Cryptography Project Walkthrough

Python 2.7 edition

Cryptography Regular Edition

- *Implement Caesar Cipher*
  - *both encrypt and decrypt functions*

Hacker Edition

- *regular edition +*
- *Vigenère's Cipher*
  - *(vision-a*

Confederate cipher disk

Caesar cipher:

plain text	*A*	*B*	*C*	*D*	*E*	*F*	*G*	*H*	*I*	*J*	...	*Z*
encrypted	*A*	*B*	*C*	*D*	*E*	*F*	*G*	*H*	*I*	*J*	...	*Z*

k = how much to shift

Caesar cipher:

plain text	*A*	*B*	*C*	*D*	*E*	*F*	*G*	*H*	*I*	*J*	...	*Z*
encrypted		*A*	*B*	*C*	*D*	*E*	*F*	*G*	*H*	*I*	*J*	*K*

bed

hide

$k = 1$

Caesar cipher:

plain text	*A*	*B*	*C*	*D*	*E*	*F*	*G*	*H*	*I*	*J*	...	*Z*
encrypted	*V*	*W*	*X*	*Y*	*Z*	*A*	*B*	*C*	*D*	*E*	*F*	*G*

k = 5

encrypt: cafe

fig

basic idea in English

- *each letter represented by a number*
  A = 0, B = 1 ... Z = 25
- *p_i is the i^{th} letter of the plain text.*
- *c_i is the i^{th} letter of the cipher text*
- *k is the secret key, a non-negative number-- how much to shift.*

the formula

- $c_i = (p_i + k)$
- *any problems? (think my last name)*

the revised formula

- $c_i = (p_i + k) \quad \% 26$

Doing it by pencil:

plain text

0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	...	*25*
A	*B*	*C*	*D*	*E*	*F*	*G*	*H*	*I*	*J*		*Z*

$$c_i = (p_i + k) \quad \% 26$$

k = 5 *encrypt cafe*

k = 2 *encrypt hide*

Doing it by Python:

plain text

0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	...	*25*
A	*B*	*C*	*D*	*E*	*F*	*G*	*H*	*I*	*J*		*Z*

$$c_i = (p_i + k) \quad \% 26$$

$k = 5$ *encrypt cafe*

```
>>> word = 'CAFE'
>>>
>>> ch = word[0]
>>> ch
'C'
>>> ord(ch)
67
>>> ord(word[1])
65
>>>
```

Doing it by Python:

plain text

0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*...*	*25*
A	*B*	*C*	*D*	*E*	*F*	*G*	*H*	*I*	*J*		*Z*

$$c_i = (p_i + k) \quad \% 26$$

k = 5 *encrypt cafe*

A is 65 but we want it to be 0
C is 67 but we want 2

```
>>> word = 'CAFE'
>>>
>>> ch = word[0]
>>> ch
'C'
>>> ord(ch)
67
>>> ord(word[1])
65
>>>
```

Doing it by Python:

plain text

0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*...*	*25*
A	*B*	*C*	*D*	*E*	*F*	*G*	*H*	*I*	*J*		*Z*

$$c_i = (p_i + k) \quad \% 26$$

$k = 5$ *encrypt cafe*

```
>>> word = 'CAFE'
>>>
>>> ch = word[0]
>>> ch
'C'
>>> ord(ch)
67
>>> ord(word[1])
65
>>>
```

A is 65 but we want it to be 0
C is 67 but we want 2

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	00	Null	32	20	Space	64	40	@	96	60	`
1	01	Start of heading	33	21	!	65	41	A	97	61	a
2	02	Start of text	34	22	"	66	42	B	98	62	b
3	03	End of text	35	23	#	67	43	C	99	63	c
4	04	End of transmit	36	24	$	68	44	D	100	64	d
5	05	Enquiry	37	25	%	69	45	E	101	65	e
6	06	Acknowledge	38	26	&	70	46	F	102	66	f
7	07	Audible bell	39	27	'	71	47	G	103	67	g
8	08	Backspace	40	28	(	72	48	H	104	68	h
9	09	Horizontal tab	41	29	)	73	49	I	105	69	i
10	0A	Line feed	42	2A	*	74	4A	J	106	6A	j
11	0B	Vertical tab	43	2B	+	75	4B	K	107	6B	k
12	0C	Form feed	44	2C	,	76	4C	L	108	6C	l
13	0D	Carriage return	45	2D	-	77	4D	M	109	6D	m
14	0E	Shift out	46	2E	.	78	4E	N	110	6E	n
15	0F	Shift in	47	2F	/	79	4F	O	111	6F	o
16	10	Data link escape	48	30	0	80	50	P	112	70	p
17	11	Device control 1	49	31	1	81	51	Q	113	71	q
18	12	Device control 2	50	32	2	82	52	R	114	72	r
19	13	Device control 3	51	33	3	83	53	S	115	73	s
20	14	Device control 4	52	34	4	84	54	T	116	74	t
21	15	Neg. acknowledge	53	35	5	85	55	U	117	75	u
22	16	Synchronous idle	54	36	6	86	56	V	118	76	v
23	17	End trans. block	55	37	7	87	57	W	119	77	w
24	18	Cancel	56	38	8	88	58	X	120	78	x
25	19	End of medium	57	39	9	89	59	Y	121	79	y
26	1A	Substitution	58	3A	:	90	5A	Z	122	7A	z
27	1B	Escape	59	3B	;	91	5B	[	123	7B	{
28	1C	File separator	60	3C	<	92	5C	\	124	7C	|
29	1D	Group separator	61	3D	=	93	5D	]	125	7D	}
30	1E	Record separator	62	3E	>	94	5E	^	126	7E	~
31	1F	Unit separator	63	3F	?	95	5F	_	127	7F	□

Doing it by Python:

plain text

0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*...*	*25*
A	*B*	*C*	*D*	*E*	*F*	*G*	*H*	*I*	*J*		*Z*

$$c_i = (p_i + k) \quad \% 26$$

$$c_i = (ord(word[i]) - 65 + k) \quad \% 26$$

```
>>> ord(word[0])
67
>>> ord(word[1])
65
>>> ord(word[2])
70
>>> ord(word[3])
69
>>>
```

It looks like we should subtract 65 from each letter.

Doing it by Python:

plain text

0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*...*	*25*
A	*B*	*C*	*D*	*E*	*F*	*G*	*H*	*I*	*J*		*Z*

for the cipher text we want characters

7 5 10 9

how do we turn these back to ascii?

```
>>> ord(word[0])
67
>>> ord(word[1])
65
>>> ord(word[2])
70
>>> ord(word[3])
69
>>>
```

Doing it by Python:

plain text

0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*...*	*25*
A	*B*	*C*	*D*	*E*	*F*	*G*	*H*	*I*	*J*		*Z*

for the cipher text we want characters

7 5 10 9

add 65:

72 70 75 74

```
>>> ord(word[0])
67
>>> ord(word[1])
65
>>> ord(word[2])
70
>>> ord(word[3])
69
>>>
```

Doing it by Python:

plain text	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	...	*25*
	A	*B*	*C*	*D*	*E*	*F*	*G*	*H*	*I*	*J*		*Z*

Finally, convert these ASCII numbers back to characters

72 70 75 74

```
>>> chr(72)
'H'
>>> chr(70)
'F'
>>> chr(75)
'K'
>>> chr(74)
'J'
```

Sweet!!

questions?

encrypt

```
def encrypt(plaintext, k):
```

encrypt

```
def encrypt(plaintext, k):
    result = ''
    # step 1: loop over each character
    # in the string
```

encrypt

```
def encrypt(plaintext, k):
    result = ''
    # step 1: loop over each character
    # in the string
    for ch in plaintext:
        result += ch
```

encrypt

```
def encrypt(plaintext, k):
    result = ''
    # step 1: loop over each character
    # in the string
    for ch in plaintext:
        result += ch
        # step 2: output each encoded
        # letter making sure not to
        #encode non letters.
```

encrypt

```
def encrypt(plaintext, k):
    result = ''
    # step 1: loop over each character
    # in the string
    for ch in plaintext:
        result += ch
        # step 2: output each encoded
        # letter making sure not to
        #encode non letters.
```

Should work for upper and lower case letters

$A = 65 \quad a = 97$

decrypt

```
def decrypt(ciphertext, k):
    result = ''
    # step 1: loop over each character
    # in the string
    for ch in plaintext:
        result += ch
```

decrypt

```
def decrypt(ciphertext, k):
    result = ''
    # step 1: loop over each character
    # in the string
    for ch in plaintext:
        result += ch
        # step 2: output each decoded
        # letter making sure not to
        # decode non letters.
```

Keep in mind

- *capitalization must be preserved*
- *letters should never become symbols*
- *symbols should not be changed*

Hacker edition

Vignère's Cipher

Vignère's Cipher

rotate each letter by a different amount

Just do encrypt

```
def vignere(plaintext, keyword):
```

vignere(“POODLE”, “DOG”)

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T

20	21	22	23	24	25
U	V	W	X	Y	Z

P	O	O	D	L	E

`vignere("POODLE", "DOG")`

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T

20	21	22	23	24	25
U	V	W	X	Y	Z

P	O	O	D	L	E
D	O	G	D	O	G

`vignere("POODLE", "DOG")`

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T

20	21	22	23	24	25
U	V	W	X	Y	Z

15	14	14	3	11	4
D	O	G	D	O	G

vignere(“POODLE”, “DOG”)

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T

20	21	22	23	24	25
U	V	W	X	Y	Z

15	14	14	3	11	4
3	14	6	3	14	6

`vignere("POODLE", "DOG")`

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T

20	21	22	23	24	25
U	V	W	X	Y	Z

(plain + key) % 26

15	14	14	3	11	4
3	14	6	3	14	6
18	2	20	6	25	10

`vignere("POODLE", "DOG")`

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T

20	21	22	23	24	25
U	V	W	X	Y	Z

(plain + key) % 26

15	14	14	3	11	4
3	14	6	3	14	6
18	2	20	6	25	10
S	C	U	G	Z	K

how to test your program

http://rosemary.umw.edu/~raz/vignere

Due

Sunday 11 November 11:59pm

submit.o.bot@gmail.com