1. 3

2. 2

3.

*def* returnLenStr(inputStr):  
 *return* len(inputStr)  
  
*def* returnLenEmptyStr(inputStr):  
 *return* inputStr.count(' ')  
  
*def* removeBlank(inputStr):  
 *return* inputStr.lstrip()

4. as c

5.

*def* calculateDiscountRate(qtyGoods):  
 *if* qtyGoods == 1:  
 discountRate = 5  
 *elif* qtyGoods == 2:  
 discountRate = 10  
 *elif* qtyGoods == 3:  
 discountRate = 20  
 *elif* qtyGoods >= 4:  
 discountRate = 30  
 *return* discountRate

6.

*import* math  
  
print(math.gcd(15, 21))

7.

*import* math  
  
inputInt = int(input('팩토리얼을 계산할 정수 입력. '))  
print(math.factorial(inputInt))

8.

*import* math  
  
print(round(math.sqrt(int(input('제곱근을 구할 정수입력 '))), 1))

9.

*import* random  
  
print(random.randint(1, 6))

10.

*import* random  
  
print(random.sample(range(1, 101), 5))

11.

*import* time  
  
print(f'현재 시간 : {time.strftime("%Y-%m-%d %H:%M:%S", time.localtime())}')

12. Y, m, d, H, M, S

13. 1

14. 3

15. car1 = Car(‘blue’, 500), car2 = Car(‘black’, 450)

16. 17.

*class* Bank:  
 *def \_\_init\_\_*(*self*):  
 *self*.name = ""  
 *self*.money = 0  
 *self*.history = ""  
  
 *def* deposit(*self*, dm, dh):  
 *self*.name = 'deposit'  
 *self*.money += dm  
 *self*.history = dh  
  
 *def* withdraw(*self*, wm, wh):  
 *self*.name = 'withdraw'  
 *self*.money -= wm  
 *self*.history = wh  
  
 *def* printBalance(*self*):  
 *return self*.money  
  
 *def* printHistory(*self*):  
 *return self*.history  
  
*def* app():  
 bank = Bank()  
 *while True*:  
 selectInput = input('1. 입금 2. 출금 3. 종료')  
 *if* selectInput == '1':  
 depositMoney = int(input('입금액 : '))  
 depositHistory = input('입금내역 : ')  
 print('입금 완료')  
 print(f'입금전 계좌잔액 : {bank.printBalance()}')  
 bank.deposit(depositMoney, depositHistory)  
 print(f'입금액 : {depositMoney} 입금내역 : {bank.history}')  
 print(f'현재 계좌잔액 : {bank.printBalance()}')  
 *elif* selectInput == '2':  
 withdrawMoney = int(input('출금액 : '))  
 withdrawHistory = input('출금내역 : ')  
 print('출금 완료')  
 print(f'출금전 계좌잔액 : {bank.printBalance()}')  
 bank.withdraw(withdrawMoney, withdrawHistory)  
 print(f'출금액 : {withdrawMoney} 출금내역 : {bank.history}')  
 print(f'현재 계좌잔액 : {bank.printBalance()}')  
 print()  
 *elif* selectInput == '3':  
 *break  
  
if* \_\_name\_\_ == '\_\_main\_\_':  
 app()

18.

*class* Student:  
 *def \_\_init\_\_*(*self*, n, m, a, g):  
 *self*.name = n  
 *self*.major = m  
 *self*.age = a  
 *self*.gender = g  
  
*class* StudentManageSystem:  
 *def \_\_init\_\_*(*self*):  
 *self*.students = {}  
  
 *def* addStudent(*self*, s):  
 *self*.students[s.name] = [s.major, s.age, s.gender]  
  
 *def* removeStudent(*self*, n):  
 *del self*.students[n]  
  
 *def* printStudentInfo(*self*, n):  
 print(f'전공 : {*self*.students[n][0]}')  
 print(f'나이 : {*self*.students[n][1]}')  
 print(f'성별 : {*self*.students[n][2]}')  
  
 *def* printStudentsInfo(*self*):  
 *for* student *in self*.students.keys():  
 print(f'이름 : {student}, 전공 : {*self*.students[student][0]}, 나이 : {*self*.students[student][1]}, 성별 : {*self*.students[student][2]}')  
  
*def* app():  
 sms = StudentManageSystem()  
 *while True*:  
 selectInput = input('1. 학생정보등록 2. 학생정보수정 3. 학생정보삭제 4. 개별학생정보 5. 전체학생정보 6. 종료 ')  
 *if* selectInput == '1':  
 studentName = input('이름 : ')  
 *while* studentName *in* sms.students.keys():  
 print('중복되는 이름입니다.')  
 studentName = input('이름 : ')  
 studentMajor = input('전공 : ')  
 studentAge = input('나이 : ')  
 studentGender = input('성별 : ')  
 student = Student(studentName, studentMajor, studentAge, studentGender)  
 sms.addStudent(student)  
 print(f'{studentName} 학생 등록 완료')  
 *elif* selectInput == '2':  
 studentName = input('이름 : ')  
 *while* studentName *not in* sms.students.keys():  
 print('해당이름의 학생이 없습니다.')  
 studentName = input('이름 : ')  
 studentMajor = input('전공 : ')  
 studentAge = input('나이 : ')  
 studentGender = input('성별 : ')  
 student = Student(studentName, studentMajor, studentAge, studentGender)  
 sms.addStudent(student)  
 print(f'{studentName} 학생 등록 완료')  
 *elif* selectInput == '3':  
 studentName = input('이름 : ')  
 *while* studentName *not in* sms.students.keys():  
 print('해당이름의 학생이 없습니다.')  
 studentName = input('이름 : ')  
 sms.removeStudent(studentName)  
 *elif* selectInput == '4':  
 studentName = input('이름 : ')  
 *while* studentName *not in* sms.students.keys():  
 print('해당이름의 학생이 없습니다.')  
 studentName = input('이름 : ')  
 sms.printStudentInfo(studentName)  
 *elif* selectInput == '5':  
 sms.printStudentsInfo()  
 *elif* selectInput == '6':  
 *break  
  
if* \_\_name\_\_ == '\_\_main\_\_':  
 app()

19. 2

20.

*import* time  
  
*while True*:  
 selectInput = input('다음 항목을 선택하세요.\n1. 일기작성 2. 일기보기 3. 종료 ')  
 *if* selectInput == '1':  
 print(f'{time.strftime("[%Y-%m-%d]", time.localtime())} 한줄 일기를 작성하세요.')  
 *with* open('./diary.txt', 'a', encoding='UTF-8') *as* f:  
 f.write(f'[{time.strftime("[%Y-%m-%d %H:%M:%S] ", time.localtime())}]' + input() + '\n')  
 *elif* selectInput == '2':  
 *with* open('./diary.txt', 'r', encoding='UTF-8') *as* f:  
 *for* line *in* f.readlines():  
 print(line, end="")  
 *elif* selectInput == '3':  
 *break*

21.

inputStr = input('암호화할 문자열을 입력하세요.\n')  
encryptionStr = ''  
i = 1  
*for* c *in* inputStr:  
 *if* i % 6 == 1:  
 encryptionStr += c + 'a'  
 *elif* i % 6 == 3:  
 encryptionStr += c + 'pp'  
 *elif* i % 6 == 0:  
 encryptionStr += c + 'le.'  
 *else*:  
 encryptionStr += c  
 i += 1  
print(f'암호화 문자열 : {encryptionStr}')  
  
inputStr = input('해독할 암호화 문자열을 입력하세요.\n')  
decodingStr = ''  
i = 1  
*for* c *in* inputStr:  
 *if* (i % 12 != 2) *and* (i % 12 != 5) *and* (i % 12 != 6) *and* (i % 12 != 10) *and* (i % 12 != 11) *and* (i % 12 != 0):  
 decodingStr += c  
 i += 1  
print(f'해독한 문자열 : {decodingStr}')

22.

*from* bs4 *import* BeautifulSoup  
*import* urllib.request  
*from* openpyxl *import* Workbook  
*import* time  
  
*def* getStockData(kospiDatas, kosdaqDatas):  
 urlList = [['kospi', 'https://finance.naver.com/sise/sise\_quant.naver?sosok=0'], ['kosdaq', 'https://finance.naver.com/sise/sise\_quant.naver?sosok=1']]  
 *for* url *in* urlList:  
 responseObj = urllib.request.urlopen(url[1])  
 parsingData = BeautifulSoup(responseObj, 'html.parser', from\_encoding='cp949')  
 tag\_table = parsingData.find\_all('table')[1]  
 tag\_trs = tag\_table.find\_all('tr')  
 i = 1  
 *for* tr *in* tag\_trs[2:len(tag\_trs) - 3]:  
 *if* i % 8 != 0 *and* i % 8 != 6 *and* i % 8 != 7:  
 tag\_tds = tr.find\_all('td')  
 itemName = tag\_tds[1].text.replace('\t', '').replace('\n', '')  
 currentPrice = tag\_tds[2].text.replace('\t', '').replace('\n', '')  
 *if* (tag\_tds[3].find('img') *is not None*) *and* ('alt' *in* tag\_tds[3].find('img')):  
 dayToDay = tag\_tds[3].find('img')['alt'] + tag\_tds[3].text.replace('\t', '').replace('\n', '')  
 *else*:  
 dayToDay = tag\_tds[3].text.replace('\t', '').replace('\n', '')  
 rate = tag\_tds[4].text.replace('\t', '').replace('\n', '')  
 volume = tag\_tds[5].text.replace('\t', '').replace('\n', '')  
 amount = tag\_tds[6].text.replace('\t', '').replace('\n', '')  
 bid = tag\_tds[7].text.replace('\t', '').replace('\n', '')  
 ask = tag\_tds[8].text.replace('\t', '').replace('\n', '')  
 marketCap = tag\_tds[9].text.replace('\t', '').replace('\n', '')  
 per = tag\_tds[10].text.replace('\t', '').replace('\n', '')  
 roe = tag\_tds[11].text.replace('\t', '').replace('\n', '')  
 *if* url[0] == 'kospi':  
 kospiDatas.append({'itemName':itemName,  
 'currentPrice':currentPrice,  
 'dayToDay':dayToDay,  
 'rate':rate,  
 'volume':volume,  
 'amount':amount,  
 'bid':bid,  
 'ask':ask,  
 'marketCap':marketCap,  
 'per':per,  
 'roe':roe})  
 *elif* url[0] == 'kosdaq':  
 kosdaqDatas.append({'itemName': itemName,  
 'currentPrice': currentPrice,  
 'dayToDay': dayToDay,  
 'rate': rate,  
 'volume': volume,  
 'amount': amount,  
 'bid': bid,  
 'ask': ask,  
 'marketCap': marketCap,  
 'per': per,  
 'roe': roe})  
 i += 1  
  
*def* app():  
 kospiDatas = []  
 kosdaqDatas = []  
 getStockData(kospiDatas, kosdaqDatas)  
  
 write\_wb = Workbook()  
 write\_ws\_kospi = write\_wb.create\_sheet('kospi')  
 write\_ws\_kosdaq = write\_wb.create\_sheet('kosdaq')  
 write\_ws\_kospi.append(['번호', '종목명', '현재가', '전일비', '등락률', '거래량', '거래대금', '매수호가', '매도호가', '시가총액', 'PER', 'ROE'])  
 write\_ws\_kosdaq.append(['번호', '종목명', '현재가', '전일비', '등락률', '거래량', '거래대금', '매수호가', '매도호가', '시가총액', 'PER', 'ROE'])  
  
 *for* idx, data *in* enumerate(kospiDatas):  
 write\_ws\_kospi.append([idx + 1, data['itemName'], data['currentPrice'], data['dayToDay'], data['rate'], data['volume'], data['amount'], data['bid'], data['ask'], data['marketCap'], data['per'], data['roe']])  
 *for* idx, data *in* enumerate(kosdaqDatas):  
 write\_ws\_kosdaq.append([idx + 1, data['itemName'], data['currentPrice'], data['dayToDay'], data['rate'], data['volume'], data['amount'], data['bid'], data['ask'], data['marketCap'], data['per'], data['roe']])  
 write\_wb.save(f'./{time.strftime("%Y-%m-%d\_%H%M%S", time.localtime())}\_stock\_data.xlsx')  
  
*if* \_\_name\_\_ == '\_\_main\_\_':  
 app()