

실습 (4)

- 특정 정수의 값이 짝수 이면 “짝수 " 아니면 “홀수”를 출력하는 함수를 작성하라.

```
2  # 입력 값의 홀, 짝 여부 판별 후
3  # 출력 : "짝수" or "홀수"
4  def getEvenOdd(argValue):
5
6
7
8
9
10
11 getEvenOdd(2) # 결과값 "짝수"
12
13 getEvenOdd(3) # 결과값 "홀수"
```

학생 작성 부분!

Q/A

감사합니다

주문식교육의 산실

영진전문대학교