Pythonでいかにして暗号を破るか

古典暗号解読プログラムを自作する本

【練習問題と解答】

翻訳・注釈：IPUSIRON
編集・制作：ソシム株式会社
最終更新日：2020年12月10日

原著の解答はhttps://nostarch.com/crackingcodesanswersにあります。

第1章　紙製の暗号ツールをつくる（P.036）問題

問1　次に示す項目は、Ambrose Bierceの『The Devil's Dictionary』から抜粋したものです。指定された鍵で暗号化してください。

a. AMBIDEXTROUS: Able to pick with equal skill a right-hand pocket or a left. （鍵4）
b. GUILLOTINE: A machine which makes a Frenchman shrug his shoulders with good reason. （鍵17）
c. IMPIETY: Your irreverence toward my deity. （鍵21）

問2　指定された鍵で次の暗号文を復号してください。

a. ZXAI: P rdhijbt hdbtixbth ldgc qn Hrdirwbtc xc Pbtgxrp pcs Pbtgxrpch xc Hrdiapcs. （鍵15）
b. MQTSWXSV: E vmzep ewtmverx xs tyfpmg lsrsvw. （鍵4）

問3　次の文を鍵0で暗号化してください。

This is a silly example.

問4　ここに、いくつかの単語とその暗号文があります。それぞれの単語に対して、どの鍵が用いられましたか？

	単　語	暗号文
a	ROSEBUD	LIMYVOX
b	YAMAMOTO	PRDRDFKF
c	ASTRONOMY	HZAYVUVTF

問5　鍵8で暗号化された次の文を鍵9で復号してください。

UMMSVMAA: Cvkwuuwv xibqmvkm qv xtivvqvo i zmdmvom bpib qa ewzbp epqtm.

第1章　紙製の暗号ツールをつくる（P.036）解答

問1

a. EQFMHIBXVSYW: Efpi xs tmgo amxl iuyep wompp e vmklx-lerh tsgoix sv e pijx.
b. XLZCCFKZEV: R drtyzev nyzty drbvj r Wivetydre jyilx yzj jyflcuvij nzky xffu ivrjfe.
c. DHKDZOT: tjpm dmmzqzmzixz ojrvmy ht yzdot.

問2

a. KILT: A costume sometimes worn by Scotchmen in America and Americans in Scotland.
b. IMPOSTOR: A rival aspirant to public honors.

問3

This is a silly example.

問4

a. 20
b. 17
c. 7

問5

LDDJMDRR: Tmbnllnm ozshdmbd hm okzmmhmf z qdudmfd sgzs hr vnqsg vghkd.
（違った鍵で復号すると、意味のないテキストになります）

参考

オンライン暗号ホイール
https://inventwithpython.com/cipherwheel/

　解答を検証するのに、5章で学習するシーザー暗号プログラム（caserCipher.py）を活用できます。ただし、暗号ホイールとシーザー暗号プログラムにおける仕様の違いを理解しておく必要があります。
相違点は、次の2点です。

①暗号ホイールでは、数字、空白、ピリオドなどを暗号化の対象としていません。そのため、プログラムのSYMBOLSを次のように変更します。

```
SYMBOLS = 'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz'
```

②暗号ホイールでは大文字を暗号化すると大文字、小文字を暗号化すると小文字になります。例えば、文字Zを鍵1で暗号化すると文字Aになります。
　一方、プログラムでは上記のように定義したSYMBOLSにおいて鍵の分だけシフトします。例えば、文字

Zを鍵1で暗号化すると文字aになります。

よって、プログラムの実行結果と平文を見比べて、大文字と小文字が一致するように調整する必要があります。この機能をプログラムに組み込むのであれば、16章のプログラム（simpleSubCipher.py）が実装している大文字・小文字を保持するコードが役立ちます。

第2章　対話型シェルのプログラミング（P.047）問題

問1　除算の演算子は、/と¥（バックスラッシュの代用文字として用います）のどちらですか？

問2　次の値は整数と浮動小数点数のどちらですか？

```
42
3.141592
```

問3　次の行のうち、式ではないものはどれですか？

```
4 x 10 + 2
3 * 7 + 1
2 +
42
2 + 2
spam = 42
```

問4　対話型シェルに次の行のコードを入力すると、行❶と❷では何を表示しますか？

```
spam = 20
❶spam + 20
SPAM = 30
❷spam
```

第2章　対話型シェルのプログラミング（P.047）解答

問1

/が除算の演算子です。バックスラッシュ¥はエスケープ文字のために使用されます。
エスケープ文字については、第3章を参照。

問2　対話型シェルで試してみましょう。Pythonの組み込み関数である、type()関数でも調べられます。

```
>>> type(42)
<class 'int'>
>>> type(3.141592)
<class 'float'>
```

intは整数を、floatは浮動小数点数をそれぞれ意味します。
つまり、42は整数、3.141592は浮動小数点数です。

問3

次の3つが式ではありません。（　）内の記述がその理由です。

```
4 x 10 + 2          (1行目：xは乗算演算子ではない。正しくは*)
2 +                 (3行目：式が不完全)
spam = 42           (6行目：これは代入文であり、式ではない)
```

問4　対話型シェルで試してみましょう。赤い部分が解答です。

```
  >>> spam = 20
❶>>> spam + 20
  40
  >>> SPAM = 30
❷>>> spam
  20
```

行❶では40、行❷では20が表示されます（spamとSPAMは異なる変数です）。

第3章　文字列とプログラムの作成（P.070）問題

問1　spam = 'Cats'のようにしたとき、次の行はどのように表示されますか？

```
spam + spam + spam
spam * 3
```

問2　次の行を実行すると、何が表示されますか？

```
print("Dear Alice,¥nHow are you?¥nSincerely,¥nBob")
print('Hello' + 'Hello')
```

問3　spam = 'Four score and seven years is eighty seven years.'としたとき、次の各行は何を表示しますか？

```
print(spam[5])
print(spam[-3])
print(spam[0:4] + spam[5])
print(spam[-3:-1])
print(spam[:10])
print(spam[-5:])
print(spam[:])
```

問4　画面が>>>プロンプトを表示されているとき、対話型シェルですか？　それともファイルエディターですか？

問5　次の行を実行すると何が表示されますか？

```
#print('Hello, world!')
```

第3章　文字列とプログラムの作成（P.070）解答

問1　対話型シェルで試してみましょう。赤い部分が解答です。

```
>>> spam = 'Cats'
>>> spam + spam + spam
'CatsCatsCats'
>>> spam * 3
'CatsCatsCats'
```

問2　対話型シェルで試してみましょう。赤い部分が解答です。

```
>>> print("Dear Alice,¥nHow are you?¥nSincerely,¥nBob")
Dear Alice,
How are you?
Sincerely,
Bob
>>> print('Hello' + 'Hello')
HelloHello
```

問3　対話型シェルで試してみましょう。赤い部分が解答です。

```
>>> spam = 'Four score and seven years is eighty seven years.'
>>> print(spam[5])
s
>>> print(spam[-3])
r
>>> print(spam[0:4] + spam[5])
Fours
>>> print(spam[-3:-1])
rs
>>> print(spam[:10])
Four score
>>> print(spam[-5:])
ears.
>>> print(spam[:])
Four score and seven years is eighty seven years.
```

問4　「対話型シェル」です。

問5　何も表示されません。対話型シェルで試してみましょう。

```
>>> #print('Hello, world!')
>>>
```

第4章　逆暗号（P.86）問題

問1　次のコードを実行すると画面に何が表示されますか？

```
print(len('Hello') + len('Hello'))
```

問2　このコードを実行すると何が表示されますか？

```
i = 0
while i < 3:
    print('Hello')
    i = i + 1
```

問3　このコードの実行結果はどうなりますか？

```
i = 0
spam = 'Hello'
while i < 5:
    spam = spam + spam[i]
    i = i + 1
print(spam)
```

問4　このコードの実行結果はどうなりますか？

```
i = 0
while i < 4:
    while i < 6:
        i = i + 2
        print(i)
```

第4章　逆暗号（P.86）解答

問1　対話型シェルで試してみましょう。赤い部分が解答です。

```
>>> print(len('Hello') + len('Hello'))
10
```

len()を呼び出すと文字数の整数値が返されます。
ここではそれぞれ5が返って来るので、合計値の10が表示されます。

問2　IDLEでc04-q2.pyを開き、Run→Run Moduleしてみましょう。

```
Hello
Hello
Hello
```

問3　IDLEでc04-q3.pyを開き、Run→Run Moduleしてみましょう。

```
HelloHello
```

問4　IDLEでc04-q4.pyを開き、Run→Run Moduleしてみましょう。

```
2
4
6
```

第5章　シーザー暗号（PP.105-106）問題

問1　caesarCipher.pyを使って、次の文を暗号化してください。ただし、鍵は指定されたものを使用します。

a.
鍵8
'"You can show black is white by argument," said Filby, "but you will never convince me."'

b.
鍵21
'1234567890'

問2　caesarCipher.pyを使って、次の暗号文を復号してください。ただし、鍵は指定されたものを使用します。

a.
鍵2
'Kv?uqwpfu?rncwukdng?gpqwijB'

b.
鍵22
'XCBSw88S18A1S 2SB41SE .8zSEwAS50D5A5x81V'

問3　Pythonの命令を用いて、watermelon.pyというモジュールをインポートするにはどのようにしますか？

問4　次のコードを実行すると画面には何が表示されますか？

a.
```
spam = 'foo'
for i in spam:
    spam = spam + i
print(spam)
```

b.
```
if 10 < 5:
    print('Hello')
elif False:
    print('Alice')
elif 5 != 5:
    print('Bob')
else:
    print('Goodbye')
```

c.

```
print('f' not in 'foo')
```

d.

```
print('foo' in 'f')
```

e.

```
print('hello'.find('oo'))
```

第5章　シーザー暗号（PP.105-106）解答

問1

a. caserCipher.pyをベースにして、messageに平文"You can show black is white by argument," said Filby, "but you will never convince me."（文の最初と最後の二重引用符を含む）、keyに鍵8を指定したプログラムc05-q1-a.pyを用意します。
IDLEでc05-q1-a.pyを開き、Run→Run Moduleしてみましょう。

```
"gw3EkivE1pw5EjtiksEq1E5pq2mEj7Eizo3umv2,"E1iqlENqtj7,E"j32E7w3E5qttEvm4mzEkwv4qvkmEumH"
```

b. aと同様に、messegeに平文1234567890、keyに鍵21を指定したプログラムc05-q1-b.pyを用意します。
IDLEでc05-q1-b.pyを開き、Run→Run Moduleしてみましょう。

```
HIJKLMNOPQ
```

問2

a. caserCipher.pyをベースにして、messageに暗号文Kv?uqwpfu?rncwukdng?gpqwijB、keyに鍵2、modeにdecryptを指定したプログラムc05-q2-a.pyを用意します。
IDLEでc05-q2-a.pyを開き、Run→Run Moduleしてみましょう。

```
It sounds plausible enough.
```

b. aと同様に、messegeに暗号文XCBSw88S18A1S　2SB41SE　.8zSEwAS5OD5A5x81V、keyに鍵22、modeにdecryptを指定したプログラムc05-q2-b.pyを用意します。
IDLEでc05-q1-b.pyを開き、Run→Run Moduleしてみましょう。

```
But all else of the world was invisible.
```

問3　モジュールのインポートにはimport文を使います。拡張子「.py」は不要です。

```
import watermelon
```

問4

a. IDLEでc05-q4-a.pyを開き、Run→Run Moduleしてみましょう。

```
foofoo
```

b. IDLEでc05-q4-b.pyを開き、Run→Run Moduleしてみましょう。

```
Goodbye
```

c. 対話型シェルで試してみましょう。赤い部分が解答です。

```
>>> print('f' not in 'foo')
False
```

文字列fooには文字列fが含まれています。つまり、含まれていないことを検証する式は偽（False）になります。

d. 対話型シェルで試してみましょう。赤い部分が解答です。

```
>>> print('foo' in 'f')
False
```

文字列fには文字列fooが含まれていません。つまり、含まれていることを検証する式は偽（False）になります。

e. 対話型シェルで試してみましょう。赤い部分が解答です。

```
>>> print('hello'.find('oo'))
-1
```

find()メソッドは対象の文字列が見つからなければ、-1を返します。文字列helloには文字列ooが含まれていないため、-1になります。

第6章　総当たり攻撃によるシーザー暗号の解読（P.116）問題

問1　次の暗号文を解読してください。各行は異なる鍵が使われているので、1行ずつ復号する必要があります。

```
qeFIP?eGSeECNNS,
5coOMXXcoPSZIWoQI,
avnl1olyD4l'ylDohww6DhzDjhuDil,

z.GM?.cEQc. 7Oc.7KcKMKHA9AGFK,
?MFYp2pPJJUpZSIJWpRdpMFY,
ZqH8sl5HtqHTH4s3lyvH5zH5spH4t pHzqHlH3l5K

Zfbi,!tif!xpvme!qspcbcmz!fbu!nfA
```

第6章　総当たり攻撃によるシーザー暗号の解読（P.116）解答

問1　解読の流れは次の通りです。問題文から1行ずつ暗号文を抽出して、caeserHacker.pyのmessageに指定して実行します。すると、総当たり攻撃の結果が表示されるので、そこから英文があれば、それが答えになります。

例えば、1行目の暗号文「qeFIP?eGSeECNNS,」を解読してみます。
caserHacker.pyをベースにして、messageに暗号文「qeFIP?eGSeECNNS,」を指定したプログラムc06-q1.pyを用意します。
IDLEでc06-q1.pyを開き、Run→Run Moduleしてみましょう。すると、次の実行結果が得られます。

```
Key #0: qeFIP?eGSeECNNS,
Key #1: pdEHO!dFRdDBMMR,
Key #2: ocDGN cEQcCALLQ,
Key #3: nbCFMObDPbB.KKP,
Key #4: maBEL9aCOaA?JJO,
Key #5: lZADK8ZBNZ.!IIN,
Key #6: kY.CJ7YAMY? HHM,
Key #7: jX?BI6X.LX!OGGL,
Key #8: iW!AH5W?KW 9FFK,
Key #9: hV .G4V!JVO8EEJ,
Key #10: gUO?F3U IU97DDI,
Key #11: fT9!E2TOHT86CCH,
Key #12: eS8 D1S9GS75BBG,
Key #13: dR7OCzR8FR64AAF,
Key #14: cQ69ByQ7EQ53..E,
Key #15: bP58AxP6DP42??D,
Key #16: aO47.wO5CO31!!C,
Key #17: ZN36?vN4BN2z  B,
Key #18: YM25!uM3AM1yOOA,
Key #19: XL14 tL2.Lzx99.,
Key #20: WKz3OsK1?Kyw88?,
Key #21: VJy29rJz!Jxv77!,
Key #22: UIx18qIy Iwu66 ,
Key #23: THwz7pHxOHvt55O,
Key #24: SGvy6oGw9Gus449,
Key #25: RFux5nFv8Ftr338,
Key #26: QEtw4mEu7Esq227,
Key #27: PDsv3lDt6Drp116,
Key #28: OCru2kCs5Cqozz5,
Key #29: NBqt1jBr4Bpnyy4,
Key #30: MApsziAq3Aomxx3,
Key #31: L.oryh.p2.nlww2,
Key #32: K?nqxg?o1?mkvv1,
Key #33: J!mpwf!nz!!juuz,
```

```
Key #34: I love my kitty,
Key #35: HOknudOlxOjhssx,
Key #36: G9jmtc9kw9igrrw,
Key #37: F8ilsb8jv8hfqqv,
Key #38: E7hkra7iu7geppu,
Key #39: D6gjqZ6ht6fdoot,
Key #40: C5fipY5gs5ecnns,
Key #41: B4ehoX4fr4dbmmr,
Key #42: A3dgnW3eq3callq,
Key #43: .2cfmV2dp2bZkkp,
Key #44: ?1belU1co1aYjjo,
Key #45: !zadkTzbnzZXiin,
Key #46:  yZcjSyamyYWhhm,
Key #47: OxYbiRxZlxXVggl,
Key #48: 9wXahQwYkwWUffk,
Key #49: 8vWZgPvXjvVTeej,
Key #50: 7uVYfOuWiuUSddi,
Key #51: 6tUXeNtVhtTRcch,
Key #52: 5sTWdMsUgsSQbbg,
Key #53: 4rSVcLrTfrRPaaf,
Key #54: 3qRUbKqSeqQOZZe,
Key #55: 2pQTaJpRdpPNYYd,
Key #56: 1oPSZIoQcoOMXXc,
Key #57: znORYHnPbnNLWWb,
Key #58: ymNQXGmOamMKVVa,
Key #59: xlMPWFlNZlLJUUZ,
Key #60: wkLOVEkMYkKITTY,
Key #61: vjKNUDjLXjJHSSX,
Key #62: uiJMTCiKWiIGRRW,
Key #63: thILSBhJVhHFQQV,
Key #64: sgHKRAgIUgGEPPU,
Key #65: rfGJQ.fHTfFDOOT,
```

全体を眺めると、鍵34で英文が得られました。他に英文となる鍵は存在しないので、鍵34の英文が解読結果になります。
なお、正しい解読結果を素早く探し出すには、空白の有無、数字列の有無に注目します。英文であれば、適度な位置に空白があり、文中に意味のない数字列が登場する可能性は少ないためです。
なお、英文の検出をプログラムに実装するには11章を参考にしてください。英文の検出プログラム（detectEnglish.py）を紹介しています。

問1の解説に戻ります。
2行目以降も同様にして解読できます。全文の解読結果は次の通りです。ただし、（　）内の記述は鍵になります。また、3行目の暗号文には一重引用符を含んでいるので、messageを二重引用符で挟むように修正します。そして、空行を暗号化すると空行になるので、空行を解読する必要はありません。

```
I love my kitty,                          (鍵34)
My kitty loves me,                        (鍵44)
Together we're happy as can be,           (鍵7)

Though my head has suspicions,            (鍵32)
That I keep under my hat,                 (鍵45)
Of what if I shrank to the size of a rat. (鍵11)

Yeah, she would probably eat me.          (鍵1)
```

第7章　転置式暗号で暗号化する（P.142）問題

問1　紙と鉛筆を使って、次のメッセージを転置式暗号で暗号化してください。鍵は9とします。便宜上、文字数を表記しました。

・Underneath a huge oak tree there was of swine a huge company. (61文字)
・That grunted as they crunched the mast: For that was ripe and fell full fast. (77文字)
・Then they trotted away for the wind grew high: One acorn they left, and no more might you spy. (94文字)

問2　次のプログラムにおいて、それぞれの変数spamはグローバル変数とローカル変数のどちらですか？

```
spam = 42
def foo():
    global spam
    spam = 99
    print(spam)
```

問3　次の各行はどのように評価されますか？

```
[0, 1, 2, 3, 4][2]
[[1, 2], [3, 4]][0]
[[1, 2], [3, 4]][0][1]
['hello'][0][1]
[2, 4, 6, 8, 10][1:3]
list('Hello world!')
list(range(10))[2]
```

問4　次の各行はどのように評価されますか？

```
len([2, 4])
len([])
len(['', '', ''])
[4, 5, 6] + [1, 2, 3]
3 * [1, 2, 3] + [9]
42 in [41, 42, 42, 42]
```

問5　本章で新たに紹介した4つの拡張代入演算子は何ですか？

第7章　転置式暗号で暗号化する（P.142）解答

問1　紙上で転置式暗号に暗号化する手順を再掲します。

1. メッセージと鍵の文字数を数えます。
2. 鍵と等しい数の箱を並べて描きます（例えば、鍵が8の場合は8個の箱）。
3. 箱を左から右に埋める作業をします。1つの箱に1文字を入れていきます。
4. 空きがなくなり文字がまだ残っていたら、別の行の箱に入れます。
5. 最後の文字に達したとき、最終行の未使用の箱には影を付けます。
6. 左上から開始し、各列を下に降りていき、文字を書き出します。列の一番下に来ると、右隣の列に移動します。影付きの箱はスキップします。書き出されたものが暗号文になります。

上記の手順を用いて、1つ目のメッセージを暗号化してみます。

1：メッセージは61文字、鍵は9です。

2：9個の箱を並べて描きます。

3～5：箱に1文字ずつ埋めていき箱を拡張すると、次に示す格子状の箱が得られます。ただし、_は空白を意味します。

U	n	d	e	r	n	e	a	t
h	_	a	_	h	u	g	e	_
o	a	k	_	t	r	e	e	_
t	h	e	r	e	_	w	a	s
_	o	f	_	s	w	i	n	e
_	a	_	h	u	g	e	_	c
o	m	p	a	n	y	,		

6：左上から縦に読み取り一番下に来たら、右隣に移り同様に読み取ります。これを繰り返し、影付きの箱はスキップします。すると、次の暗号文が得られます。

```
Uhot__on_ahoamdakef_pe__r_harhtesunnur_wgyegewie,aeean_t__sec
```

2つ目以降も同様にして暗号化できます。
2つ目のメッセージを暗号化する過程で、次に示す格子状の箱が得られます。コロン（:）やカンマ（,）の直後には空白があることに注意してください。

T	h	a	t	_	g	r	u	n
t	e	d	_	a	s	_	t	h
e	y	_	c	r	u	n	c	h
e	d	_	t	h	e	_	m	a
s	t	:	_	F	o	r	_	t
h	a	t	_	w	a	s	_	r
i	p	e	_	a	n	d	_	f
e	l	l	_	f	u	l	l	_
f	a	s	t	.				

暗号文は次の通りです。

```
Tteeshiefheydtaplaad__:telst_ct____t_arhFwaf.gsueoanur_n_rsdlutcm___lnhhatrf_
```

3つ目のメッセージを暗号化する過程で、次に示す格子状の箱が得られます。

T	h	e	n	_	t	h	e	y
_	t	r	o	t	t	e	d	_
a	w	a	y	_	f	o	r	_
t	h	e	_	w	i	n	d	_
g	r	e	w	_	h	i	g	h
:	_	O	n	e	_	a	c	o
r	n	_	t	h	e	y	_	l
e	f	t	,	_	a	n	d	_
n	o	_	m	o	r	e	_	m
i	g	h	t	_	y	o	u	_
s	p	y	.					

暗号文は次の通りです。

```
T_atg:renishtwhr_nfogperaeeO_t_hynoy_wnt,mt._t_w_eh_o_ttfih_earyheoniayneoedrdgc_d_uy___hol_m
_
```

問2　このプログラム中のすべての変数spamはグローバル変数です。
global文はfoo()内の変数spamをグローバル変数にしています。

問3　対話型シェルで試してみましょう。赤い部分が解答です。

```
>>> [0, 1, 2, 3, 4][2]
2
>>> [[1, 2], [3, 4]][0]
[1, 2]
>>> [[1, 2], [3, 4]][0][1]
2
```

```
>>>
>>> ['hello'][0]
'hello'
>>> ['hello'][0][1]
'e'
>>> [2, 4, 6, 8, 10][1:3]
[4, 6]
>>> list('Hello world!')
['H', 'e', 'l', 'l', 'o', ' ', 'w', 'o', 'r', 'l', 'd', '!']
>>>
>>> range(10)
range(0, 10)
>>> list(range(10))
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> list(range(10))[2]
2
```

問4　対話型シェルで試してみましょう。赤い部分が解答です。

```
>>> len([2, 4])
2
>>> len([])
0
>>> len(['', '', ''])
3
>>> [4, 5, 6] + [1, 2, 3]
[4, 5, 6, 1, 2, 3]
>>> 3 * [1, 2, 3] + [9]
[1, 2, 3, 1, 2, 3, 1, 2, 3, 9]
>>> 42 in [41, 42, 42, 42]
True
```

問5

+=
-=
*=
/=

第8章　転置式暗号を復号する（P.159）問題

問1　紙と鉛筆を使って、鍵9で次のメッセージを復号してください。_は空白を意味します。便宜上、文字数を表記しました。

・H_cb__irhdeuousBdi___prrtyevdgp_nir__eerit_eatoreechadihf_paken_ge_b_te_dih_aoa.da_tts_tn
（89文字）
・A_b__drottthawa_nwar_eci_t_nlel_ktShw_leec,hheat_.na__e_soogmah_a__ateniAcgakh_dmnor__
（86文字）
・Bmmsrl_dpnaua!toeboo'ktn_uknrwos._yaregonr_w_nd,tu__oiady_hgtRwt___A_hhanhhasthtev__e_t_e_
_eo
（93文字）

問2　対話型シェルに次のコードを入力すると、各行は何を表示しますか？

```
>>> math.ceil(3.0)
>>> math.floor(3.1)
>>> round(3.1)
>>> round(3.5)
>>> False and False
>>> False or False
>>> not not True
```

問3　and、or、not演算子の完全な真理値表を書いてください。

問4　次のうち正しいものはどれですか？

```
if __name__ == '__main__':
if __main__ == '__name__':
if _name_ == '_main_':
if _main_ == '_name_':
```

第8章　転置式暗号を復号する（P.159）解答

問1　紙上で転置式暗号を復号する手順を再掲します。

1. 暗号文の長さを鍵の値で割ってから小数点以下を切り上げて、必要な列数を計算します。
2. 格子状の箱を描きます。ステップ1で計算した列数を使用します。行数は鍵の値と同じです。
3. 箱の総数（行数に列数を掛けた結果）から暗号文の長さを引くことで、影付きの箱の数を計算できます。
4. ステップ3で計算した結果の分だけ、右端の列の最下部から影付きにします。
5. 一番上の行から始めて、左から右へ暗号文の文字を記入していきます。その際、影付きの箱はスキップします。
6. 一番左の列を上から下に読み、各列でも同じことを続けることで、平文が得られます。

上記の手順を用いて、1つ目のメッセージを復号してみます。

1：メッセージは89文字、鍵は9です。89÷9=9余り8と計算でき、列数は10になります。

2：行が9、列が10の格子状の箱を描きます。

3：箱の総数は90であり、暗号文の長さ89を引くと、1になります。つまり、影付きの箱は1個です。

4：右下の箱を影付きにします。

5：箱に1文字ずつ埋めていくと、次に示す格子状の箱が得られます。ただし、_は空白を意味します。

H	_	c	b	_	_	i	r	h	d
e	u	o	u	s	B	d	i	_	_
_	p	r	r	t	y	e	v	d	g
p	_	n	i	r	_	_	e	e	r
i	t	_	e	a	t	o	r	e	e
c	h	a	d	i	h	f	_	p	a
k	e	n	_	g	e	_	b	_	t
e	_	d	i	h	_	a	o	a	.
d	a	_	t	t	s	_	t	n	

6：左上から縦に読み取り一番下に来たら、右隣に移り同様に読み取ります。これを繰り返し、影付きの箱はスキップします。すると、次の暗号文が得られます。

```
He_picked_up_the_acorn_and_buried_it_straight_By_the_side_of_a_river_both_deep_and_great.
```

2つ目以降も同様にして復号できます。
2つ目のメッセージを復号する過程で、次に示す格子状の箱（9行×10列）が得られます。

A	_	b	_	_	d	r	o	t	t
t	h	a	w	a	_	n	w	a	r
_	e	c	i	_	t	_	n	l	e
l	_	k	t	S	h	w	_	l	e
e	c	,	h	h	e	a	t	_	.
n	a	_	_	e	_	s	o	o	
g	m	a	h	_	a	_	_	a	
t	e	n	i	A	c	g	a	k	
h	_	d	m	n	o	r	_	_	

平文は次の通りです。

```
At_length_he_came_back,_and_with_him_a_She_And_the_acorn_was_grown_to_a_tall_oak_tree.
```

3つ目のメッセージを復号する過程で、次に示す格子状の箱（9行×11列）が得られます。

B	m	m	s	r	l	_	d	p	n	a
u	a	!	t	o	e	b	o	o	'	k
t	n	_	u	k	n	r	w	o	s	.
_	y	a	r	e	g	o	n	r	_	
w	_	n	d	,	t	u	_	_	o	
i	a	d	y	_	h	g	t	R	w	
t	_	_	_	A	_	h	h	a	n	
h	h	a	s	t	h	t	e	v	_	
_	e	_	t	_	e	_	_	e	o	

平文は次の通りです。

```
But_with_many_a_hem!_and_a_sturdy_stroke,_At_length_he_brought_down_the_poor_Raven's_own_oak.
```

問2　対話型シェルで試すには、最初にmathモジュールをインポートする必要があります。

```
>>> import math
>>> math.ceil(3.0)
3
>>> math.floor(3.1)
3
>>> round(3.1)
3
>>> round(3.5)
4
>>> False and False
False
>>> False or False
False
>>> not not True
True
```

問3

and演算子

A	B	A and B
True	True	True
True	False	False
False	True	False
False	False	False

or演算子

A	B	A or B
True	True	True
True	False	True
False	True	True
False	False	False

not演算子

A	not A
True	False
False	True

問4

```
if __name__ == '__main__':   (1行目)
```

第9章　プログラムテスト用プログラムを作成する（P.178）問題

問1　次のプログラムを実行して、数8が表示されたとき、次回実行時には何が表示されますか？

```
import random
random.seed(9)
print(random.randint(1, 10))
```

問2　次のプログラムを実行すると何が表示されますか？

```
spam = [1, 2, 3]
eggs = spam
ham = eggs
ham[0] = 99
print(ham == spam)
```

問3　deepcopy()関数はどのモジュールに含まれていますか？

問4　次のプログラムを実行すると何が表示されますか？

```
import copy
spam = [1, 2, 3]
eggs = copy.deepcopy(spam)
ham = copy.deepcopy(eggs)
ham[0] = 99
print(ham == spam)
```

第9章　プログラムテスト用プログラムを作成する（P.178）解答

問1　「8」が表示されます。
シードに同じ値を設定しているので、同じ乱数が生成されます。
IDLEでc09-q1.pyを開き、2度Run→RunModuleしてみましょう。

問2　対話型シェルで1行ずつ実行しながら、各変数の内容を確かめてみましょう。

```
>>> spam = [1, 2, 3] # 1行目
>>> spam
[1, 2, 3]
>>> eggs = spam # 2行目
>>> eggs
[1, 2, 3]
>>> spam
[1, 2, 3]
>>> ham = eggs # 3行目
>>> ham
[1, 2, 3]
>>> eggs
[1, 2, 3]
>>> spam
[1, 2, 3]
>>> ham[0] = 99 # 4行目
>>> ham
[99, 2, 3]
>>> eggs
[99, 2, 3]
>>> spam
[99, 2, 3]
>>> print(ham == spam) # 5行目
True
```

問3　「copyモジュール」です。copyモジュールをインポート後、dir()関数で確認できます。

```
>>> import copy
>>> dir(copy)
['Error', '__all__', '__builtins__', '__cached__', '__doc__', '__file__', '__loader__', '__na
me__', '__package__', '__spec__', '_copy_dispatch', '_copy_immutable', '_deepcopy_atomic', '_
deepcopy_dict', '_deepcopy_dispatch', '_deepcopy_list', '_deepcopy_method', '_deepcopy_tuple'
, '_keep_alive', '_reconstruct', 'copy', 'deepcopy', 'dispatch_table', 'error']
```

問4　対話型シェルで1行ずつ実行しながら、各変数の内容を確かめてみましょう。

```
>>> import copy # 1行目
>>> spam = [1, 2, 3] # 2行目
>>> spam
[1, 2, 3]
>>> eggs = copy.deepcopy(spam) # 3行目
>>> eggs
[1, 2, 3]
>>> spam
[1, 2, 3]
>>> ham = copy.deepcopy(eggs) # 4行目
>>> ham
[1, 2, 3]
>>> eggs
[1, 2, 3]
>>> spam
[1, 2, 3]
>>> ham[0] = 99 # 5行目
>>> ham
[99, 2, 3]
>>> eggs
[1, 2, 3]
>>> spam
[1, 2, 3]
>>> print(ham == spam) # 6行目
False
```

第10章　ファイルの暗号化と復号（P.195）問題

問1　os.exists()とos.path.exists()のどちらが正しいですか？

問2　UNIX時間の基準時刻はいつですか？

問3　次の式は何に評価されますか？

```
'Foobar'.startswith('Foo')
'Foo'.startswith('Foobar')
'Foobar'.startswith('foo')
'bar'.endswith('Foobar')
'Foobar'.endswith('bar')
'The quick brown fox jumped over the yellow lazy dog.'.title()
```

第10章　ファイルの暗号化と復号（P.195）解答

問1　os.path.exists()が正しいです。

問2　GMT（UTC）で表記すると1970/1/1 00:00:00になります。
対話型シェルにて次のように実行すると確認できます。

```
>>> import datetime
>>> dt = datetime.datetime.fromtimestamp(0, datetime.timezone.utc)
>>> print(dt)
1970-01-01 00:00:00+00:00
```

問3　対話型シェルで試してみましょう。赤い部分が解答です。

```
>>> 'Foobar'.startswith('Foo')  # 'Foobar'は'Foo'で始まるか
True
>>> 'Foo'.startswith('Foobar')  # 'Foo'は'Foobar'で始まるか
False
>>> 'Foobar'.startswith('foo')  # 'Foobar'は'foo'で始まるか
False
>>> 'bar'.endswith('Foobar')    # 'bar'は'Foobar'で終わるか
False
>>> 'Foobar'.endswith('bar')    # 'Foobar'は'bar'で終わるか
True

# 単語の先頭の一文字を大文字、他を小文字に変換
>>> 'The quick brown fox jumped over the yellow lazy dog.'.title()
'The Quick Brown Fox Jumped Over The Yellow Lazy Dog.'
```

第11章　プログラムによる英語の検出（P.222）問題

問1　次のコードを実行すると何が表示されますか？

```
spam = {'name': 'Al'}
print(spam['name'])
```

問2　このコードを実行すると何が表示されますか？

```
spam = {'eggs': 'bacon'}
print('bacon' in spam)
```

問3　次の辞書spam内の値を表示するforループのコードはどのようなものですか？

```
spam = {'name': 'Zophie', 'species':'cat', 'age':8}
```

問4　次の行を実行すると何が表示されますか？

```
print('Hello, world!'.split())
```

問5　次のコードを実行すると何が表示されますか？

```
def spam(eggs=42):
    print(eggs)
spam()
spam('Hello')
```

問6　次の文において、有効な英単語の割合（パーセンテージ）はどれだけありますか？

```
"Whether it's flobulllar in the mind to quarfalog the slings and arrows of outrageous guuuuu
uuuur."
```

第11章　プログラムによる英語の検出（P.222）解答

問1　対話型シェルで試してみましょう。赤い部分が解答です。

```
>>> spam = {'name': 'Al'}      # キーを'name'、値を'Al'として辞書spamを定義
>>> print(spam['name'])        # キーが'name'である辞書spamの値を表示
Al
```

問2　対話型シェルで試してみましょう。赤い部分が解答です。

```
>>> spam = {'eggs': 'bacon'}  # キーを'eggs'、値を'bacon'として辞書spamを定義
>>> print('bacon' in spam)     # 辞書spamに'bacon'というキーが存在するかを表示
False
```

11.5.4で解説しましたが、辞書にin演算子を用いることで特定のキーが存在するかどうかを確認できます。
辞書spamは'bacon'という値を持ちますが、'bacon'というキーを持ちません（'eggs'というキーのみを持ちます）。よって、'bacon' in spamはFalseを返します。

問3　次のように、for-in文を使います。

```
spam = {'name': 'Zophie', 'species':'cat', 'age':8} # 問題文にある辞書spamの定義
for key in spam:
    print(spam[key])
```

このプログラムはch11-q3.pyとして保存されています。実行結果は次のとおりです。

```
Zophie
cat
8
```

問4　IDLEでch11-q4.pyを開き、Run→Run Moduleしてみましょう。

```
['Hello,', 'world!']
```

split()関数は文字列'Hello, world!'をホワイトスペース文字で分割し、リストとして返します。

問5　IDLEでch11-q5.pyを開き、Run→Run Moduleしてみましょう。

```
42
Hello
```

最初のspam()関数の呼び出しでは引数を指定しておらず、デフォルト引数の値を表示します。
2度目のspam()関数の呼び出しでは引数'Hello'を指定しており、'Hello'を表示します。

問6　80%です。
次表のとおり15単語のうち12単語が英単語です。12÷15＝0.8と計算できるので80%となります。
'flobulllar'、'quarfalog'、'guuuuuuuuur'の3つは英単語ではありません。

1	Whether	○	2	it's	○	3	flobulllar	×	4	in	○
5	the	○	6	mind	○	7	to	○	8	quarfalog	×
9	the	○	10	slings	○	11	and	○	12	arrows	○
13	of	○	14	outrageous	○	15	guuuuuuuuur	×			

第12章　転置式暗号を解読する（P.234）問題

問1　この式は何に評価されますか？

```
'   Hello world'.strip()
```

問2　ホワイトスペース文字とはどの文字ですか？

問3　'Hello world'.strip('o')は依然としてなぜ文字'o'が残った文字列に評価されるのですか？

問4　'xxxHelloxxx'.strip('X')は依然としてなぜ文字'x'が残った文字列に評価されるのですか？

第12章　転置式暗号を解読する（P.234）解答

問1　対話型シェルで試してみましょう。赤い部分が解答です。

```
>>> '   Hello world'.strip()
'Hello world'
```

strip()関数は、文字列の先頭および末尾部分から引数に指定した文字集合を除去したコピーを返します。引数を省略した場合は、空白文字が除去されます。

問2　空白文字、タブ、改行
厳密な定義を知りたければhttps://docs.python.org/ja/3/library/string.htmlをご覧ください。

問3　strip()は文字列の左側（頭側）あるいは右側（末尾側）から文字'o'を取り除くだけだからです。

```
>>> 'Hello world'.strip('o')
'Hello world'
>>> 'oh, Hello world'.strip('o')
'h, Hello world'
>>> 'me, too'.strip('o')
'me, t'
```

問4　strip('X')は'X'を削除しますが、'x'を削除しないためです。

```
>>> 'xxxHelloxxx'.strip('X')
'xxxHelloxxx'
>>> 'xxxHelloxxx'.strip('x')
'Hello'
```

第13章　アフィン暗号のためのモジュラー算術モジュール（P.251）問題

問1　次の式は何に評価されますか？

```
17 % 1000
5 % 5
```

問2　10と15のGCDは何ですか？

問3　spam, eggs = 'hello', 'world'を実行した後、spamはどうなりますか？

問4　17と31のGCDは1です。17と31は互いに素ですか？

問5　なぜ6と8は互いに素にならないのですか？

問6　モジュラーCにおけるAのモジュラー逆数を求める公式はどのようなものですか？

第13章　アフィン暗号のためのモジュラー算術モジュール（P.251）解答

問1　まずは暗算で何に評価されるかを考えてみます。

17 ÷ 1000 ＝ 0 余り 17なので、17 % 1000 は 17 になるはずです。
 5 ÷   5 ＝ 1 余り 0なので、 5 %   5 は 0 になるはずです。

次に、対話型シェルで結果を確認してみましょう。

```
>>> 17 % 1000
17
>>> 5 % 5
0
```

問2　10と15のGCDは5です（10と15を割り切る数のうちもっとも大きいものは5であるため）。
本文では独自にgcd()関数を実装しましたが、mathモジュールにもgcd()関数が定義されています。
対話型シェルで次のように使います。

```
>>> import math
>>> math.gcd(10, 15)
5
```

問3　多重代入を用いており、spamの値は'hello'になります。

```
>>> spam, eggs = 'hello', 'world'
>>> spam
'hello'
>>> eggs
'world'
```

問4　はい、17と31は互いに素です。互いに素であることの定義は、GCDが1になることだからです。
本文では独自にgcd()関数を実装しましたが、mathモジュールにもgcd()関数が定義されています。
対話型シェルで次のように使います。

```
>>> import math
>>> math.gcd(17, 31)
1
```

問5　GCDが2になり、1ではないからです。

```
>>> import math
>>> math.gcd(6, 8)
2
```

なお、偶数は2の倍数であるため、2つの偶数のGCDは常に2以上の倍数になります。よって、GCDが1では

ないので、互いに素になりません。

問6　モジュラーCにおけるAのモジュラー逆数iは、(A * i) % C == 1という式を満たします。

第14章　アフィン暗号のプログラミング（P.268）問題

問1　アフィン暗号はどの2つの暗号の組み合わせですか？

問2　タプルとは何ですか？　タプルとリストの違いは何ですか？

問3　鍵Aが1のとき、アフィン暗号が脆弱になる理由は何ですか？

問4　鍵Bが0のとき、アフィン暗号が脆弱になる理由は何ですか？

第14章　アフィン暗号のプログラミング（P.268）解答

問1　アフィン暗号は乗法暗号とシフト暗号（あるいはシーザー暗号）の組み合わせです。

問2　タプルとはリストのように複数の値を保持できるデータ型です。
しかし、リストとは異なり、値は不変であり、変更できません。

問3　任意の数に1を掛けても数は変わらないので、アフィン暗号の乗算処理では同じ文字に変換されてしまいます。
この鍵Aで暗号化することは、完全にシフト暗号と同等になります。つまり、シフト暗号の脆弱性が現れます。

問4　任意の数に0を足しても数は変わらないので、アフィン暗号の加算処理では同じ文字に変換されてしまいます。
この鍵Bで暗号化することは、完全に乗法暗号と同等になります。つまり、乗法暗号の脆弱性が現れます。

第15章　アフィン暗号を解読する（P.280）問題

問1　2 ** 5は何に評価されますか？

問2　6 ** 2は何に評価されますか？

問3　次のコードを実行すると何が表示されますか？

```
for i in range(5):
    if i == 2:
        continue
    print(i)
```

問4　別のプログラムがimport affineHackerを実行したとき、affineHacker.pyのmain()関数は呼び出されますか？

第15章　アフィン暗号を解読する（P.280）解答

問1　対話型シェルで試してみましょう。赤い部分が解答です。

```
>>> 2 ** 5
32
```

2 ** 5は2^5＝2×2×2×2×2＝32です。

問2　対話型シェルで試してみましょう。赤い部分が解答です。

```
>>> 6 ** 2
36
```

問1と同様に、6 ** 2は6^2＝6×6＝36です。

問3　IDLEでc15-q3.pyを開き、Run→Run Moduleしてみましょう。

```
0
1
3
4
```

i == 2のときだけ、continue文によって制御が（print(i)に到達することなく）for文の先頭に戻ります。そのため2だけが表示されません。

問4　いいえ。プログラムがインポートされたとき、変数__name__には'affineHacker'がセットされているため、main()関数は呼び出されません。

第16章　単一換字式暗号のプログラミング（P.298）問題

問1　強力なコンピュータがあっても、単一換字式暗号に対して総当たり攻撃を適用できないのはなぜですか？

問2　このコードを実行したとき、変数spamには何が含まれますか？

```
spam = [4, 6, 2, 8]
spam.sort()
```

問3　ラッパー関数とは何ですか？

問4　'hello'.islower()は何に評価されますか？

問5　'HELLO 123'.isupper()は何に評価されますか？

問6　'123'.islower()は何に評価されますか？

第16章　単一換字式暗号のプログラミング（P.298）解答

問1　強力なコンピュータであっても、鍵の候補が多すぎるため、単一換字式暗号に対して総当たり攻撃を適用できません（厳密にいうと、総当たり攻撃をしても成功する可能性はほぼないという意味です）。

問2　対話型シェルで試してみましょう。赤い部分が解答です。

```
>>> spam = [4, 6, 2, 8]
>>> spam
[4, 6, 2, 8]
>>> spam.sort()
>>> spam
[2, 4, 6, 8]
```

問3　ラッパー関数は、引数を別の関数に渡しつつ呼び出して、その結果を返すような関数です。
実際には、引数を若干変更してから別の関数を呼び出したり、その関数の返り値を加工してから返すこともあります。

問4　対話型シェルで試してみましょう。赤い部分が解答です。

```
>>> 'hello'.islower()
True
```

小文字が少なくとも1文字あり、かつ大文字が含まれていないとき、islower()はTrueを返します。

問5　対話型シェルで試してみましょう。赤い部分が解答です。

```
>>> 'HELLO 123'.isupper()
True
```

大文字が少なくとも1文字あり、かつ小文字が含まれていないとき、isupper()はTrueを返します。

問6　対話型シェルで試してみましょう。赤い部分が解答です。

```
>>> '123'.islower()
False
```

小文字が少なくとも1文字あり、かつ大文字が含まれていないとき、islower()はTrueを返します。
'123'には小文字が1文字も含まれていないので、この式はFalseを返します。

第17章　単一換字式暗号を解読する（P.332）問題

問1　helloの単語パターンは何ですか？

問2　mammothとgogglesは同じ単語パターンになりますか？

問3　Alleged、efficiently、poodleのうちどれが暗号単語PYYACAOの平文単語になる可能性がありますか？

第17章　単一換字式暗号を解読する（P.332）解答

問1

helloの単語パターンは'0.1.2.2.3'です。

問2

はい、mammothとgogglesは同じ単語パターンになります。
単語パターンは'0.1.0.0.2.3.4'です。

問3

暗号単語PYYACAOの単語パターンは'0.1.1.2.3.2.4'です。
これと同じ単語パターンになるのは、Allegedです。

第18章　ヴィジュネル暗号のプログラミング（P.347）問題

問1　ヴィジュネル暗号は1つの鍵ではなく複数の鍵を使用します。複数の鍵を用いることを除いて、ヴィジュネル暗号はどの暗号に似ていますか？

問2　ヴィジュネル暗号の鍵の長さが10であれば、鍵の候補の総数はいくつありますか？

a. 数百
b. 数千
c. 数百万
d. 1兆を超える

問3　ヴィジュネル暗号はどのような種類の暗号ですか？

第18章　ヴィジュネル暗号のプログラミング（P.347）解答

問1　複数の鍵を用いることを除けば、ヴィジュネル暗号はシーザー暗号と似ています。
ヴィジュネル暗号に長さが1である鍵を使用すると、シーザー暗号そのものになります。

問2　1兆より大きくなるので「d. 1兆を超える」になります。
具体的な鍵の総数は141,167,095,653,376になります。

問3　ヴィジュネル暗号は、複数の置換集合を使用する（シーザー暗号のような）シフト暗号であるため、多表式暗号として知られています。

第19章　頻度分析（P.373）問題

問1　頻度分析とは何ですか？

問2　英語でもっともよく使われている6文字は何ですか？

問3　次のコードを実行したとき、変数spamには何が格納されていますか？

```
spam = [4, 6, 2, 8]
spam.sort(reverse=True)
```

問4　変数spamが辞書を含むとき、辞書中のキー一覧のリスト値をどのようにして取得できますか？

第19章　頻度分析（P.373）解答

問1　頻度分析は、暗号解読手法の1つで、暗号文における文字の出現頻度を数える手法です。

問2　英語でもっともよく使われている6文字はETAOINです。

問3　[8, 6, 4, 2]が格納されています。対話型シェルで試してみましょう。赤い部分が解答です。

```
>>> spam = [4, 6, 2, 8]
>>> spam.sort(reverse=True)
>>> spam
[8, 6, 4, 2]
```

問4

対話型シェルで試してみましょう。
❶変数spamに辞書{'cats': 10, 'dogs': 3, 'mice': 3}を代入してみます。
❷spam.keys()とすると、辞書spamのキーを取得できます。
❸list()関数に❷の結果を渡すと、目的のリストを得られます。

```
❶>>> spam = {'cats': 10, 'dogs': 3, 'mice': 3}
  >>> spam
  {'cats': 10, 'dogs': 3, 'mice': 3}
❷>>> spam.keys()
  dict_keys(['cats', 'dogs', 'mice'])
❸>>>list(spam.keys())
  ['cats', 'dogs', 'mice']
```

第20章　ヴィジュネル暗号を解読する（P.422）問題

問1　辞書式攻撃とは何ですか？

問2　カシスキー検査で暗号文の何を明らかにできますか？

問3　set()関数を使って、リストを集合に変換するとき、何が起きますか？

問4　変数spamに['cat', 'dog', 'mouse', 'dog']が格納されているなら、このリストには4つの要素があります。list(set(spam))で得られるリストは、何個の要素を持ちますか？

問5　次のコードを実行すると何が表示されますか？

```
print('Hello', end='')
print('World')
```

第20章　ヴィジュネル暗号を解読する（P.422）解答

問1

辞書式攻撃とは辞書ファイルを用いた総当たり攻撃です。

問2

カシスキー検査はヴィジュネル暗号の暗号文から鍵長を特定します。

問3

set()関数を使ってリストを集合に変換すると、重複する値が削除され、値の順序が失われます。
リストとは異なり、集合の値には順序がありません。

問4

3個の要素を持ちます（重複した値'dog'は削除されます）。
対話型シェルで試してみましょう。

```
>>> spam = ['cat', 'dog', 'mouse', 'dog']
>>> list(set(spam))
['dog', 'mouse', 'cat']
```

問5

IDLEでc20-q5.pyを開き、Run→Run Moduleしてみてください。
次のように1行で表示されます。

```
HelloWorld
```

第21章　ワンタイムパッド暗号（P.429）問題

問1　本章ではなぜワンタイムパッド暗号のプログラムがなぜ提示されなかったのでしょうか？

問2　ツータイムパッド暗号はどのような暗号と同等でしょうか？

問3　平文メッセージの2倍の長さの鍵を使うと、ワンタイムパッドの安全性は2倍になりますか？

第21章　ワンタイムパッド暗号（P.429）解答

問1

メッセージ長と同じ長さのランダムな鍵を用いてヴィジュネル暗号のプログラムを使用すれば、ワンタイムパッド暗号のプログラムとして代用できるからです。

問2

ワンタイムパッドを2回使って得られた暗号文は、ヴィジュネル暗号の暗号文と同等になります。

問3

いいえ、ワンタイムパッドの安全性は2倍にはなりません。平文を暗号化するのに同一の長さの鍵を必要とします。つまり、平文の長さの2倍ある鍵であれば、その鍵の半分は使われません。そして、通常のワンタイムパッド暗号と同等の安全性になります。
余談ですが、ワンタイムパッド暗号は情報理論的に安全と呼ばれる最高の安全性になります。よって、鍵の長さを平文の長さよりも大きくしたところで、最高の安全性を超えることはありません。

第22章　素数の検索と生成（P.448）問題

問1　素数はいくつありますか？

問2　素数でない整数を何といいますか？

問3　素数を見つけるためのアルゴリズムを2つ挙げてください。

第22章　素数の検索と生成（P.448）解答

問1　素数は無限に存在します。よって、最大の素数は存在しません。

問2　合成数といいます。

問3　下記よりいずれか2つを挙げてください。

・試し割り法
・エラトステネスのふるい
・Rabin-Miller法　など

本書では上記のアルゴリズムを紹介しましたが、その他にも存在します。

第23章　公開鍵暗号の鍵を生成する（P.467）問題

問1　対称暗号と非対称暗号の違いは何ですか？

問2　アリスは公開鍵と秘密鍵を生成します。残念ながら、アリスは秘密鍵をなくしてしまいました。

a. 他の人はアリスに暗号文を送信できますか？
b. アリスは以前に自分宛に送信された暗号文を復号できますか？
c. アリスは文書にデジタル署名を添付できますか？
d. 他の人は、アリスが以前に署名したデジタル署名を検証できますか？

問3　認証や機密性とは何ですか？　それらの違いは何ですか？

問4　否認防止とは何ですか？

第23章　公開鍵暗号の鍵を生成する（P.467）解答

問1

対称暗号は暗号化と復号に同じ鍵を用います。
一方、非対称暗号は暗号化と復号でそれぞれ異なる鍵を用います。

問2

a.
はい。他の人はアリスに暗号化されたメッセージを送信するため、アリスの公開鍵を使用します。
厳密にいうと、送信するためにアリスの公開鍵が必要なわけでなく、（アリスが復号できる）暗号文を生成するために必要です。暗号化では秘密鍵を用いないため、アリスが秘密鍵をなくしていても問題になりません。

b.
いいえ。アリスは暗号化されたメッセージを復号するために自身の秘密鍵が必要です。

c.
いいえ。アリスは文書にデジタル署名を付けるためには、自分の秘密鍵が必要だからです。
厳密にいうと、秘密鍵が必要というわけではなく、秘密に管理しなければならない（アリス自身の）署名鍵を用います。

d.
はい。アリスが秘密鍵を紛失する前の署名を確認するには、アリスの公開鍵を用います。
厳密にいうと、デジタル署名を検証するには公開されている（アリスの）検証鍵を用います。

問3

認証は個人の身元を確認することです。一方、機密性は情報を秘密にして、盗聴者がいても内容を読み取れないようにすることです。

問4

否認防止とはデジタル署名の特徴の1つであり、かつて署名したにもかかわらず、後で署名していないと主張することを防ぐことです。