aws re/start

Python 기본 사항

Python 기본 사항

발표자 이름

날짜

학습 내용

강의 핵심 내용

학습 내용:

- Linux 컴퓨터에 Python을 설치합니다.
- 기본적인 Python 용어를 정의합니다.
- Python 스크립트에서 변수를 선언하고 해당 변수에 작업을 수행합니다.
- 스테이트먼트, 함수, 예외를 설명합니다.

Python을 사용하기 위한 시스템 요구 사항

Microsoft Windows

macOS

Linux

데모: Python 설치

현재 설치된 Python을 확인하고 새로운 버전을 설치합니다.

aws re/start

Python 구문 기본 사항

Python은 들여쓰기와 공백을 사용하여 코드 블록을 그룹화합니다.

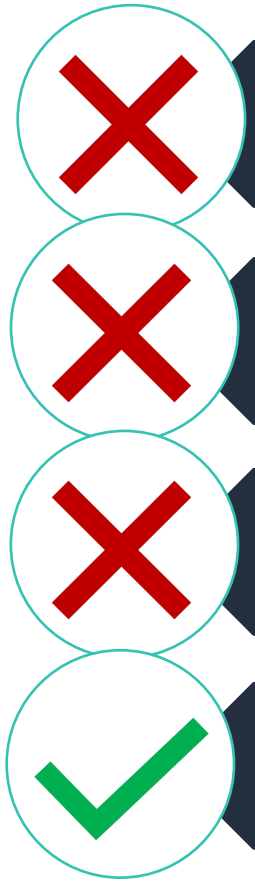
- 런타임 오류가 발생하면 먼저 공백을 확인하십시오.
- 다음으로, 콜론 등 구두점이 누락되지 않았는지 들여쓰기를 확인하십시오.

Python은 대소문자를 구분하므로 대소문자가 중요합니다.

식별자

Python에서 **식별자**는 클래스, 함수, 변수와 같은 개체의 이름입니다. 개체를 구별하는 데 도움이 됩니다.

Python에서 객체의 이름을 지정할 때 다음과 같이 지켜야 할 규칙이 있습니다.



식별자는 숫자로 시작해서는 안 됩니다. 1variable은 유효하지 않지만 variable1은 유효합니다.

키워드는 식별자로 사용할 수 없습니다.

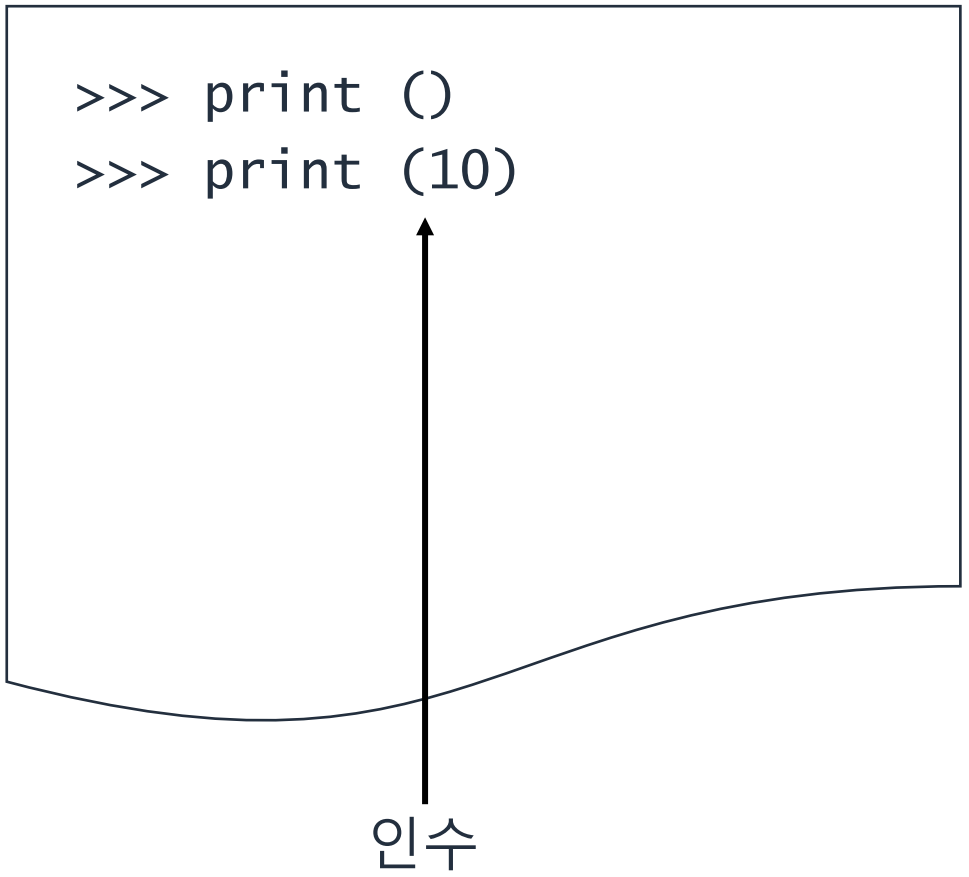
식별자에 !, @, #, $, %와 같은 특수 기호를 사용해서는 안 됩니다.

식별자의 길이는 상관이 없습니다.

파일 확장명: .py 파일

.py라는 확장명은 Python에서 사용하는 스크립트 파일 확장명입니다.
확장명이 .py인 파일은 어느 텍스트 편집기에서나 만들 수 있지만, 실행하려면 Python 인터프리터가 필요합니다.

함수



- 함수는 컴퓨터가 특정 태스크를 수행하도록 지시합니다.
- 함수에는 **이름**이 있으며, 이름 다음에 괄호를 추가하여 호출합니다.
- 많은 함수가 **인수**를 취합니다. 인수는 함수가 특정 태스크를 수행하는 데 사용되는 정보입니다.
- **print** 함수는 유용한 함수입니다.
- 함수를 직접 작성할 수 있습니다.
- Python에는 이미 내장된 기본 제공 함수(예: **print**)가 있습니다.

Vim 검토

여기에서 사용할 텍스트 편집기는 Vim(Vi improved)입니다.

- vim <file name> 또는 vim을 입력하여 이를 명령줄에서 엽니다.
- 일반 모드에서 Vim이 열립니다. Vim에 작성하려면 i를 입력합니다(i는 삽입을 의미).
- 일반 모드로 돌아오려면 ESC 키를 누릅니다.
- 일반 모드에서 편집을 저장하거나 중단하거나 계속할 수 있습니다.
- 파일을 저장하려면 :w <file name> 또는 :w를 입력합니다(w는 쓰기를 의미).
- Vim을 닫으려면 :q를 입력합니다. q는 quit, 즉 종료를 의미합니다.
- 저장 후 닫으려면 :wq를 입력합니다.

데모: Hello World

1. CentOS에서 **helloworld.py**라는 파일을 만듭니다.
2. 파일을 열고 파일에 씁니다.
   print("Hello World")
3. 파일을 저장하고 명령줄로 돌아옵니다.
4. Python으로 파일을 실행하고 결과를 관찰합니다.

코멘트

```
# 기호는 코멘트를 나타냅니다.
```

- 코멘트는 본인 또는 다른 개발자를 위한 메모입니다.
- 코멘트를 통해 프로그램의 개념과 작동 방식을 안내하여 소스 코드를 검토하는 사람의 이해를 돕습니다.
- # 기호를 사용하여 코멘트 작성을 시작합니다.
- # 기호로 시작하는 텍스트 줄을 시작하면 인터프리터가 이 줄을 코드로 실행하지 **않도록** 지시하는 것입니다.

데모: Hello World와 코멘트 작성

1. helloworld.py 파일을 엽니다.
2. i를 입력하여 더 많은 텍스트를 삽입합니다.
3. 다른 줄에 코멘트를 입력합니다.
   - 예: `# 나는 Python이 좋아요`
4. 작성한 후 종료합니다.
5. 다음 명령을 사용하여 파일을 다시 실행합니다.
   `python3.7 helloworld.py`
6. 변화가 있습니까? 그 이유는 무엇입니까?

aws re/start

데이터 유형

데이터 유형은 객체가 어떤 작업을 수행하는지 또는 논리적으로 말이 되는지 판단합니다.

Python은 객체와 함께 객체의 유형을 저장합니다. 작업이 수행되면 해당 작업이 해당 객체에 논리적으로 적절한지 확인합니다(이 기법을 **동적 타이핑**이라고 함).

예: Python의 일부 함수와 메서드는 특정 데이터 유형을 요구합니다. 부동 소수점 수를 요구하는 함수에 문자열을 전달할 수 없습니다.

기본 데이터 유형

Integer(int):

정수. 음수일 수 있음

예:
4
3000
-29

Float:

소수를 포함하는 수. 음수일 수도 있음

예:
3.390
8.090
-0.001

Boolean(bool):

True 또는 False 조건

대소문자가 중요합니다.

String(str):

따옴표(" " 또는 ' ' 또는 """ """)로 구분되는 문자 세트. 숫자, 글자, 특수 문자일 수 있음

예:
"I am a string"
'23456'
"'I have 56 sheep'"

참고: 문자열에 있는 숫자는 문자열로 취급합니다. Python은 이 숫자를 정수 또는 부동 소수점 수로 간주하지 않습니다.

데이터 유형: 변경 가능/변경 불가능

Mutable은 **변경할 수 있다**는 뜻입니다. Python에는 **변경 가능**한 데이터 유형과 **변경 불가능한** 데이터 유형이 있습니다.

변경 가능

- 목록
- 세트
- 사전
- 바이트 어레이

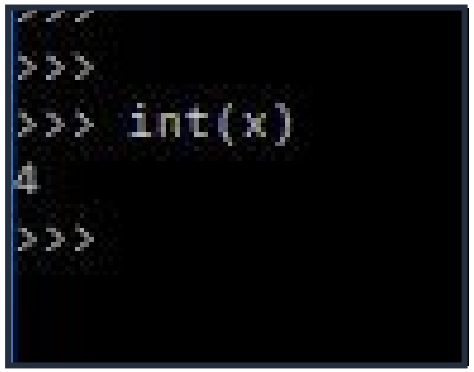
변경 불가능

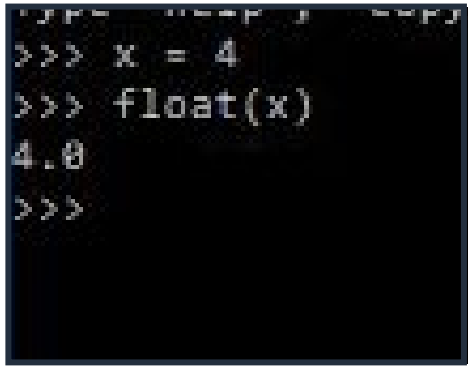
- 정수, 부동 소수점 수, 복합
- 문자열
- 튜플
- 동결 세트

데이터 유형 변환

Python에는 객체의 데이터 유형을 변경하는 데 사용되는 명령이 있습니다.

- **float()**: 예에서는 4가 *x* 변수에 할당되어 있습니다. 기본값으로 4는 정수이지만, **float()** 명령을 사용하기 위해서는 x가 부동 소수점 수로 식별되어야 합니다. 결과는 4.0입니다.
- **int()**: *x*를 **int(x)** 명령에 다시 전달하면 다시 4가 됩니다. 이 명령은 소수점 이하를 자릅니다.
- 참고: **int()**는 반올림/반내림과는 다릅니다. 숫자에 있는 소수점 이하의 수를 제거하기만 합니다. 소수를 반올림하거나 반내림하지 않으니 유의하시기 바랍니다.





목록, 세트, 튜플과 기타 데이터 유형은 나중에 다룹니다.

문자열

```
>>> "I'm a string!"
>>> 'Me too'
>>> '''Me too!'''
>>> "I have written 3 strings"
```

- 문자열에는 **텍스트**가 포함됩니다. 텍스트는 하나의 문자일 수도 있고, 텍스트 단락일 수도 있습니다.
- 문자열은 양쪽 끝에 따옴표가 있는 문자입니다.
- 문자열에는 숫자를 포함할 수 있습니다.
- 문자열을 나타내는 방법은 다음의 세 가지입니다.
  - 홑따옴표(' ')
  - 겹따옴표(" ")
  - 세겹따옴표(''' ''')

문자열 연결

```
>>> "I'm a string!"
>>> 'Me too'
>>> "I'm a string" + "Me too"
>>> "I'm a stringMe too"
```

- 문자열은 **변경 불가능**합니다. 문자열을 조작하면 새로운 문자열이 만들어집니다.
- 문자열을 **연결**, 즉 함께 **덧붙일** 수 있으며 연결하면 새로운 문자열이 만들어집니다.
- 빈 문자열을 만들 수도 있습니다.
  - 예: **x = ""**

참고: 새 문자열의 단어 사이에 공백을 넣으려면 원래 문자열 중 하나에 추가해야 합니다. 직접 추가하지 않으면 Python은 공백을 삽입하지 않고 있는 문자열을 그대로 추가합니다.

간단한 Python 프로그램

```
>>> 1 + 1
>>> 2
```

- 이 간단한 Python 프로그램은 숫자 1과 1을 더합니다.
- 결과는 다음 줄에 표시된 것처럼 2입니다.
- 화살괄호(>>>)는 프로그램에 속하지 않습니다. Python 인터프리터가 표시하는 것입니다.

변수

```
>>> x = 1
>>> x + x
2
```

- 이 예에서 x는 숫자 1이 포함된 변수입니다.
- 변수로 작업할 때 Python은 값을 조회하여 어떤 작업을 수행해야 할지 파악합니다.

변수(계속)

```
>>> apples = 2
>>> oranges = 3
>>> apples = oranges
>>> car_color = red
```

- 원하는 만큼 변수를 만들 수 있습니다.
- 몇 가지 제약을 지키면 변수의 이름도 마음대로 정할 수 있습니다.
- 제약:
  - 변수 이름에는 글자, 숫자, 밑줄(_)만 포함해야 합니다.
  - 변수 이름은 숫자로 시작해서는 안 됩니다.
  - 변수 이름은 Python의 키워드여서는 안 됩니다.

데모: 변수

지침

1. variables.py라는 새 파일을 만듭니다.
2. 원하는 대로 변수를 몇 개 할당합니다. 문자열, 부동 소수점 수, 정수를 사용합니다.
3. 예를 들면 다음과 같습니다.

```
x = 33
y = 7.0
variable_challenge = "I love Python"
```

4. **Print()** 함수를 사용하여 Python이 변수를 출력하도록 지시합니다. 지시하지 않으면 파일을 호출할 때 결과를 볼 수 없습니다.

   예:

```
print(x)
print(y)
print(variable_challenge)
```

5. 쓰고 파일을 종료합니다. 명령줄에서 파일을 호출합니다. 어떤 일이 발생했습니까?

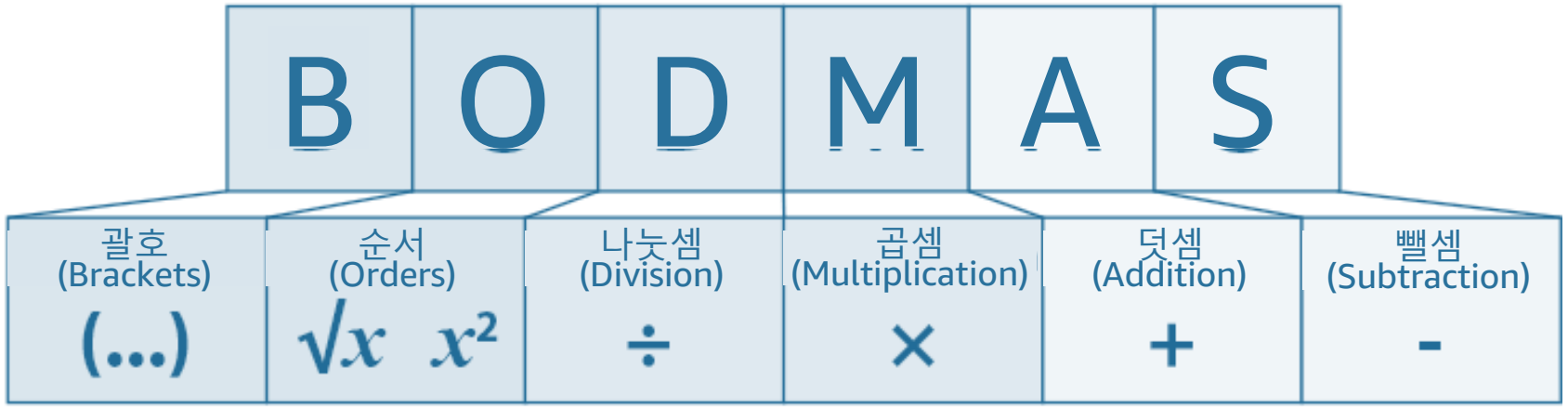
연산자

산술 연산자	(+ - * / % ** //)
비교(관계) 연산자	(<, >, <=, >=, !=)
할당 연산자	(=, +=, -=, *=, /=, %=)
논리 연산자	(and, or, not)
멤버 연산자	(in, not in)
아이덴티티 연산자	(is, is not)

BODMAS

BODMAS는 작업 순서를 설명합니다.

- 괄호(Bracket)
- 순서(Orders, 지수화)
- 나눗셈(Division)
- 곱셈(Multiplication)
- 덧셈(Addition)
- 뺄셈(Subtraction)



데모: 기본적인 계산

지침

1. basicmath.py라는 새 파일을 만듭니다.
2. 파일에 텍스트를 삽입하고, 기본적인 계산 함수를 작성해 봅니다.
   예:
   1+1
   100/33
   2**2
3. 이제 **print()** 함수를 사용하여 Python에 이 표현식을 출력하도록 지시합니다.
4. 쓰고 파일을 종료합니다.
5. 실행합니다. 어떤 일이 발생했습니까?

데모: 변수와 기본적인 계산

지침

변수 할당을 계산과 결합하면 어떤 일이 발생합니까?

1. basicmath.py 파일을 엽니다.
2. 기본적인 계산 함수에 변수 이름을 할당하도록 파일을 편집합니다.
   예:
   ```
   easy_math = 1+1
   one_third = 100/33
   powerful = 2**2
   ```
3. Python이 변수 이름을 출력하도록 지시합니다.
4. 변수 이름만으로도 계산할 수 있습니다.
   예:
   ```
   x = 2
   y = 4
   z = x+y
   print(x + y)
   print (z)
   ```
5. z와 x+y를 출력했을 때 동일한 결과가 나올 것으로 예상됩니까? 그 이유는 무엇입니까?

데모: 문자열과 연결

지침

1. funwithstrings.py라는 새 파일을 만듭니다.
2. 변수를 만들고 변수에 문자열을 할당합니다.
   예:
   ```
   first_str = "I love dogs"
   ```
3. 두 번째 변수를 만들고 변수에 다른 문자열을 할당합니다.
   예:
   ```
   second_str = " and cats"
   ```
4. newstr이라는 세 번째 변수를 만들고 여기에 문자열 1과 문자열 2의 연결을 할당합니다.
   예:
   ```
   newstr = first_str + second_str
   ```
5. newstr을 출력합니다. 어떤 일이 발생했습니까?

aws re/start

스테이트먼트

```
>>> apples = 2
>>> oranges = 3
>>> apples = oranges
```

- **스테이트먼트**는 일반적으로 코드 줄이며, 지금까지 본 각 코드 줄이 하나의 스테이트먼트입니다.
- 스테이트먼트는 Python에 전달하는 개별적인 지침입니다.
- 이 코드에는 세 개의 스테이트먼트가 있습니다.
- apples를 oranges에 할당하면 apples의 값이 바뀝니다. `print(apples)`를 출력하면 어떤 응답이 나옵니까?

함수, 문자열, 변수

```
>>> name = "Kwesi"
>>> print (name)
```

- 변수에 문자열을 저장합니다.

예외

- 예외는 오류를 보고합니다.
- 예를 들면 다음과 같습니다.
  - 변수를 잘못된 이름으로 참조합니다.
  - 숫자를 0으로 나눕니다.
  - 컴퓨터에 없는 파일을 읽으려고 시도합니다.
- 예외는 **스택 트레이스**를 유발하는데, 이는 잘못된 점이 있는 다양한 방식의 리스팅입니다.

데모: 예외 찾기

지침

- 온라인에서 Python 예외를 조사합니다. 가능한 한 많은 예외 유형을 찾습니다. 학급 전체와 함께 논의할 준비를 합니다.

aws re/start

학습 내용 확인

- Python에서 변수를 선언할 때 사용할 수 있는 문자에 제약이 있습니까?
- Python에서 함수를 어떻게 호출합니까?
- 1.7 값을 저장할 때 정수와 부동 소수점 수 중 어떤 데이터 유형을 사용합니까?
- **+=** 연산자는 어디에 사용합니까?

요점

- Python은 Microsoft Windows, macOS, Linux 컴퓨터에 설치할 수 있습니다.
- Python 파일의 확장명은 .py입니다.
- Python은 들여쓰기와 공백을 사용하며 대소문자를 구분합니다.
- 함수는 이름으로 호출할 수 있는 명령의 모음입니다.
- 데이터 유형은 변수에 포함된 데이터의 유형(정수, 부동 소수점 수 등)을 식별합니다.
- Python의 연산자는 계산, 등가 시험, 문자열 작업에 사용됩니다.
- 예외는 코드에 오류가 발생할 때 Python 인터프리터가 보고하는 오류입니다.

aws re/start