1	`15`	
2	`25`	(** raises to a power, so 3**2 is 9 and 4**2 is 16.)
3	`21`	
4	Nothing	(Notice that this is a comment.)
5	`1.0`	
6	`0`	(// is integer division, so 1//2 is 0.)
7	`1`	(% is modulus, or the remainder when divided by the number. So, odd numbers %2 will be 1, while even numbers %2 will be 0.)
8	`print (12*8)`	
9	`print(180/7)`	
10	`print("I love Python")`	
11	`#This is my first program`	

REMEMBER

There is more than one way to write code correctly.

```
25

15

115

250

10.0

10

Welcome Home                    (The comma causes a space to be printed.)

WelcomeHome                     (Concatenation eliminates the space.)

1015                            (The + indicates concatenation. The
                                conversion to an integer comes
                                after the concatenation.)
bread_price = 2.0

total_price = 2*bread_price + 3*cheese_price

age = int(input("What is your age?"))    (Remember: Convert
                                a number that's input
                                to an integer.)

knight_name = input("What is the knight's name?")
knight_trait = input("What is a characteristic of the knight?")
print("Sir "+knight_name+" the "+knight_trait)
```

```
1  False

2  False                                  (== tests for equality.)

3  True                                   (!= tests for inequality.)

4  True

5  True                                   (Strings are compared letter by letter.)

6  False

7  True                                   (Capital letters always come before
                                           lowercase letters, so "O" < "o.")

8  False                                  (All capital letters come before
                                           lowercase letters, so "o" > "T.")

9  True

10 False

11 True

12 True

13 Reasonable cost

14 Eligible for reduced benefits
```

15
```
age = 67
income = 10000
if (income < 15000) and (age >= 70):
    print("Eligible for benefits")
elif (income < 20000):
    print("Eligible for reduced benefits")
else:
    print("Not eligible for benefits")
```

16
```
if user_guess < hidden_answer:
    print("Too low")
elif user_guess > hidden_answer:
    print("Too high")
else:
    print("You guessed it!")
```

17 a)
```
if year % 4 == 0:
      if year % 100 == 0:
          if year % 400 == 0:
              print("Leap year")
          else:
              print("Not a leap year")
      else:
          print("Leap year")
  else:
      print("Not a leap year")
```

b)
```
if year % 400 == 0:
      print("Leap year")
  elif year % 100 == 0:
      print("Not a leap year")
  elif year % 4 == 0:
      print("Leap year")
  else:
      print("Not a leap year")
```

c)
```
if (year%4 != 0) or ((year%4 == 0) and (year%100 == 0)
  and (year%400 != 0)):
      print("Not a leap year")
  else:
      print("Leap year")
```

The following is one example of the modified code.

For the first part, notice that we added a variable, “cost,” and then created another variable, “saved,” which we read in from the user. We then compute “balance” as the difference.

For the second part, notice that we have a new input line that gets the period being saved for and that this variable is used in the later input and output statements.

```
#Get information from user
print("I'll help you determine how long you will need to save.")
name = input("What's your name? ")
item = input("What is it you are saving up for? ")
cost = float(input("OK, "+name+".  Please enter the cost of the
  "+item+": "))
saved = float(input("How much have you already saved? "))
balance = cost-saved
period = input("How often will you save (day, week, month)? ")
if (balance<0):
    print("Looks like you already saved enough!")
    balance = 0
    payment = 1
else:
    payment = float(input("Enter how much you will save each
      "+period+": "))
```

```
    if (payment <= 0):
        payment = float(input("Savings must be positive.  Please
          enter a positive value:"))
        if (payment <=0):
            print(name+", you still didn't enter a positive
              number!  I am going to just assume you save 1 per
              "+period+".")
            payment = 1
#Calculate number of payments that will be needed
num_remaining_payments = int(balance/payment)
if (num_remaining_payments < balance/payment):
    num_remaining_payments = num_remaining_payments + 1
#Present information to user
print(name+", if you save", payment, "each "+period+", you must
  make", num_remaining_payments, "more payments, and then you'll
  have your "+item+"!")
```

1
```
10
5.0
2.5
1.25
```

2
```
0
1
3
6
10
15
21
```

(Notice that the value can exceed 20 inside the loop, and the check to see if it is less than 20 is only after the loop body is completed.)

3
```
0
1
2
3
```
(Remember that the range starts at 0, and only continues while less than the number given in parentheses.)

4
```
3
4
```
(The range starts at 3 and continues while *i* is less than 5.)

5
```
1
4
7
```

6 Nothing printed

(Notice that the increment is negative, so we are counting down. The starting value, 1, is less than the minimum, 10.)

7
```
10
7
4
```

8 The following are two versions: one with a while loop and one with a for loop. Notice that you need to start at 1, not 0, and include the final number in the list, either by using "<=," as in the while loop, or "num+1," as in the for loop.

```
num = int(input("Enter a number to count to: "))
i = 1
while (i<=num):
    print(i)
    i += 1
```

```
num = int(input("Enter a number to count to: "))
for i in range(1,num+1):
    print(i)
```

9
```
for i in range(2,7,3):
    print(i)
```

10 The following are two possible versions.

```
secret_number = 7
while True:
    guess = int(input("Enter your guess from 1 to 10: "))
    if guess == secret_number:
        break
    else:
        print("No!  Try again.")
print("You guessed it!")
```

```
secret_number = 7
guess = 0
while guess != secret_number:
    guess = int(input("Enter your guess from 1 to 10: "))
    if guess != secret_number:
        print("No!  Try again.")
print("You guessed it!")
```

1
```
infile = open("data.txt", 'r')
```

2
```
outfile = open("../data.txt", 'w')
```

3
```
infile.close()
outfile.close()
```

4
```
for l in infile.readlines():
    print(l ,end='')
```

(Note: By adding the ",end=''" to the print statement, we eliminate the final newline that is printed at the end of each print statement. This was not asked for in the problem but can be a useful technique.)

5
```
filename = input("What file should we write to? ")
outfile = open(filename, 'w')
for i in range(1,11):
    outfile.write(str(i)+'\n')
outfile.close()
```

6
```
infile = open("data.txt", 'r')
i = 0
sum = 0
for l in infile.readlines():
    num = int(l)
    sum += num
    i+=1
average = (sum)/i
infile.close()
print(average)
```

1
```
4
```

2
```
[6, 8, 10]
```

3
```
[2, 4, 6]
```

4
```
[18, 20]
```

5
```
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20]
```

6
```
20
```

7
```
[16, 18, 20]
```

8
```
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20]
```

9
```
2
4
6
8
10
12
14
16
18
20
```

10
```
[2, 4, 6, 100, 10, 12, 14, 16, 18, 20]
```

11
```
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20]
```

12
```
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 100]
```

13
```
[2, 12, 14, 16, 18, 20]
```

14
```
[2, 4, 100, 200, 18, 20]
```

15
```
[2, 4, 100, 6, 8, 10, 12, 14, 16, 18, 20]
```

16
```
minor_ages = []
for age in ages:
    if age < 18:
        minor_ages.append(age)
```

17
```
mylist = [["John", "James", "Joel"], [25, 28, 30]]
```

The following are changes to the relevant parts of the program. The new lines are in bold.

```
# Perform analysis
minsofar = 120
maxsofar = -100
numgooddates = 0
sumofmin=0
sumofmax=0
raindays = 0
for singleday in gooddata:
    numgooddates += 1
    sumofmin += singleday[1]
    sumofmax += singleday[2]
    if singleday[1] < minsofar:
        minsofar = singleday[1]
    if singleday[2] > maxsofar:
        maxsofar = singleday[2]
    if singleday[3] > 0:
        raindays += 1
avglow = sumofmin / numgooddates
avghigh = sumofmax / numgooddates
rainpercent = raindays / numgooddates * 100
########## Present Results ##########
print("There were", numgooddates,"days")
print("The lowest temperature on record was", minsofar)
print("The highest temperature on record was", maxsofar)
print("The average low has been", avglow)
print("The average high has been", avghigh)
print("The chance of rain is", rainpercent, "%")
```

```
XXXXX
```

```
256
```

```
4 6
```

```
21
```

```
9 12
```

```
def print_string(numtimes, str):
    for i in range(numtimes):
        print(str)
```

```
def small_list(listA, listB):
    newlist = []
    for i in range(len(listA)):
        if listA[i] < listB[i]:
            newlist.append(listA[i])
        else:
            newlist.append(listB[i])
    return newlist
```

```
def middle(a, b, c):
    if ((a >= b) and (b >= c)) or ((a <= b) and (b <= c)):
        return b
    elif ((a >= c) and (c >= b)) or ((a <= c) and (c <= b)):
        return c
    else:
        return a
```

```
def findincrease(val1, val2):
    if val2 > val1:
        return val2 - val1
    else:
        return 0
salary1 = float(input("Enter previous salary"))
benefits1 = float(input("Enter previous benefits"))
bonus1 = float(input("Enter previous bonus"))
salary2 = float(input("Enter new salary"))
benefits2 = float(input("Enter new benefits"))
bonus2 = float(input("Enter new bonus"))
salaryincrease = findincrease(salary1, salary2)
benefitsincrease = findincrease(benefits1, benefits2)
bonusincrease = findincrease(bonus1, bonus2)
```

```
3
[0, 2, 3]
21
2
3
[0, 2, 3]
2 1
def increment_list(a):
    for i in range(len(a)):
        a[i] += 1
```

9
```
def multiply4(a, b=1, c=1, d=1):
    return a*b*c*d
```

---

1 The four cases on either side of a “boundary”: 2, 3, 12, 13.

Some “middle” values, such as 1, 7, 25.

“Extreme” cases, such as 0 or 100.

2 a)
```
def findmiddle(a):
    if len(a) < 3:
        raise TypeError  # This could be a different
          exception
    if ((a[0] >= a[1]) and (a[1] >= a[2])) or ((a[0] <=
      a[1]) and (a[1] <= a[2])):
        return a[1]
    elif ((a[0] >= a[2]) and (a[2] >= a[1])) or ((a[0] <=
      a[2]) and (a[2] <= a[1])):
        return a[2]
    else:
        return a[0]
```

b)
```
try:
      middle = findmiddle(a)   #a is some
  except TypeError:
      print("Problem: You need a list of at least length
        3!")
```

3

```
Michael Palin
Michael Palin
Terry Gilliam
```

4

```
{2, 3, 37, 5, 7, 11, 23, 29, 31}
{17, 19, 13}
{2, 3, 37, 5, 7, 11, 13, 14, 15,
  16, 17, 18, 19, 23, 29, 31}
{2, 3, 5, 7, 11, 14, 15, 16, 18, 23, 29, 31, 37}
```

(Note: The order of elements in a set does not matter.)