Python Study

Part 2

[if 문]

1. 조건을 판단한 후 상황에 맞게 처리할 수 있다.
   - money에 True를 입력했으므로 money는 참이다.
   - 따라서 if문 다음 문장이 수행되어 '택시를 타고 가라'가 출력된다.

```
money = True

if money:
    print("택시를 타고 가라")
else:
    print("걸어 가라")
```

```
택시를 타고 가라
```

2. 들여쓰기가 매우 중요!
   - 들여쓰기는 Tab 또는 Space로 가능하다.
   - 둘을 혼용해서 사용하는 건 비추천

3. If 조건문 - 참과 거짓을 판단하는 문장 (위 예제에서는 `money`)
4. If 조건문 뒤에 콜론(:) 사용!

[if 문]

5. If 조건문에 연산자 사용
   - 조건문 `money >= 3000` 이 false 이므로 else 문이 실행

```
money = 2000

if money >= 3000:
    print("택시를 타고 가라")
else:
    print("걸어가라")
```

```
걸어가라
```

   - 조건문 `money >= 3000` 는 false, `card` 는 true / or 연산자 사용했으므로 조건문은 true

```
money = 2000
card = True
if money >= 3000 or card:
    print("택시를 타고 가라")
else:
    print("걸어가라")
```

```
택시를 타고 가라
```

[if 문]

6. If 조건문에 “x in s”, “x not in s”

- **다른 프로그래밍 언어에서 볼 수 없는 문법**

in	not in
x in 리스트	x not in 리스트
x in 튜플	x not in 튜플
x in 문자열	x not in 문자열

```
pocket = ['paper', 'cellphone', 'money’]

if 'money' in pocket:
    print("택시를 타고 가라")
else:
    print("걸어가라")
```

```
택시를 타고 가라
```

[if 문]

7. Pass - 조건문에서 아무 일도 하지 않게 설정

- "주머니에 돈이 있으면 가만히 있고 주머니에 돈이 없으면 카드를 꺼내라."
- **Pass가 수행되어 아무 일도 일어나지 않는다.**

```
pocket = ['paper', 'money', 'cellphone']

if 'money' in pocket:
    pass
else:
    print("카드를 꺼내라")
```

[if 문]

8. elif 문

- **다양한 조건을 판단**
- **If, else 만으로는 한 가지 조건에 대해서만 실행할 수 있어 제한적**

- "주머니에 돈이 있으면 택시를 타고, 주머니에 돈은 없지만 카드가 있으면 택시를 타고, 돈도 없고 카드도 없으면 걸어 가라."

```
pocket = ['paper', 'handphone']
card = True
if 'money' in pocket:
    print("택시를 타고가라")
else:
    if card:
        print("택시를 타고가라")
    else:
        print("걸어가라")
```

```
택시를 타고가라
```

elif 사용

```
pocket = ['paper', 'cellphone']
card = True

if 'money' in pocket:
    print("택시를 타고가라")
elif card:
    print("택시를 타고가라")
else:
    print("걸어가라")
```

```
택시를 타고가라
```

[문제 1]

1. 다음 코드의 결과값은 무엇일까?

```
a = "Life is too short, you need python"

if "wife" in a: print("wife")
elif "python" in a and "you" not in a: print("python")
elif "shirt" not in a: print("shirt")
elif "need" in a: print("need")
else: print("none")
```

[if 문]

9. 조건부 표현식

- **간단하게 표현 가능**
- **조건부 표현식 :** `조건문이 참인 경우` if `조건문` else `조건문이 거짓인 경우`

- score가 60 이상일 경우 message에 문자열 "success"를, 아닐 경우에는 "failure"를 대입하라.

```
if score >= 60:
    message = "success"
else:
    message = "failure"
```

조건부 표현식 사용

```
message = "success" if score >= 60 else "failure"
```

[반복문]

1. 반복해서 문장을 수행해야 할 경우 사용
2. 조건문이 참인 동안에 반복문 아래의 문장이 반복해서 수행

3. For 문 – 매우 유용

- i가 list의 element를 출력
- list를 다 돌면 exit

```
list = [1, 3, 5, 7]

for i in list:
  print(i)
```

```
1
3
5
7
```

- 한 줄 for문

```
word = ["Python", "Java", "C++"]
word = [i.upper() for i in word]
print(word)
```

```
['PYTHON', 'JAVA', 'C++']
```

4. while 문

- i가 0이 되면 exit
- "i -= 1" 은 "i = i-1"과 같다.

```
i = 3
while i != 0:
  print(i)
  i -= 1
```

```
3
2
1
```

- 무한 루프 (영원히 반복)

```
while True:
    print("Ctrl+C를 눌러야 while문을 빠져나갈 수 있습니다.")
```

```
Ctrl+C를 눌러야 while문을 빠져나갈 수 있습니다.
Ctrl+C를 눌러야 while문을 빠져나갈 수 있습니다.
Ctrl+C를 눌러야 while문을 빠져나갈 수 있습니다.
Ctrl+C를 눌러야 while문을 빠져나갈 수 있습니다.
Ctrl+C를 눌러야 while문을 빠져나갈 수 있습니다.
Ctrl+C를 눌러야 while문을 빠져나갈 수 있습니다.
```

[문제 2]

While 문을 사용해 다음과 같이 별(*)을 표시하는 프로그램을 작성해 보자.

```
*
**
***
****
*****
```

조건1. While 문 사용

예시)

```
i = 3
while i != 0:  #i가 0이 아니면
  print(i)     #i를 출력해라
  i -= 1       #i-1 수행
```

```
3
2
1
```

[반복문]

5. Continue – 건너뛴다
   - 짝수이면 다음 상황으로 건너뛰고(continue), 홀수이면 출력
   - range(1,10) : j에 1에서 10까지의 값이 차례대로 들어간다.

```
for j in range(1, 10):
  if j % 2 == 0: #짝수
    continue
  else: #홀수
    print(j)
```

```
1
3
5
7
9
```

6. Break – 반복문 exit
   - 짝수이면 반복문을 나가고(break), 홀수이면 출력한다.

```
for j in range(1, 10):
  if j % 2 == 0: #짝수
    break
  else: #홀수
    print(j)
```

```
1
```

[문제 3]

While 문을 사용해 1부터 1000까지의 자연수 중 3의 배수의 합을 구해 보자.

조건1. While 문 사용

예시)

```
i = 3
while i != 0:  #i가 0이 아니면
  print(i)     #i를 출력해라
  i -= 1       #i-1 수행
```

```
3
2
1
```

조건2. Continue 사용

예시)

```
for j in range(1, 10):
  if j % 2 == 0: #짝수이면
    continue
  else: #홀수이면
    print(j)
```

```
1
3
5
7
9
```

[함수]

1. 똑같은 내용을 반복해서 작성하는 것은 비효율적
2. 프로그램을 함수화하면 프로그램 흐름을 알아보기 쉽다.

3. 두 숫자를 매개변수로 전해주면 두 수의 합을 return 하는 함수 add() 만들기

```
def add(num1, num2):
    return num1+num2

a=3
b=4
sum = add(a,b) #return 값을 sum에 저장

print(sum)
```

```
7
```

4. 들여쓰기 매우 중요!
5. 함수 이름 뒤에 콜론(:) 꼭 쓰기!

[함수]

6. 값을 입력받아 제곱해주는 계산기 함수 만들기
   - insert() : 숫자 입력받아 return 해주는 함수
     - input() : 파이썬에서 제공하는, 값을 터미널에서 입력받도록 하는 함수
     - Return 해준다 = main 문에서 변수로 저장
   - sqa(a) : 제곱한 값을 출력해주는 함수
     - int(number)를 매개변수 a로 받아 함수 내에서 a라는 변수로 사용 가능

```
def insert():
  print("값을 입력해주세요: ")
  num = input()
  return num

def sqa(a):
  print(a**2)

number = insert() #insert() 함수 호출
sqa(int(number))  #spa()함수 호출
```

```
값을 입력해주세요:
5
25
```

[문제 4]

주어진 자연수가 홀수인지 짝수인지 판별해 주는 함수 is_odd()를 작성해 보자.

조건1. If 문 사용

예시)

```
money = True   #돈이 있는 상태
if money:      #money가 true이면
    print("택시를 타고 가라")
else:          #money가 false이면
    print("걸어 가라")
```

```
택시를 타고 가라
```

조건2. 함수 사용

예시)

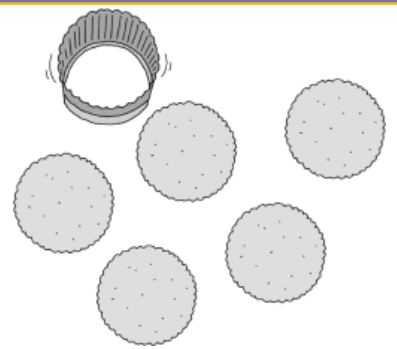
```
def add(num1, num2): #add()함수 정의
    return num1+num2

a=3
b=4
sum = add(a,b) #return 값을 sum에 저장
print(sum)
```

```
7
```

파이썬 문법 - 클래스

[클래스]



클래스로 표현

1. 클래스와 객체
   ① 과자 틀 = 클래스 (class)
   ② 과자 틀에 의해서 만들어진 과자 = 객체 (object)

2. 함수의 한계
   ① 한 프로그램에서 2대의 계산기가 필요한 경우
   ② 각 결과값 유지해야 하므로 함수를 따로 생성
   ③ 필요한 계산기 수가 증가할수록 부담

3. Calculator 클래스로 별개의 object cal1, cal2 생성
   ① 각 객체 cal1, cal2가 각각의 역할을 수행한다.
   ② 계산기(cal1, cal2)의 결과값은 서로 독립적인 값을 유지

```
result1 = 0
result2 = 0

def add1(num):
    global result1
    result1 += num
    return result1

def add2(num):
    global result2
    result2 += num
    return result2

print(add1(3))
print(add1(4))
print(add2(3))
print(add2(7))
```

```
3
7
3
10
```

```
class Calculator:
    def __init__(self):
        self.result = 0

    def add(self, num):
        self.result += num
        return self.result

cal1 = Calculator()
cal2 = Calculator()

print(cal1.add(3))
print(cal1.add(4))
print(cal2.add(3))
print(cal2.add(7))
```

```
3
7
3
10
```

[클래스]

4. 생성자(Constructor) __init__(self, ...) - 클래스 내에서 사용할 변수들의 초기값 설정

5. 덧셈, 뺄셈, 나눗셈, 곱셈 하는 계산기

```
class calculator:
  def __init__(self, first, second):
    self.first = first
    self.second = second

  def add(self):
    result = self.first + self.second
    return result

  def sub(self):
    result = self.first - self.second
    return result

  def mul(self):
    result = self.first * self.second
    return result
```

생성자

```
  def div(self):
    result = self.first / self.second
    return result


f = int(input("first number: "))
s = int(input("second number: "))
a = calculator(f,s)

print(a.add())
print(a.sub())
print(a.mul())
print(a.div())
```

```
first number: 3
second number: 4
7
-1
12
0.75
```