CODINGON X posco

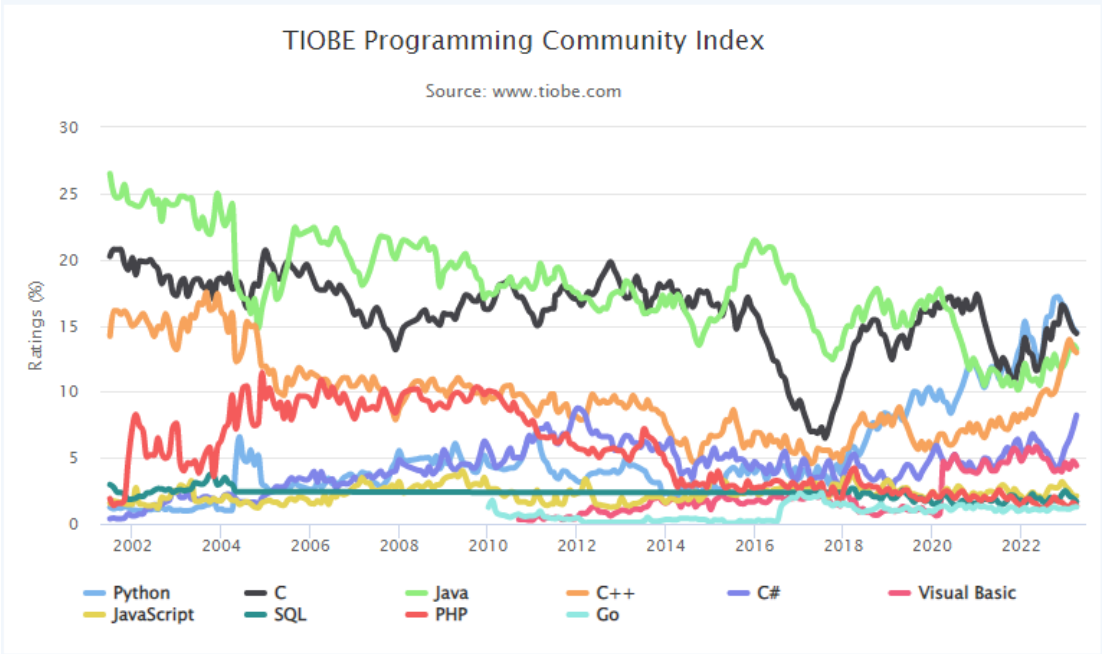
K-Digital Training 스마트 팩토리 3기

Python 기초

Python

- Python은 프로그래밍 언어로 세계에서 가장 많이 사용되는 언어 중 하나입니다.

Apr 2023	Apr 2022	Change	Programming Language	Ratings	Change
1	1		Python	14.51%	+0.59%
2	2		C	14.41%	+1.71%
3	3		Java	13.23%	+2.41%
4	4		C++	12.96%	+4.68%
5	5		C#	8.21%	+1.39%
6	6		Visual Basic	4.40%	-1.00%
7	7		JavaScript	2.10%	-0.31%



출처: TIOBE (티오베)

Python 장점

- 코드가 간결하고 문법이 쉽다.
- 다양한 분야 활용 가능
  - Ex. 웹 개발, 해킹 도구, AI, 데이터 분석 등
- C, C++, JAVA 등 다른 프로그래밍 언어와 쉽게 통합할 수 있다.
- 많은 기능들을 갖고 있는 라이브러리가 풍부해 그대로 가져다가 사용할 수 있어 쉽게 개발이 가능하다.

Python 설치전 주의사항

- 사용자명이 한글로 되어있는 경우 많은 경우에서 에러발생
- 사용자명을 영어로 바꾸거나 영어이름의 사용자 추가하여 설치

Python 설치(비권장)

- 공식 홈페이지에서 다운 및 설치 가능
- https://www.python.org/

Downloads -> Download for Windows Python 3.11.3

* [Add Python 3.11 to Path] 항목 체크

Python 실행 방법(비권장)

- IDLE (Integrated Development Environment)
- 통합개발환경
- 파이썬과 함께 기본적으로 설치되는 텍스트 에디터

Python 설치(Anaconda)

- 공식 홈페이지에서 다운 및 설치 가능
- https://www.anaconda.com/)

free Download -> Download

- Anaconda를 path에 등록하고 기본 python으로 설정

용어 정리

표현식 & 문장

- 표현식 (expression)
  - 어떠한 **값을 만들어내는** 간단한 코드
  - 값: 숫자, 수식, 문자열 등
- 문장(statement)
  - 표현식이 하나 이상 모인 것
  - 파이썬에서는 한 줄이 문장
- 프로그램(program)
  - 문장이 모인 것

```
273
10 + 20 + 30 * 10
"Python Programming"
```

키워드

- 특정한 의미가 부여된 단어
- 파이썬이 만들어질 때 예약해 놓은 것
- 이미 특정 기능을 수행하고 있기 때문에, 사용자가 이름을 정할 때 키워드를 사용하면 안 됨
- 코드 전용 에디터에서 구분해 줌 (다 외울 필요는 없다)

```
import keyword
print(keyword.kwlist) # 파이썬에서 사용하는 키워드 출력 명령어
```

주석

- 프로그램의 진행에 **전혀 영향을 주지 않는** 코드
- 프로그램을 설명하기 위해 사용
- # 사용 – 한 문장 처리
- 쌍 따옴표 or 단 따옴표 3개 – 긴 문장

```
# 주석으로 컴퓨터가 해석하지 않는 부분임.

'''
긴 문장을 주석처리하는 경우
단따옴표 3개로 작성 가능
'''


"""
쌍 따옴표 3개로 표현 가능~!
"""
```

연산자와 피연산자

- 연산자: 값과 값 사이에 기능을 적용할 때 사용, 사칙연산 등
- 피연산자: 연산에 쓰이는 자료값

```
>>> 1 + 1
2
>>> 10 - 10
0
```

출력: print()

```
print("Hello, World")
print("Hello", "Python") # 여러 개 출력
```

- 메시지 출력 기능을 담당하는 함수
- 여러 개 출력 시 쉼표 (,)로 구분 -> 무조건 띄워 쓰기가 하나 들어감
  - 띄워 쓰기 없애려면 sep 옵션 추가

```
print("Hello", "World!", sep="")
```

- 문자열, 숫자 가능

```
print( "dkssudkflasd" + 44 ) #문법 오류!
print( "dkssudkflasd", 44 ) #올바른 문법
```

- 괄호 안에 아무것도 넣지 않으면 줄 바꿈 역할
- 엔터 없이 출력하고 싶다면, end 옵션을 추가하기

```
print( "안녕하세요.", end="" )
print( "코딩온입니다." )
```

추가) f 문자열 포매팅

- 파이썬 3.6 버전 부터 사용 가능한 기능
- 문자열 앞에 f 접두사를 붙이면 f 문자열 포매팅 기능을 사용 할 수 있다

```
>>> name = '홍길동'
>>> age = 30
>>> f'나의 이름은 {name}입니다. 나이는 {age}입니다.'
'나의 이름은 홍길동입니다. 나이는 30입니다.'
```

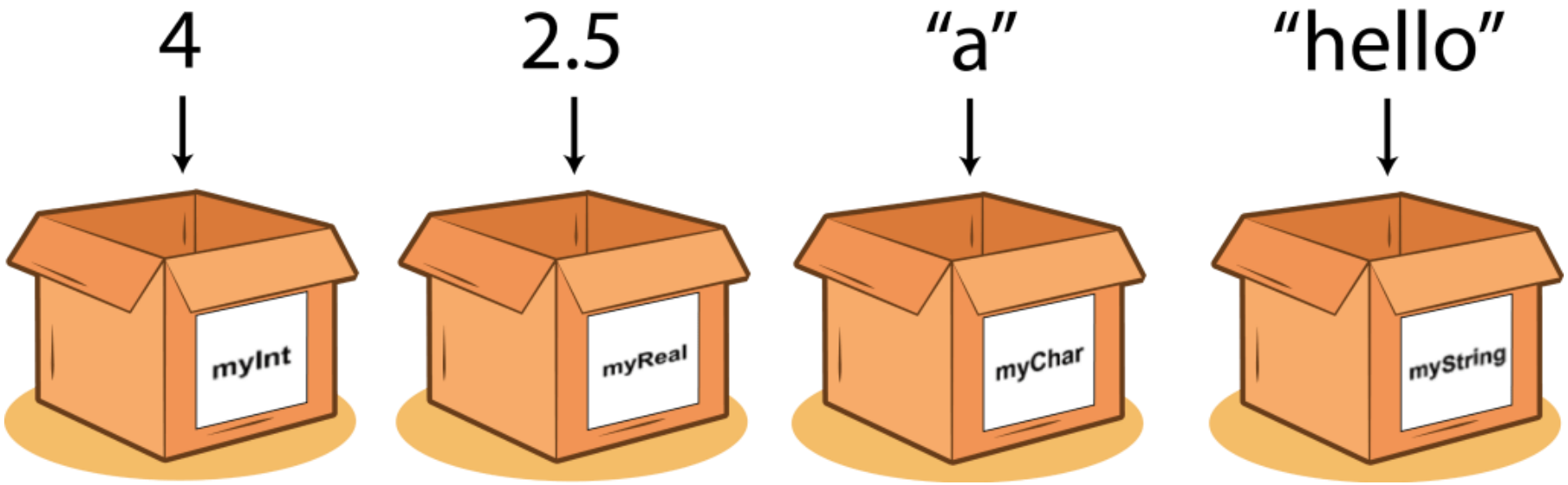
- name, age 같은 변수 값을 생성한 후에 그 값을 참조할 수 있음
- 표현식 지원 (문자열 안에서 변수와 +,- 같은 수식 함께 사용 가능)

```
>>> age = 30
>>> f'나는 내년이면 {age+1}살이 된다.'
'나는 내년이면 31살이 된다.'
```

변수

변수

- 변수 = variable, 변하는 값
- 변수는 데이터를 담는 빈 그릇!



변수 사용하기

- 등호 기호(=)를 사용해 데이터 저장

[변수 이름] = [변수에 저장할 데이터]

- 변수 이름은 식별자 네이밍 규칙이 적용 됨
- 키워드나 print()와 같은 함수명은 변수로 사용 안됨

```
ive = "I AM"
print(ive)
```

```
print = "a"

print( print )

#문법 오류!!
```

식별자

- 프로그래밍 언어에서 **이름을 붙일 때** 사용하는 단어
- 주로 변수, 함수 이름 등으로 사용함
- 규칙
  - 첫 문자는 알파벳 문자이거나 밑줄이어야 함
  - 나머지 문자는 문자, 밑줄(_) 또는 숫자(0-9)여야 함
  - **대소문자 구분**
  - 키워드는 사용 불가
  - snake_case : 함수나 변수 이름에 사용, CamelCase : 클래스 이름에 사용

hi apple (x) → 공백

hello* (x) → 특수문자

hello_ (o) → 특수문자이기는 하지만, _ 허용

hello23 (o)

1hello (x) → 숫자로 시작 불가능

식별자 규칙

- snake_case
  - 함수나 변수에 사용
  - 단어 사이에 언더바(_) 기호로 식별자 만듦
- CamelCase
  - 클래스에 사용
  - 단어들의 첫 글자를 대문자로 만들어 식별자 만듦
    - CamelCase (O)
    - camelCase (x)
  - 첫번째 글자를 대문자 or 첫번째 글자는 소문자 : 두 종류 있으나 파이썬 에서는 첫 글자는 대문자로 적는 방법만 사용함

자료형

자료형

- 숫자형
  - 정수형: 10, 5, -1
  - 실수형: 11.1, 10.0
- 문자열: 문자의 양쪽 끝을 따옴표로 감싸서 표기

⇒차이점: 숫자형은 수리적 계산 가능

- type() 을 통해 자료형 알 수 있음

```
>>> type(77)
<class 'int'>
>>> type(91.5)
<class 'float'>
>>> type("77")
<class 'str'>
```

자료형

- 이스케이프 문자 : 원래 역할에서 벗어나게 하는 문자
- Q. 만약 따옴표 자체를 문자 데이터로 표시하고 싶다면?

```
>>> print('What\'s your name?') # \기호 사용
What's your name?

# \ : ", ', \ 등을 표현하기 위해
# \n : 줄바꿈
# \t : 탭

# or
>>> print("'오늘 저녁 뭐먹지?' 라고 생각하는 중이다.")
# 작은 따옴표를 문자 데이터로 표시하고 싶다면 큰 따옴표로 문자열 감싼다
```

연산자

숫자형 연산하기

- 사칙 연산
  - 더하기, 빼기, 곱하기, 나누기
  - +, -, *, /
- 특별한 연산자
  - 정수 나누기 연산자 : // , 정수 부분 (몫) 만 출력
  - 나머지 연산자: % , 코딩에서 자주 사용
  - 제곱 연산자 : **, 어떤 수를 여러 번 곱하는 연산

```
print(3 // 2) # 1
print(3.25 // 2) # 1.0
```

```
print(11 % 3) # 2
print(3.25 % 7) # 3.25
```

숫자형 연산하기

- 연산자 우선순위
  - 수학에서 곱셈과 나눗셈이 덧셈, 뺄셈보다 우선순위가 높은 것처럼 프로그래밍에서도 연산자 사이에 우선순위가 있음

1. 곱하기(*), 나누기(/)
2. 더하기(+), 빼기(-)

- 소괄호 () 사용하기
  - 괄호를 이용해 수식의 가독성을 높임
  - 명시적으로 우선 순위를 지정

```
>>> 2 + 3 * 4
14
>>> 2 + (3 * 4)
14
>>> (2 + 3) * 4
20
>>>
>>> 2 ** 5 // 3 + 1
11
>>> 2 ** ((5 // 3)  + 1)
4
>>>  (2 ** 5) // (3 + 1)
8
```

문자열 연산하기

- + 연산자 : 문자 연결 기능, 더하기
- * 연산자 : 문자 반복해서 연결하는 기능

```
>>> name = '코디'
>>> age = ' 5살'
>>> name + age
'코디 5살'
```

```
print("안녕" + "반가워")
print("!" * 10)
```

```
>>> python = '파이썬'
>>> python * 3
'파이썬파이썬파이썬'
```

문자열 함수

문자열 함수

- len(): 문자열의 문자 개수를 반환
  - 공백, 특수 문자도 한 개의 문자로 인식

```
print(len("바나나"))
print(len("Hello, World!")
```

- count(): 찾을 문자열이 몇 개 들어있는지 개수를 반환
  - 해당 문자를 찾지 못하면 -1을 반환

```
print("banana".count("a")) # banana에서 a의 개수 알려줌
# 3
```

문자열 함수

- upper(): 문자열을 알파벳 대문자로 반환
- lower(): 문자열을 알파벳 소문자로 반환

```
upper_case = "HellO".upper()
lower_case = "HELLo".lower()

print(upper_case, lower_case) # HELLO hello
```

- find(), rfind(): 왼쪽, 오른쪽부터 해당 문자의 위치 찾음
  - 문자가 여러 개인 경우 처음 등장하는 위치 찾음

```
result1 = "코딩온. 저는 코딩온입니다.".find("코딩온")
result2 = "코딩온. 저는 코딩온입니다.".rfind("코딩온")

print(result1, result2) # 0 8
```

문자열 함수

- split(): 문자열을 공백이나 다른 문자로 나누어 리스트로 반환
  - 문자열을 특정한 문자로 자를 때 사용!
  - 괄호 안의 문자 기준으로 자름
  - 실행 결과 -> 리스트 라는 데이터 형식
  - 리스트? 여러 값 할당하는 자료형

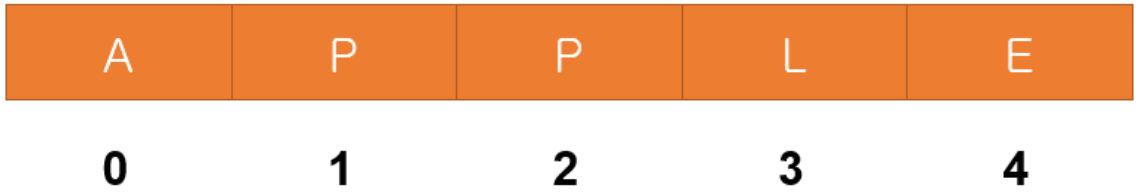
```
# case1
friends = "원영 유진 레이"
print(friends.split(" "))

# case2
email = "luna@spreatics.com"
print(email.split("@"))

# ['원영', '유진', '레이']
# ['luna', 'spreatics.com']
```

문자열 인덱싱

- 원소의 위치, 순서를 인덱스라고 부른다.
- 인덱스 번호는 0부터 시작
- 문자열 슬라이싱
  - 슬라이스 치즈처럼 문자열에서 범위를 지정해 일부를 잘라낸 것
  - 문자열[시작 위치:끝 위치]




```
"LOVEDIVE"[0:4] # "LOVE"
# 0번 인덱스 부터 3번 인덱스까지 -> 아하! 마지막 인덱스는 포함하지 않는구나
```

입력 함수

입력: input()

- 메시지 입력 기능을 담당하는 함수
- 사용자마다 다른 값을 입력 받아서 사용
- input("프롬프트 문자열")

```
input("요즘 자주 듣는 노래는? ")
```

* input으로 입력 받은 값은 문자열!

```
# step1
song = input("너의 최애 노래는? ")
print(song)

# step2
print(type(song)) # song 변수의 데이터 자료형 확인
```

형변환

- 문자 <-> 숫자, 자료형을 바꿔주는 것
- int() : 정수형으로 변환
- float() : 실수형으로 변환

```
str1 = input("str1 (정수 값): ")
int1 = int(str1)
print(str1, int1)
print(type(str1), type(int1))

str2 = input("str2 (실수 값): ")
float2 = float(str2)
print(str2, float2)
print(type(str2), type(float2))
```

주의) 변환이 불가능한 데이터를 변환하고자 하는 경우 Value Error 발생!

- 숫자가 아닌 걸 숫자로 변환하려고 할 때, ex) int("name")
- 소수점이 있는 숫자 형식의 문자열을 정수형으로 변환하려고 할 때,
  ex) int("11.2")