주석으로 코드를 설명하기

주석(Comment)

- **코드에 부가 정보를 추가하는 목적으로 사용 됨**
- **파이썬 인터프리터에 의해 무시되어 실행 되지 않음**
- **잘 사용하면 코드의 가독성을 높일 수 있음**
  - 과도한 주석은 오히려 가독성을 저하 시킬 수 있음

주석(Comment)의 문법

- **단일 행 주석**
  - 1라인만 주석처리
  - # 기호 사용
- **다중 행 주석**
  - 2라인 이상을 주석 처리
  - 따옴표(‘ 또는 “)기호 사용
- **단일 행 주석 예시**

```
# Hello World!를 출력하는 코드입니다.

print("Hello World!")
```

- **다중 행 주석 예시**

```
'''
문자열을 출력하는 코드입니다.
Hello World!를 출력합니다.
'''
print("Hello World!")
```

주석(Comment)의 실습

```
'''
주석을 실습해봅니다.
주석은 파이썬 인터프리터에 의해 무시됩니다.
주석은 코드에 부가 정보를 제공합니다.
'''
print("line 1")
# 이 부분은 실행 되지 않습니다.
print("line 2") # 주석은 코드 오른 편에 위치할 수도 있습니다.
# print("파이썬 코드도 주석으로 만들 수 있고, 실행 되지 않습니다.")
print("line 3")
```

파이썬 연산자

파이썬 연산자(Operator)

- **프로그래밍 언어에서 연산자란 특정 작업을 수행하는 기호**
  - '+' 연산자는 더하는 작업을 수행
  - '*' 연산자는 곱하는 작업을 수행

- **연산자는 피연산자가 필요**
  - 3 + 9 에서 '+'는 연산자, '3'과 '9'는 피연산자

- **연산자에 의한 연산은 결과가 존재**
  - 3 + 9 연산의 결과는 숫자 12

파이썬 연산자(Operator)의 종류

연산자 그룹	설명	연산자 예
산술 연산자	숫자 값을 대상으로 수학적 연산을 하기 위한 연산자	+, -, *, /, %, **
할당 연산자	값을 특정한 변수에 대입하기 위한 연산자	=, +=, -=, *=, /=
비교 연산자	두 개의 값을 비교 하기 위한 연산자	==, !=, >, <, >=, <=
논리 연산자	다양한 조건들이 결합된 논리식의 결과를 판단하기 위한 연산자	and, or, not
식별 연산자	주어진 데이터가	is, is not
멤버 연산자	특정 값이, 값의 그룹에 포함되어 있는지 판별하기 위한 연산자	in, not in
비트 연산자	비트 단위의 연산을 하기 위한 연산자	&, |, ^, ~

파이썬 연산자(Operator)의 우선순위

- 3 + 9 * 2 = ?
  - 24
  - 21 ✓

- 연산자 우선 순위

**
+x, -x,
*, /, %, //
+, -
in, not in, is, is not, <, <=, >, >=, <>, !=, ==
is, is not
not x
and
or

파이썬 연산자(Operator)와 괄호의 사용

- ()를 사용하면 연산자 우선순위와 상관 없이 연산 수행

- (3 + 9) * 2 = ?
  - 24 
  - 21

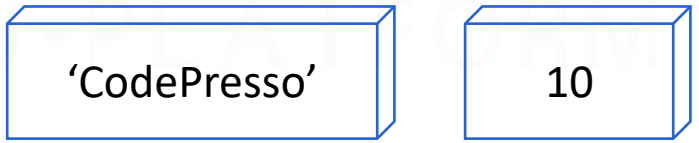
- 괄호를 사용하여 연산의 의도를 명확히 표현하는 것이 좋음

변수(Variable)

파이썬 변수(Variable)

- 컴퓨터 프로그램은 데이터와 그 데이터를 처리하는 알고리즘으로 구성 됨

- 변수(Variable)는 데이터를 저장하기 위한 공간



  - 실제 데이터는 컴퓨터 메모리에 저장 됨

- 변수는 이름이 존재

  - name, age, title, content, …
  - 명확한 이름은 그 변수에 어떤 데이터가 들어 있는지 쉽게 파악할 수 있게 함

변수의 생성과 값의 할당(저장)

- **파이썬에서 변수는 값을 할당(저장)하는 동시에 생성 됨**

- **할당 연산자 '='를 사용하여 변수에 값을 할당**
  - 변수 이름 = 변수에 저장할 값

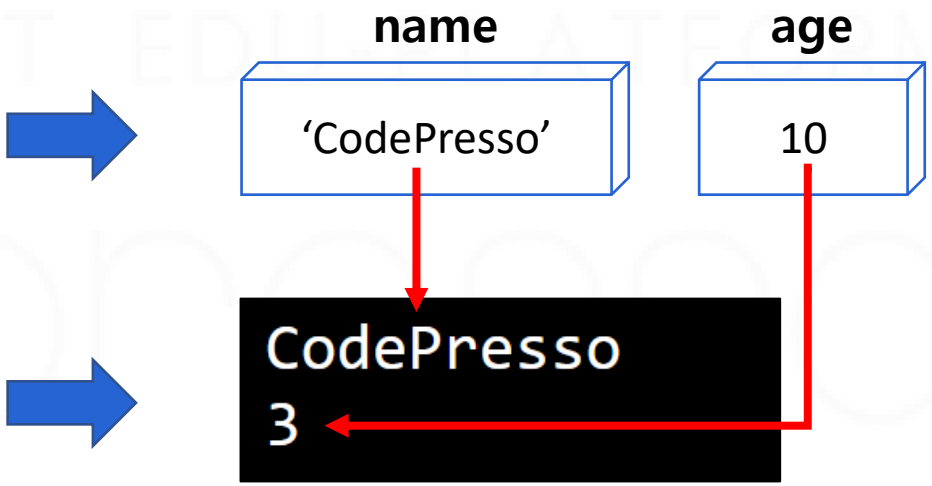
```
name = "CodePresso"
age = 3
```

변수 값의 사용

- 변수의 이름으로 변수에 저장 되어 있는 값을 사용

```
name = "CodePresso"
age = 3
```

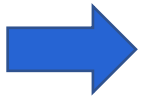
```
print(name)
print(age)
```

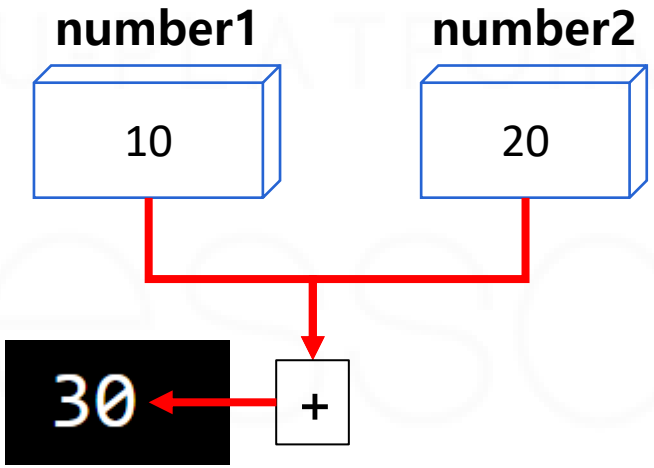


변수 값의 사용

- 변수의 이름으로 변수에 저장 되어 있는 값을 사용

```
number1 = 10
number2 = 20
```






```
print(number1 + number2)
```

파이썬 자료형(Data Type)

자료형(Data Type)

- 프로그램은 다양한 형태의 데이터가 사용 됨
  - 이름(문자), 나이(정수), 키(실수), 몸무게(실수), ...
- 자료형(Data Type)이란 문자형, 정수형, 실수형 등의 데이터의 형태
- 프로그래밍 언어는 다양한 종류의 데이터를 저장할 수 있게 지원
  - 프로그래밍 언어마다 상이

파이썬 자료형(Data Type)

- 파이썬에서 자주 활용되는 자료형(Data Type)의 종류

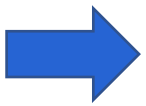
자료형(Data Type) 이름	설명
int	정수 자료형
float	실수 자료형
str	문자 자료형
bool	True, False를 나타내는 Bool 자료형
list	연속 된 데이터의 집합을 나타내는 리스트 자료형
tuple	연속 된 데이터의 집합을 나타내는 튜플 자료형
dict	Key, Value 형태의 데이터 집합을 나타내는 딕셔너리 자료형

자료형(Data Type)의 확인

- type()을 사용하여 특정 값, 변수의 자료형 확인
  - 보통 print()와 함께 사용
  - print(type(변수명))

```
number1 = 100
number2 = 1.2

print(type(number1))
print(type(number2))
```



```
<class 'int'>
<class 'float'>
```

문자형 데이터의 활용

파이썬 문자형(String) 데이터

- **문자 및 문자열 데이터**
  - 문자(Character) - 1개의 문자 (‘a’, ‘b’, ‘C’)
  - 문자열(String) - 1개 이상의 연속 된 문자로 구성(“abcde”, “CodePresso”, “Python”)

- **숫자형 데이터와 함께 가장 많이 사용 되는 타입**
  - 페이스북 게시물, 블로그 댓글, 사람 이름, 집 주소, …

- **파이썬 문자형 데이터는 큰 따옴표, 작은 따옴표를 사용**
  - 'CodePresso’
  - “CodePresso”

문자형 변수의 생성 및 자료형 확인

```
name = "CodePresso"
print(name)
print(type(name))
```

문자형 데이터를 변수에 저장

name 변수 안에 저장 되어 있는 문자형 데이터 출력

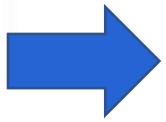
name 변수 안에 저장 되어 있는 데이터의 자료형 출력

```
CodePresso
<class 'str'>
```

문자형 데이터의 연결(Concatenation)

- '+' 연산자를 활용하여 2개 이상의 문자형 데이터를 연결
  - "Hello" + " " +"Python!"

```
a = "Code"
b = "Presso"
print(a + b)
```



```
CodePresso
```

문자열의 인덱스(index)

- **값이 연속 되어 있는 형태의 데이터는 순서가 존재**
  - Sequence Data Type - 문자열, 리스트, 튜플, ...

- **“CodePresso” 문자열**

문자열	C	o	d	e	P	r	e	s	s	o
인덱스	0	1	2	3	4	5	6	7	8	9

문자열의 인덱싱(indexing)

- **인덱싱(indexing) - 인덱스를 활용하여 연속 된 값 중 특정 값만 참조**
  - 문자형 변수 이름 오른편에 대괄호 [] 사용, [] 안에 인덱스 번호 지정

```
name = "CodePresso"
print(name[0])
```

C

문자열	C	o	d	e	P	r	e	s	s	o
인덱스	0	1	2	3	4	5	6	7	8	9

```
name = "CodePresso"
print(name[4])
```

P

문자열	C	o	d	e	P	r	e	s	s	o
인덱스	0	1	2	3	4	5	6	7	8	9

문자열의 슬라이싱(slicing)

- 인덱스의 시작/끝 값을 지정하여 연속 된 일부 데이터를 참조
  - 인덱싱은 1개의 단일 값을 가져 옴
  - 슬라이싱은 1개 이상의 연속 된 값을 가져 옴
- 인덱싱

문자열	C	o	d	e	P	r	e	s	s	o
인덱스	0	1	2	3	4	5	6	7	8	9

- 슬라이싱

문자열	C	o	d	e	P	r	e	s	s	o
인덱스	0	1	2	3	4	5	6	7	8	9

문자열의 슬라이싱(slicing) 활용

- 슬라이싱 문법 - 변수명[시작 인덱스:끝 인덱스]
  - 시작 인덱스의 값을 포함
  - 끝 인덱스의 값은 포함하지 않음, 끝 인덱스 바로 직전 인덱스 까지 포함

```
print(name[0:4])
```

→ Code

문자열	C	o	d	e	P	r	e	s	s	o
인덱스	0	1	2	3	4	5	6	7	8	9

```
print(name[4:10])
```

→ Presso

문자열	C	o	d	e	P	r	e	s	s	o
인덱스	0	1	2	3	4	5	6	7	8	9

인덱스 에러(IndexError)

- 시퀀스형 데이터의 인덱싱 시 인덱스의 범위를 벗어나면 에러 발생

```
name = "CodePresso"
print(name[10])
```

```
Traceback (most recent call last):
  File "./prog.py", line 34, in <module>
IndexError: string index out of range
```

문자열	C	o	d	e	P	r	e	s	s	o	?
인덱스	0	1	2	3	4	5	6	7	8	9	

변수의 이름 규칙

변수의 이름 규칙

- 알파벳 대소문자와 숫자, 언더스코어만 사용 가능
- 알파벳 대소문자와 언더스코어로만 시작 가능(숫자로 시작 불가)
- 알파벳 대소문자를 구별함(name, Name, NAME은 서로 다른 변수)

변수의 이름 예시

```
myname = "CodePresso"
my_name = "CodePresso"
myName = "CodePresso"
my_name2 = "CodePresso"
_my_name = "CodePresso"
```

```
2my_name = "CodePresso"
my_name! = "CodePresso"
my-name = "CodePresso"
```

```
SyntaxError: invalid syntax
```

좋은 변수 이름

- **코드의 가독성을 위해 변수의 이름은 최대한 의미 있게 지어야 함**
  - a, b, n 등과 같이 의미를 파악하기 어려운 이름은 지양
- **변수는 소문자만 사용**
  - name, address, height, width, …
- **단어와 단어 연결 시 언더스코어 “_” 사용**
  - my_name, first_name, last_name

파이썬 에러와 NameError 처리

파이썬 에러

- **파이썬 코드 또는 실행된 프로그램에서 문제가 발생한 상태**

- **문법 에러**
  - 파이썬의 문법 규칙을 따르지 않은 코드에서 발생

```
2my_name = "CodePresso"
```

```
  File782.py", line 1
    2my_name = "CodePresso"
     ^
SyntaxError: invalid syntax
```

- **예외**
  - 문법은 잘 지켰지만, 프로그램 실행 중에 특정 원인에 의해 에러가 발생

```
my_name = "CodePresso"
print(my_name[10])
```

```
Traceback (most recent call last):
  File "./prog.py", line 2, in <module>
IndexError: string index out of range
```

파이썬 에러의 처리

- **프로그래밍 시에 다양한 에러를 만나게 됨**

- **발생한 에러의 의미를 최대한 빨리 파악 해야 함**
  - 에러의 정보를 바탕으로 원인 분석

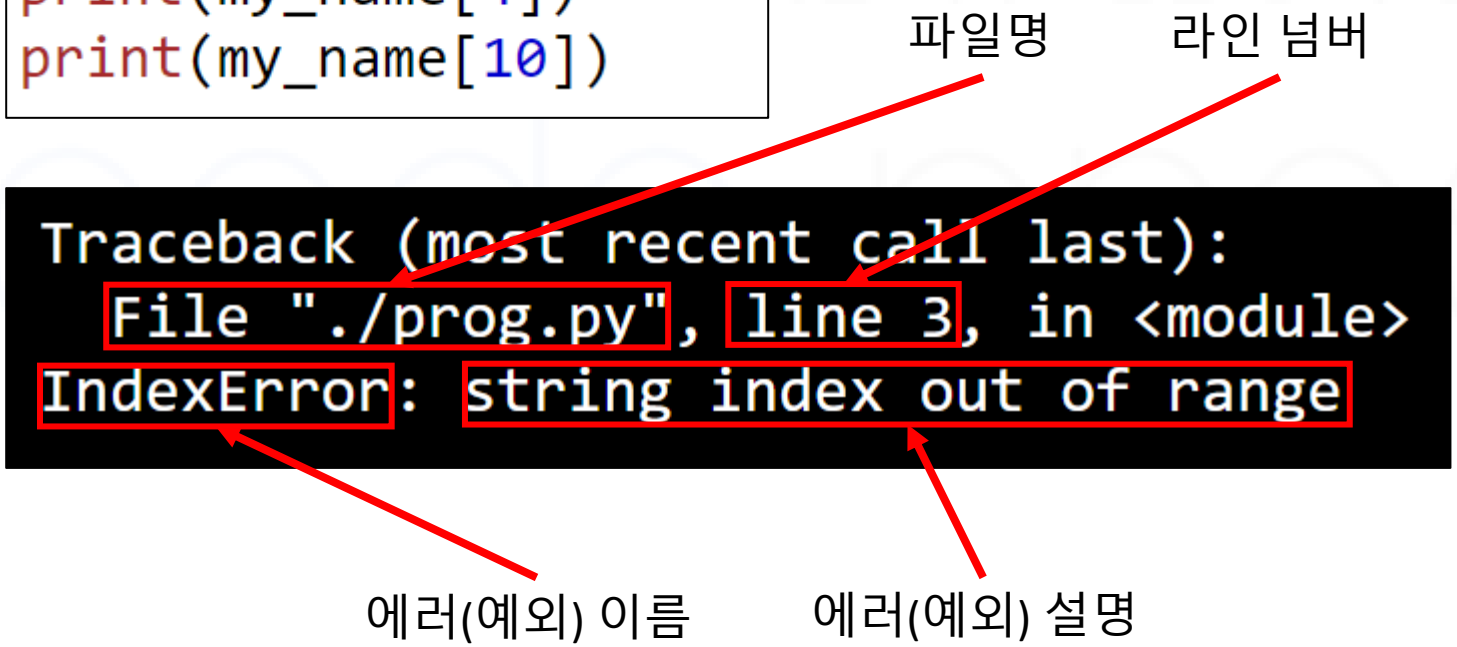
- **에러의 의미가 원인을 파악한 후 해결 방법을 찾아야 함**
  - 경험
  - Googling

파이썬 에러의 해석

- 에러의 이름, 에러의 설명, 파일명, 라인 넘버를 확인

```
my_name = "CodePresso"
print(my_name[4])
print(my_name[10])
```

```
Traceback (most recent call last):
  File "./prog.py", line 3, in <module>
IndexError: string index out of range
```



변수와 NameError

- **생성한 적 없는 변수를 사용하려고 할 때 발생**
  - 대부분의 경우 변수 이름에 오타가 있을 경우 발생
  - NameError를 만나면, 변수에 오타가 없는지 가장 먼저 확인

```
my_name = "CodePresso"
print(my_nama)
```

```
Traceback (most recent call last):
  File "./prog.py", line 2, in <module>
NameError: name 'my_nama' is not defined
```