#### **Python** ―――

긴 프로그램은 다음과 같이 세 부분으로 나누어 작성한다.

**## 함수 선언 부분 ##**

프로그램에서 사용될 함수들을 만들어둠 (global 사용할 전역 변수 / 내가 만든 것)

파이썬은 변수를 선언하지 않고 바로 사용해도 된다. 그러나 가급적 선언하는게 좋다!

**## 변수 선언 부분 ##**

프로그램 전체에서 사용될 전역 변수를 미리 선언

**## 메인(Main) 코드 부분 ##**

프로그램이 실제로 처리되는 주요한 부분

#### 

#### **용어편** 기본값

| **특수코드값** | **설명** | **비고** |
| --- | --- | --- |
| # | 주석 | 한 줄 주석 |
| '''  ''' | 작은 따옴표 3개  주석 | 여러줄 주석 |
| \ | 줄을 바꾸어 써도 한줄로 인식  오른쪽 예문의 경우 아래처럼 출력된다.  **안녕하세요?** | data= "안녕" + \  "하세요?"  print(data) |
| \n | 출력(print)시 새로운 줄로 이동 | enter를 누른 효과 |
| \t | 출력(print)시 다음 탭으로 이동 | tab 누른 효과 |
| \b | 출력(print)시 뒤로 한 칸 이동 | backspace 누른 효과 |
| \\  \'  \\* | 출력(print)시 \ 출력  출력(print)시 ' 출력  출력(print)시 \* 출력 |  |
| "100"  "100+100" | 100  100+100 | 문자, 일영영 이다.  큰 따옴표 안에 있는 건  문자로 취급. |
| %d | 정수  %5d 오른쪽에 붙여서 5자리 수 정렬  %05d 오른쪽에 붙여서 정렬+빈칸 0으로  print("%5d" % 12345)  print("%5d" % 123)  print("%05d" % 123) | print("100") > 문자 **100**  print("%d" % 100) > 정수(숫자) **100**  print("%d" % 100.99) > 정수(숫자) **100**  **12345**  **123**  **00123** |
| %f  %a.bf | 실수 (소수점까지) \_ 기본 6자리까지 지원  a는 총 자리 맞추기 (소수/온점포함계산)  b는 소수 자리 맞추기 (반올림)  >배열 느낌으로 넘는 정수는 그냥 표기  print("%8.1f" % 12.365)  print("%8.5f" % 1.2345)  print("%8.0f" % 1234.5) | print("100") > 문자 **100**  print("%f" % 100.99) > 실수(숫자) **100.990000**  print("%2.1f" % 10.99) > 실수(숫자) **10.9**  print("%4.2f" % 10.99) > 실수(숫자) **10.99**  **12.4**  **1.23450**  **1234** |
| %s | "안녕하세요" | print("100") > 문자 **100**  print("%s" %100") > 문자 **100** |

#### **산술 연산자**

| **특수코드값** | **설명** | **예시** |
| --- | --- | --- |
| + | 더하기 | a = 3+5  5와 3을 더한 값을 a에 대입한다. |
| - | 빼기 | a = 5-3  5에서 3을 뺀 값을 a에 대입한다. |
| \* | 곱하기 | a = 5 \* 3  5와 3을 곱한 값을 a에 대입한다. |
| \*\* | 제곱 | a = 5\*\*3  5의 3제곱을 a에 대입한다. |
| % | 나머지 값 | a = 5%3  5를 3으로 나눈 후 나머지 값을 a에 대입한다.  a = 2 |
| // | 나눈 후 소수점을 버림 (몫) | a = 5 //3  5를 3으로 나눈 후 소수점을 버리고 값을 a에 대입한다.  a = 1 |
| / | 나누기 | a = 5 /3  5를 3으로 나눈 값을 a에 대입한다.  a = 1.66666666666666667 |
| = | 대입 연산자  (+=, -=, \*=, /= 등등도 있다)  파이썬에는 증가 연산자(++) 및  감소 연산자(--)가 없다. | a = 3  정수 3을 a에 대입한다. |
| += | a = a + 3 과 동일 | a += 3 |
| -= | a = a - 3 과 동일 | a -= 3 |
| \*= | a = a \* 3 과 동일 | a \*= 3 |
| /= | a = a / 3 과 동일 | a /= 3 |
| //= | a = a // 3 과 동일 | a //= 3 |
| %= | a = a % 3 과 동일 | a %= 3 |
| \*\*= | a = a \*\*3 과 동일 | a \*\*= 3 |

#### **관계 연산자**

제어문에서 주로 사용됨. 단독으로는 거의 사용하지 않는다.

| 어떤 것이 크거나 작거나 같은지 비교하는 것. (참은 True 거짓은 False로 표시)  주로 조건문(if)나 반복문(while)에서 사용. |
| --- |

| **특수코드값** | **설명** | **예시** |
| --- | --- | --- |
| == | 같다 | 두 값이 동일하면 참 |
| != | 같지 않다. | 두 값이 다르면 참 |
| > | 크다 | 왼쪽이 크면 참 |
| < | 작다 | 왼쪽이 작으면 참 |
| >= | 크거나 같다 | 왼쪽이 크거나 같으면 참 |
| <= | 작거나 같다 | 왼쪽이 작거나 같으면 참 |

#### **논리 연산자**

and, or, not 세 가지 종류

| **특수코드값** | **설명** | **예시** |
| --- | --- | --- |
| and | ~이고, 그리고  둘 다 참이어야 참 | a = 99  (a>100) and (a<200  False |
| or | ~이거나, 또는  둘 중 하나만 참이어도 참 | a = 99  (a > 100) or (a < 200)  True |
| not | ~아니다, 부정  참이면 거짓, 거짓이면 참 | a= 99  not(a == 100)  True |

#### **함수** 추가값

| **특수코드값** | **비고** |
| --- | --- |
| 함수 | def 함수명(매개변수):  전역변수  return 반환 |
| print("안녕하세요?") | **안녕하세요?**  가 출력된다. 출력 함수라고 생각하면 될 것 같다. |
| format() | print("%d %5d %05d" % (1,2,3))  ==  print("{0:%d} {1:%5d} {2:%05d}" .format (1,2,3))  ==  print("{0:%d} {2:%05d} {1:%5d}" .format (1,3,2)) |
| str() | 숫자를 문자로 전환 |
| input() | 키보드로 치는 값을 입력받음 |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

#### **변수** 의 선언

| ‘정수, 실수, 불린(boolean), 문자(열)’ 4 종류를 담을 수 있다.  boolVar = True  intVar = 0  floatVar = 0.0  strVar = ""  위를 아래와 같이 표기할 수 있다. (근데 좀 헷갈려서 난 별로인 것 같아……..)  boolVar, intVar, floatVar, strVar = True, 0, 0.0, "" |
| --- |

| 대 소문자를 구분한다. (myVar과 MyVar는 다른 변수)  문자, 숫자, 언더바를 포함할 수 있으나 숫자로 시작하면 안 된다.  변수는 값을 담으면 덮어쓰기 된다. |
| --- |

| **특수코드값** | **비고** |
| --- | --- |
| 불형 (boolean) | True 혹은 False  두 가지 답만 담을 수 있는 변수 #제어문과 밀접한 관계래! |
| var1 = var2 = var3 = var4 = 100 | var4 = 100  var3 = var4  var2 = var3  var1 = var2 |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

#### **데이터** 의 표현

2진수 > 10진수/16진수 or 10진수/16진수 > 2진수

| **비트(Bit)** : 컴퓨터에서 표현할 수 있는 제일 작은 단위 / 0과 1만 존재  **바이트(Byre)** : 비트가 8개 모였다! |
| --- |

| **특수코드값** | **숫자형** | **비고** |
| --- | --- | --- |
| bin() | 0b | 2진수 (0, 1) |
| oct() | 0o | 10진수 (0~9) |
| hex | 0x | 16진수 (0~F) |
| a= 0xFF  b= 0o77  c= 0b1111  print(a, b, c) | | 255 63 15  a = 255  b = 63  c =15 |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

#### **조건문** ―――

#### **if** 문

| 파이썬은 들여쓰기가 매우 중요하다. if 문 다음에 '실행할 문장'은 if 문 다음 줄에서 들여쓰기를 해서 작성한다. 실행할 문장이 모두 끝나고 Enter 2번 눌러야 if문이 끝나는 것으로 간주한다. |
| --- |

| **특수코드값** | **형태** | **비고** |
| --- | --- | --- |
| if문 | if a < 100 :  print("100보다 작군요.")  print("거짓이므로 이 문장은 안 보이겠죠?") | 들여쓰기 오류의 예다  세번째 문장까지 출력 됨.  if문은 두번째줄까지만 포함. |
| if~else | if 조건식 :  실행할 문장 1 (True)  else:  실행할 문장2 (False) | |
| if~else  ~if~else | if 조건식 1 :  if 조건식 2 :  실행할 문장1  else :  실행할 문장2  else :  실행할 문장3 | 중첩 if 문이다  if문을 한 번 실행한 후  그 결과에서 if문을 다시 실행  elif 사용해서 중첩가능 |

데이터를 여러개 한 곳에 담아놓는 리스트와 함께 사용도 가능하다.

리스트: 대괄호 [] 로 묶고 그 안에 필요한 것들을 한번에 넣음

| **특수코드값** | **형태** | **출력값** |
| --- | --- | --- |
| List | fruit = ["사과", "배", "딸기", "포도"]  print(fruit) | ["사과", "배", "딸기", "포도"] |
| List 추가속성  append | fruit.append("귤")  print(fruit) | ["사과", "배", "딸기", "포도", "귤"] |
| if 항목  in 리스트  리스트에 항목이 있다면 True를 반환 | if "딸기" in fruit :  print ("딸기가 있네요!") | 딸기가 있네요! |

#### **for** 문

| for 변수 in range (시작값, 끝값+1, 증가값):  이 부분을 반복 |
| --- |

| **특수코드값** | **형태** | **비고** |
| --- | --- | --- |
| for문 | for i in range(0, 3, 1):  print("%d: 안녕하세요?" % i) | 0: 안녕하세요?  1: 안녕하세요?  2: 안녕하세요? |
| hap = 0  i = 0  for i in range(1, 11, 1):  hap = hap+1  print ("1에서 10까지의 합계: %d" % hap ) | 변수 hap이 계속 누적.  시작할 때 i 와 hap 초기화 잊지말기 |

| **특수코드값** | **형태** | **출력값** |
| --- | --- | --- |
| 중첩 for문 | ##전역변수선언##  i = 0  k = 0  guguLine = ""  ##메인코드##  for i in range (2, 10):  guguLine = guguLine + ("# %d 단 #" % i)  print(guguLine)  for i in range(1,10):  guguLine = ""  for k in range(2, 10):  guguLine = guguLine + str("%2d X %2d = %2d"%(k, i, k \* i))  print(guguLine) | 구구단 출력됨  2단부터  9단까지  2단의  9까지 돌고  3단의  9까지 돌고  방식으로  중첩해서  적용됨 |

#### **While** 문

| 조건이 참일때 반복한다.  ※ for 문은 while문을 대신할 수 없다  ※ while문은 for문을 대신할 수 있다  변수 = 시작값  while 변수 < 끝값:  이 부분을 반복 |
| --- |

| **특수코드값** | **형태** | **비고** |
| --- | --- | --- |
| while문 | i = 0  hap = 0  i = 1  while i < 11 :  hap = hap+i  i = i+1 #이게 for 문의 증가값과 동일  print ("1에서 10까지의 합계 : %d" % hap) | i 가 11보다 작은 게 참일때  아래의 내용을 반복한다 |
| while문  무한반복 | hap = 0  a, b = 0 ,0  while True :  a= int(input("첫번째 수를 입력해주세요: "))  b= int(input("두번째 수를 입력해주세요: "))  hap = a+b  print("%d + %d = %d" % (a,b,hap)) | 조건식이 True일 경우  무조건 참이기 때문에  아래의 내용을 계속 반복한다 |

#### **Break** 문

| 반복문을 강제적으로 종료시켜준다.  for/while문과 함께 사용된다. (if문에는 break가 없다) |
| --- |

| **특수코드값** | **형태** | **비고** |
| --- | --- | --- |
| while문  +  break | hap = 0  a, b = 0 ,0  while True :  a= int(input("더할 첫번째 수를 입력해주세요: "))  if a == 0:  break  b= int(input("더할 두번째 수를 입력해주세요: "))  hap = a+b  print("%d + %d = %d" % (a,b,hap))  print("0을 입력해 반복문을 탈출했습니다.") | 0을 입력할 경우  두번째 수 입력문항은  실행되지 않고  바로 종료된다 |
| hap, i = 0, 0  for i in range(1,101):  hap += i  if hap >= 1000 :  break  print ("1~100의 합계를 1000이 넘게 하는 숫자: %d" %i) | 원래는 100번  반복해야지만,  if문에서  break에 걸렸으므로  print를 출력하고  종료된다 |

#### **Continue** 문

| continue 뒤에 있는 내용은 건너뛰고  반복문(for while) 으로 다시 돌아가게 한다. |
| --- |

| **특수코드값** | **형태** | **비고** |
| --- | --- | --- |
| while문  +  continue | hap, i = 0, 0  for i in range (1, 101):  if i % 3 == 0 :  continue  hap += i  print (" 1~100의 합계 (3의 배수 제외): %d" %hap) | i가 3으로 나눈 나머지가  0일 경우  (= 3의 배수일경우)  이전의 합계(=hap)에  합해지지 않고  다음 숫자로 넘어간다. |

#### **리스트, 튜플, 딕셔너리** ―――

#### **리스트**

정수, 문자열, 실수 등 서로 다른 데이터 형을 하나로 묶는다. for과 친함

| C/C++나 자바 같은 프로그래밍 언어에는 리스트가 없다. 리스트와 비슷한 개념인 배열(Arry)을 사용. 단, 배열은 동일한 데이터형만 묶을 수 있다. |
| --- |

| **특수코드값** | **설명** | **예시** |
| --- | --- | --- |
| == | 같다 | 두 값이 동일하면 참 |
| != | 같지 않다. | 두 값이 다르면 참 |
| > | 크다 | 왼쪽이 크면 참 |
| < | 작다 | 왼쪽이 작으면 참 |
| >= | 크거나 같다 | 왼쪽이 크거나 같으면 참 |
| <= | 작거나 같다 | 왼쪽이 작거나 같으면 참 |