기초 문법의 이해

리터럴 Literal

: 소스코드 상에서 내장 자료형의 상수 값을 나타내는 용어

* 다양한 형태가 있음
* 정수형, 부동소수점 숫자형, 문자열, 부울형, 리스트형
* type() : 리터럴의 자료형을 확인하고 싶을 때
  + 파이썬은 값에 의해 자료형이 결정되는 동적 타이핑 언어라서 type을 확인해야 할 때가 있음
* int = 정수형, float = 부동소수점 숫자형, str = 문자열형, bool = 부울형, list = 리스트형

숫자형

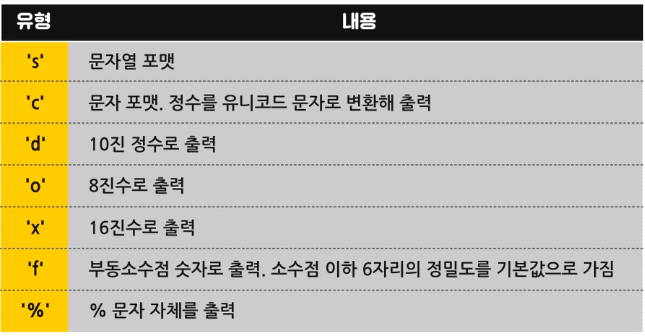
: 정수형, 부동소수점형 (3.14), 허수형 (3 + 2i)

* 정수
  + 양, 0, 음의 정수 다 사용 가능
  + 0o 접두어: 8진수 사용
  + 0x 접두어: 16진수
  + 0b 접두어: 2진수
  + 리터럴 내의 \_는 무시됨
* 부동소수점형
  + 양, 음 사용 가능
  + 10. 과 같이 소수부 생략 혹은 .001과 같이 정수부 생략 가능
  + 지수 (e)로 표현 가능
  + 리터럴 내의 \_는 무시됨
* 허수형
  + 파이썬의 허수형은 j 접미사를 사용함
  + 리터럴 내의 \_는 무시됨

문자열

: 문자들의 집합

* “안녕하세요” → 5음절의 문자열로, 길이가 5
* 파이썬은 자료형으로서의 문자형은 제공하지 않음
* “”안에 ‘’ 포함 가능; 그러나 ‘’로 시작한 문자열에 “”를 안에 넣으면 syntax error가 뜸
  + “‘시간의 역사’ 남기고 떠난 스티브 호킹…”아웃풋을 원하면
  + ‘\’시간의 역사\’ 남기고 떠난 스티븐 호킹…’으로 입력해야한다
  + 작은따옴표’ 문자 자체를 의미하는 이스케이프 시퀀스 작은따옴표 \’ 혹은 “ 문자를 의미하는 \”
* 이스케이프 시퀀스 \: 프로그램의 소스 코드 내에서 사용할 수 있도록 백슬래시 기호와 조합해서 사용하는 사전에 정의해둔 문자 조합으로 문자열의 출력 결과를 제어하기 위해 사용함
  + \\: 백슬래시 \; 하나만 있을 때 따옴표와 만나면 새로운 이스케이프 시퀀스가 됨
  + \’: 작은 따옴표 ‘
  + \”: 큰 따옴표
  + \n: enter; 줄바꿈
  + \t: tab; 탭
* **문자열 포맷팅**
* 문자열 내에 사용된 문자열 표시 유형 (문자열 포맷 코드)을 특정 값으로 변경하는 기법



* 문자열 %s 사용
  + “이름: %s\n나이: %s세” % “홍길동, 20” 튜플형
  + “이름: %(name)s\n나이: %(age)s 세” % {“name”:”홍길동”, “age” : 20} 사전형
  + “%10s” % “우측정렬”
    - 문자열의 폭이 10, 정렬의 방향이 우측임
    - ‘      우측정렬’ → 왼쪽에 6개 (총 길이가 10이라)의 빈 문자열이 생성됨
  + “%-10s” % “좌측정렬”
    - ‘좌측정렬      ‘ → 우측에 6개의 공백문자
* 정수형 %d, 유니코드 문자 변환 %c
  + “%d %o %x” % (10, 10, 10)
    - ‘10 12 a’ → 정수 10에 대해 8진 정수 12, 16진 정수 a
  + “%s %d %x” % (“가”, ord(“가”), ord(“가”))
    - 유니코드 ‘가’의 값은 44032
    - ‘가 44032 ac00’
* 부동소수점 숫자 %f
  + “%f %d” % (3.14, 3.14)
    - ‘3.140000 3’ → f는 소수점 6자리까지, d는 정수형만 출력되기 때문
  + %% : % 문자 자체가 출력값이 됨
  + “%d 점은 상위 %d%%에 속합니다.” %(98, 1)
    - 98 점은 상위 1%에 속합니다.
  + “%0.2f” % 3.141592 → 소수점 2자리까지 표시
    - ‘3.14’
  + “%10.2f” % 3.141592 → 부동소수점을 포함한 전체 자릿수를 10으로 함
    - ‘      3.14’ → 앞에 6개의 공백 문자열을 가진 소수점 2자릿수 부동소수점 출력
  + “%010.2f” % 3.141592 → 전체 자릿수는 10, 소수점 두자리까지 표시, 앞의 남은 자리를 0으로 채움
    - ‘0000003.14’
* **str.format() 함수를 이용한 문자열 포맷팅**
* 전달할 인자의 위치 인덱스는 0부터 시작
* “이름: {0}, 나이: {1} 세”.format(“홍길동”, 20)
  + 값의 위치로 지정
  + {}안을 공백으로 두면 해당 인자를 순차적으로 적용해 변환된 결과를 생성함
* “{0:c} => {1}”.format(97, 97)
  + {0:c} → 첫 번째 인자를 유니코드 a로 변환
  + {1} → 97 문자열 그대로
  + ‘a => 97’ → 고로 a는 97보다 크거나 같다라는 문자열 출력
* “{0}, {1}, {2:x}”.format(“가”, ord(“가”), ord(“가”))
  + {0} = 가
  + {1} = 44032
  + {2:x} = 44032의 16진수, ac00으로 변환
  + ‘가, 44032, ac00’ 출력
* “{0:f} {1:.2f}”.format(3.14, 3.14)
  + {0:f} = 3.140000
  + {1:.2f} = 3.14
  + ‘3.140000, 3.14’ 출력
* 이름=값 형식으로 인자를 구성하면 이름을 이용해 인자 전달 가능
  + “이름: {name}, 나이: {age} 세”.format(name=”홍길동”, age=20)
    - ‘이름: 홍길동, 나이: 20 세
* 문자열 폭, 정렬방향 지정
  + “{0:<10}”.format(“좌측정렬”)
    - <는 정렬의 방향이 좌측임을 나타냄
    - ‘좌측정렬      ’ 출력 (공백 6개)
  + “{0:>10}”.format(“우측정렬”)
    - >는 정렬의 방향이 우측임을 나타냄
    - ‘      우측정렬’ 출력 (공백 6개)
  + “{0:^10}”.format(“중앙정렬”)
    - ^는 정렬의 방향이 중앙임을 나타냄
    - ‘   중앙정렬   ‘ 출력 (좌우로 빈칸 3개)
  + “{0:\*^10}”.format(“중앙정렬”)
    - \*는 공백 채울 문자
    - ‘\*\*\*중앙정렬\*\*\*’ 출력
* 부동소수점 숫자 폭, 소수점 이하 정밀도 표현
  + “{0:0.2f}”.format)3.141592)
    - ‘3.14’ 출력
  + “{0:10.2f}”.format(3.141592)
    - ‘      3.14’ 출력 (공백 6칸)
  + “{0:010.2f}”.format(3.141592)
    - ‘0000003.14’ 출력
  + “{{ {0:.1f} }}”.format(98.5)
    - ‘{ 98.5 }’ 출력 → “{{}}”는 ‘{}’문자열로 나타남

주석

: 프로그램의 코드 앞에 #을 붙여 작성된 부분으로, 소스 코드에 대한 상세 설명을 달거나 특정 코드를 실행하지 않을 목적으로 사용됨