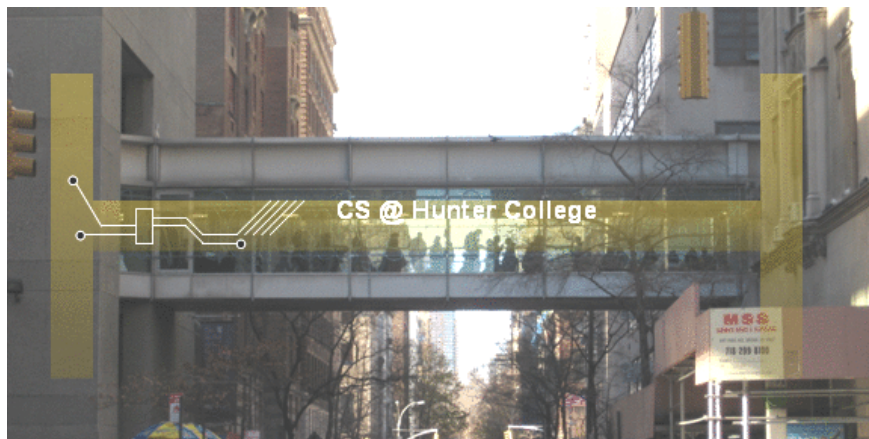


CSci 127: Introduction to Computer Science



hunter.cuny.edu/csci

Frequently Asked Questions

From email

Frequently Asked Questions

From email

- **I am not sure how to submit the Lab.**

Frequently Asked Questions

From email

- **I am not sure how to submit the Lab.**
You don't submit the lab, you read the lab.

Frequently Asked Questions

From email

- **I am not sure how to submit the Lab.**

*You don't submit the lab, you **read the lab**.*

When you are done, start working on this week's 5 programming assignments (this week we will be working on programs 6-10)

Frequently Asked Questions

From email

- **I am not sure how to submit the Lab.**

You don't submit the lab, you read the lab.

When you are done, start working on this week's 5 programming assignments (this week we will be working on programs 6-10)

- **Can I work ahead?**

Frequently Asked Questions

From email

- **I am not sure how to submit the Lab.**

You don't submit the lab, you read the lab.

When you are done, start working on this week's 5 programming assignments (this week we will be working on programs 6-10)

- **Can I work ahead?**

Absolutely! Submission is open on Gradescope, 3 weeks before the deadline.

Frequently Asked Questions

From email

- **I am not sure how to submit the Lab.**

You don't submit the lab, you read the lab.

When you are done, start working on this week's 5 programming assignments (this week we will be working on programs 6-10)

- **Can I work ahead?**

Absolutely! Submission is open on Gradescope, 3 weeks before the deadline.

IMPORTANT: Students who work on the due dates in this class tend to miss deadlines and fall behind. If, instead, you work on programs the week of the associated lecture, you will have time to ask for help if you get stuck and still make the deadline.

Frequently Asked Questions

From email

- **I am not sure how to submit the Lab.**

You don't submit the lab, you read the lab.

When you are done, start working on this week's 5 programming assignments (this week we will be working on programs 6-10)

- **Can I work ahead?**

Absolutely! Submission is open on Gradescope, 3 weeks before the deadline.

IMPORTANT: Students who work on the due dates in this class tend to miss deadlines and fall behind. If, instead, you work on programs the week of the associated lecture, you will have time to ask for help if you get stuck and still make the deadline.

- **When is the midterm?**

Frequently Asked Questions

From email

- **I am not sure how to submit the Lab.**

You don't submit the lab, you read the lab.

When you are done, start working on this week's 5 programming assignments (this week we will be working on programs 6-10)

- **Can I work ahead?**

Absolutely! Submission is open on Gradescope, 3 weeks before the deadline.

IMPORTANT: Students who work on the due dates in this class tend to miss deadlines and fall behind. If, instead, you work on programs the week of the associated lecture, you will have time to ask for help if you get stuck and still make the deadline.

- **When is the midterm?**

There is no midterm. Instead there's required weekly quizzes, code reviews and programming assignments.

Seat Number

- For contact tracing purposes, the College requests that you **remain in the same seat for the entire semester.**

Seat Number

- For contact tracing purposes, the College requests that you **remain in the same seat for the entire semester.**
- Please, **write down the row and seat number** you are seating in and continue to seat there for the rest of the semester.

Seat Number

- For contact tracing purposes, the College requests that you **remain in the same seat for the entire semester.**
- Please, **write down the row and seat number** you are seating in and continue to seat there for the rest of the semester.
- **Submit your row and seat number** using this form: bit.ly/127_seat_number

Seat Number

- For contact tracing purposes, the College requests that you **remain in the same seat for the entire semester.**
- Please, **write down the row and seat number** you are seating in and continue to seat there for the rest of the semester.
- **Submit your row and seat number** using this form: bit.ly/127_seat_number
- The link to the form can also be found on Blackboard under Announcements.

Today's Topics



- For-loops
- `range()`
- Variables
- Characters
- Strings
- Guests: Internships & Clubs

Today's Topics



- **For-loops**
- `range()`
- Variables
- Characters
- Strings
- Guests: Internships & Clubs

Group Work

Some review and some novel challenges:

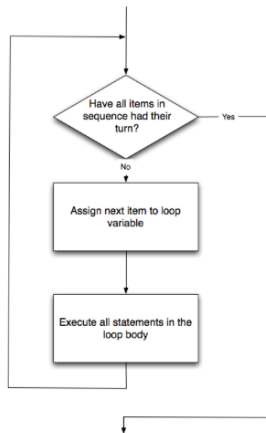
```
1 #Predict what will be printed:
2 for i in range(4):
3     print('The world turned upside down')
4 for j in [0,1,2,3,4,5]:
5     print(j)
6 for count in range(6):
7     print(count)
8 for color in ['red', 'green', 'blue']:
9     print(color)
10 for i in range(2):
11     for j in range(2):
12         print('Look around,')
13     print('How lucky we are to be alive!')
```

Python Tutor

```
1 #Predict what will be printed:
2 for i in range(4):
3     print('The world turned upside down')
4 for j in [0,1,2,3,4,5]:
5     print(j)
6 for count in range(6):
7     print(count)
8 for color in ['red', 'green', 'blue']:
9     print(color)
10 for i in range(2):
11     for j in range(2):
12         print('Look around,')
13     print('How lucky we are to be alive!')
```

(Demo with pythonTutor)

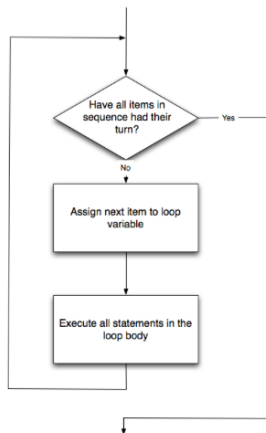
for-loop



```
for i in list:  
    statement1  
    statement2  
    statement3
```

How to Think Like CS, §4.5

for-loop



How to Think Like CS, §4.5

```
for i in list:  
    statement1  
    statement2  
    statement3
```

where `list` is a list of items:

- stated explicitly (e.g. `[1,2,3]`) or
- generated by a function, e.g. `range()`.

Today's Topics



- For-loops
- **range()**
- Variables
- Characters
- Strings
- Guests: Internships & Clubs

More on range():

```
1 #Predict what will be printed:
2
3 for num in [2,4,6,8,10]:
4     print(num)
5
6 sum = 0
7 for x in range(0,12,2):
8     print(x)
9     sum = sum + x
10
11 print(sum)
12
13 for c in "ABCD":
14     print(c)
```

Python Tutor

```
1 #Predict what will be printed:
2
3 for num in [2,4,6,8,10]:
4     print(num)
5
6 sum = 0
7 for x in range(0,12,2):
8     print(x)
9     sum = sum + x
10
11 print(sum)
12
13 for c in "ABCD":
14     print(c)
```

(Demo with pythonTutor)

range()

Simplest version:

- `range(stop)`



range()



Simplest version:

- `range(stop)`
- Produces a list: `[0,1,2,3,...,stop-1]`

range()



Simplest version:

- `range(stop)`
- Produces a list: `[0,1,2,3,...,stop-1]`
- For example, if you want the the list `[0,1,2,3,...,100]`, you would write:

range()



Simplest version:

- `range(stop)`
- Produces a list: `[0,1,2,3,...,stop-1]`
- For example, if you want the the list `[0,1,2,3,...,100]`, you would write:

`range(101)`

`range()`

What if you wanted to start somewhere else:



range()

What if you wanted to start somewhere else:

- `range(start, stop)`



range()



What if you wanted to start somewhere else:

- `range(start, stop)`
- Produces a list:
`[start, start+1, ..., stop-1]`

range()



What if you wanted to start somewhere else:

- `range(start, stop)`
- Produces a list:
`[start, start+1, ..., stop-1]`
- For example, if you want the the list
`[10, 11, ..., 20]`
you would write:

range()



What if you wanted to start somewhere else:

- `range(start, stop)`
- Produces a list:
`[start, start+1, ..., stop-1]`
- For example, if you want the the list
`[10, 11, ..., 20]`
you would write:

```
range(10, 21)
```

range()

What if you wanted to count by twos, or some other number:



range()

What if you wanted to count by twos, or some other number:

- `range(start, stop, step)`



range()

What if you wanted to count by twos, or some other number:

- `range(start, stop, step)`
- Produces a list:
`[start, start+step, start+2*step..., last]`
(where last is the largest $\text{start} + k * \text{step}$ less than stop)



range()

What if you wanted to count by twos, or some other number:

- `range(start, stop, step)`
- Produces a list:
`[start, start+step, start+2*step..., last]`
(where last is the largest $\text{start} + k * \text{step}$ less than stop)
- For example, if you want the the list `[5, 10, ..., 50]` you would write:



range()



What if you wanted to count by twos, or some other number:

- `range(start, stop, step)`
- Produces a list:
`[start, start+step, start+2*step..., last]`
(where last is the largest $\text{start} + k * \text{step}$ less than stop)
- For example, if you want the the list `[5, 10, ..., 50]` you would write:

```
range(5, 51, 5)
```

In summary: `range()`



The three versions:

In summary: `range()`



The three versions:

- `range(stop)`

In summary: `range()`



The three versions:

- `range(stop)`
- `range(start, stop)`

In summary: `range()`



The three versions:

- `range(stop)`
- `range(start, stop)`
- `range(start, stop, step)`

Today's Topics



- For-loops
- `range()`
- **Variables**
- Characters
- Strings
- Guests: Internships & Clubs

Variables

- A **variable** is a reserved memory location for storing a value.



Variables

- A **variable** is a reserved memory location for storing a value.
- Different kinds, or **types**, of values need different amounts of space:
 - ▶ **int**: integer or whole numbers



Variables

- A **variable** is a reserved memory location for storing a value.
- Different kinds, or **types**, of values need different amounts of space:
 - ▶ **int**: integer or whole numbers
 - ▶ **float**: floating point or real numbers



Variables

- A **variable** is a reserved memory location for storing a value.
- Different kinds, or **types**, of values need different amounts of space:
 - ▶ **int**: integer or whole numbers
 - ▶ **float**: floating point or real numbers
 - ▶ **string**: sequence of characters



Variables

- A **variable** is a reserved memory location for storing a value.
- Different kinds, or **types**, of values need different amounts of space:
 - ▶ **int**: integer or whole numbers
 - ▶ **float**: floating point or real numbers
 - ▶ **string**: sequence of characters
 - ▶ **list**: a sequence of items



Variables



- A **variable** is a reserved memory location for storing a value.
- Different kinds, or **types**, of values need different amounts of space:
 - ▶ **int**: integer or whole numbers
 - ▶ **float**: floating point or real numbers
 - ▶ **string**: sequence of characters
 - ▶ **list**: a sequence of items
e.g. [3, 1, 4, 5, 9] or
['violet', 'purple', 'indigo']

Variables



- A **variable** is a reserved memory location for storing a value.
- Different kinds, or **types**, of values need different amounts of space:
 - ▶ **int**: integer or whole numbers
 - ▶ **float**: floating point or real numbers
 - ▶ **string**: sequence of characters
 - ▶ **list**: a sequence of items
e.g. [3, 1, 4, 5, 9] or
['violet', 'purple', 'indigo']
 - ▶ **class variables**: for complex objects, like turtles.
- In Python (unlike other languages) you don't need to specify the type; it is deduced by its value.

Variable Names

- There's some rules about valid names for variables.



Variable Names



- There's some rules about valid names for variables.
- Can use the underscore ('_'), upper and lower case letters.

Variable Names



- There's some rules about valid names for variables.
- Can use the underscore ('_'), upper and lower case letters.
- Can also use numbers, just can't start a name with a number.

Variable Names



- There's some rules about valid names for variables.
- Can use the underscore ('_'), upper and lower case letters.
- Can also use numbers, just can't start a name with a number.
- Can't use symbols (like '+' or '*') since used for arithmetic.

Variable Names



- There's some rules about valid names for variables.
- Can use the underscore ('_'), upper and lower case letters.
- Can also use numbers, just can't start a name with a number.
- Can't use symbols (like '+' or '*') since used for arithmetic.
- Can't use some words that Python has reserved for itself (e.g. `for`).
(List of reserved words in *Think CS*, §2.5.)

Today's Topics



- For-loops
- `range()`
- Variables
- **Characters**
- Strings
- Guests: Internships & Clubs

Standardized Code for Characters

American Standard Code for Information Interchange (ASCII), 1960.

Standardized Code for Characters

American Standard Code for Information Interchange (ASCII), 1960.
(New version called: Unicode).

Standardized Code for Characters

American Standard Code for Information Interchange (ASCII), 1960.
(New version called: Unicode).

ASCII TABLE

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	\$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[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	[NEGATIVE 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	[ENG OF 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	[SUBSTITUTE]	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	[DEL]

(wiki)

Converting from Character to Code:

(There is a link to the ASCII table on the course webpage, under 'Useful Links'.)

ASCII TABLE

Decimal	Hex Char	Decimal	Hex Char	Decimal	Hex Char	Decimal	Hex Char
0		16	P	32	@	48	0
1	SOH	17	Q	33	A	49	1
2	STX	18	R	34	B	50	2
3	ETX	19	S	35	C	51	3
4	END	20	T	36	D	52	4
5	SO	21	U	37	E	53	5
6	SI	22	V	38	F	54	6
7	DEL	23	W	39	G	55	7
8		24	X	40	H	56	8
9		25	Y	41	I	57	9
10	LF	26	Z	42	J	58	:
11		27	[43	K	59	;
12	FF	28	\	44	L	60	<
13	CR	29]	45	M	61	=
14		30	^	46	N	62	>
15		31	_	47	O	63	?
16	P	32	@	48	0	64	SP
17	Q	33	A	49	1	65	a
18	R	34	B	50	2	66	b
19	S	35	C	51	3	67	c
20	T	36	D	52	4	68	d
21	U	37	E	53	5	69	e
22	V	38	F	54	6	70	f
23	W	39	G	55	7	71	g
24	X	40	H	56	8	72	h
25	Y	41	I	57	9	73	i
26	Z	42	J	58	:	74	j
27	[43	K	59	;	75	k
28	\	44	L	60	<	76	l
29]	45	M	61	=	77	m
30	^	46	N	62	>	78	n
31	_	47	O	63	?	79	o
32	@	48	0	64	SP	80	p
33	A	49	1	65	a	81	q
34	B	50	2	66	b	82	r
35	C	51	3	67	c	83	s
36	D	52	4	68	d	84	t
37	E	53	5	69	e	85	u
38	F	54	6	70	f	86	v
39	G	55	7	71	g	87	w
40	H	56	8	72	h	88	x
41	I	57	9	73	i	89	y
42	J	58	:	74	j	90	z
43	K	59	;	75	k	91	{
44	L	60	<	76	l	92	
45	M	61	=	77	m	93	}
46	N	62	>	78	n	94	~
47	O	63	?	79	o	95	
48	0	64	SP	80	p	96	
49	1	65	a	81	q	97	
50	2	66	b	82	r	98	
51	3	67	c	83	s	99	
52	4	68	d	84	t	100	
53	5	69	e	85	u	101	
54	6	70	f	86	v	102	
55	7	71	g	87	w	103	
56	8	72	h	88	x	104	
57	9	73	i	89	y	105	
58	:	74	j	90	z	106	
59	;	75	k	91	{	107	
60	<	76	l	92		108	
61	=	77	m	93	}	109	
62	>	78	n	94	~	110	
63	?	79	o	95		111	
64	SP	80	p	96		112	
65	a	81	q	97		113	
66	b	82	r	98		114	
67	c	83	s	99		115	
68	d	84	t	100		116	
69	e	85	u	101		117	
70	f	86	v	102		118	
71	g	87	w	103		119	
72	h	88	x	104		120	
73	i	89	y	105		121	
74	j	90	z	106		122	
75	k	91	{	107		123	
76	l	92		108		124	
77	m	93	}	109		125	
78	n	94	~	110		126	
79	o	95		111		127	DEL

Converting from Character to Code:

(There is a link to the ASCII table on the course webpage, under 'Useful Links'.)

- `ord(c)`: returns Unicode (ASCII) of the character.

ASCII TABLE

Decimal	Hex Char	Decimal	Hex Char	Decimal	Hex Char	Decimal	Hex Char
0		16		32		48	
1		17		33	!	49	1
2		18		34	"	50	2
3		19		35	#	51	3
4		20		36	\$	52	4
5		21		37	%	53	5
6		22		38	&	54	6
7		23		39	'	55	7
8		24		40	(56	8
9		25		41)	57	9
10		26		42	*	58	:
11		27		43	+	59	;
12		28		44	,	60	<
13		29		45	-	61	=
14		30		46	.	62	>
15		31		47	/	63	?
16		32	!	64	@	80	P
17		33	"	65	A	81	Q
18		34	"	66	B	82	R
19		35	#	67	C	83	S
20		36	\$	68	D	84	T
21		37	%	69	E	85	U
22		38	&	70	F	86	V
23		39	'	71	G	87	W
24		40	(72	H	88	X
25		41)	73	I	89	Y
26		42	*	74	J	90	Z
27		43	+	75	K	91	[
28		44	,	76	L	92	\
29		45	-	77	M	93]
30		46	.	78	N	94	^
31		47	/	79	O	95	_
32	!	48	0	80	P	96	`
33	"	49	1	81	Q	97	a
34	"	50	2	82	R	98	b
35	#	51	3	83	S	99	c
36	\$	52	4	84	T	100	d
37	%	53	5	85	U	101	e
38	&	54	6	86	V	102	f
39	'	55	7	87	W	103	g
40	(56	8	88	X	104	h
41)	57	9	89	Y	105	i
42	*	58	:	90	Z	106	j
43	+	59	;	91	[107	k
44	,	60	<	92	\	108	l
45	-	61	=	93]	109	m
46	.	62	>	94	^	110	n
47	/	63	?	95	_	111	o
48	0	64	@	96	`	112	p
49	1	65	A	97	a	113	q
50	2	66	B	98	b	114	r
51	3	67	C	99	c	115	s
52	4	70	F	100	d	116	t
53	5	71	G	101	e	117	u
54	6	72	H	102	f	118	v
55	7	73	I	103	g	119	w
56	8	74	J	104	h	120	x
57	9	75	K	105	i	121	y
58	:	76	L	106	j	122	z
59	;	77	M	107	k	123	{
60	<	78	N	108	l	124	
61	=	79	O	109	m	125	}
62	>	80	P	110	n	126	~
63	?	81	Q	111	o	127	
64	@	82	R	112	p		
65	A	83	S	113	q		
66	B	84	T	114	r		
67	C	85	U	115	s		
68	D	86	V	116	t		
69	E	87	W	117	u		
70	F	88	X	118	v		
71	G	89	Y	119	w		
72	H	90	Z	120	x		
73	I	91	[121	y		
74	J	92	\	122	z		
75	K	93]				
76	L	94	^				
77	M	95	_				
78	N	96	`				
79	O	97	a				
80	P	98	b				
81	Q	99	c				
82	R	100	d				
83	S	101	e				
84	T	102	f				
85	U	103	g				
86	V	104	h				
87	W	105	i				
88	X	106	j				
89	Y	107	k				
90	Z	108	l				
91	[109	m				
92	\	110	n				
93]	111	o				
94	^	112	p				
95	_	113	q				
96	`	114	r				
97	a	115	s				
98	b	116	t				
99	c	117	u				
100	d	118	v				
101	e	119	w				
102	f	120	x				
103	g	121	y				
104	h	122	z				
105	i						
106	j						
107	k						
108	l						
109	m						
110	n						
111	o						
112	p						
113	q						
114	r						
115	s						
116	t						
117	u						
118	v						
119	w						
120	x						
121	y						
122	z						
123	{						
124							
125	}						
126	~						
127							

Converting from Character to Code:

(There is a link to the ASCII table on the course webpage, under 'Useful Links'.)

ASCII TABLE

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	00		16	10	P	32	20	R	48	30	T
1	01	SOH	17	11	Q	33	21	S	49	31	U
2	02	STX	18	12	R	34	22	T	50	32	V
3	03	ETX	19	13	S	35	23	U	51	33	W
4	04	END	20	14	T	36	24	V	52	34	X
5	05	SO	21	15	U	37	25	W	53	35	Y
6	06	SI	22	16	V	38	26	X	54	36	Z
7	07	BS	23	17	W	39	27	Y	55	37	[
8	08	HT	24	18	X	40	28	Z	56	38	\
9	09	LF	25	19	Y	41	29	[57	39]
10	0A	VT	26	1A	Z	42	2A	\	58	3A	^
11	0B	FF	27	1B	[43	2B]	59	3B	_
12	0C	DEL	28	1C	\	44	2C	^	60	3C	`
13	0D		29	1D]	45	2D	_	61	3D	a
14	0E		30	1E	^	46	2E	`	62	3E	b
15	0F		31	1F	_	47	2F	a	63	3F	c
16	10	SP	32	20	R	48	30	T	64	40	d
17	11	P	33	21	S	49	31	U	65	41	e
18	12	Q	34	22	T	50	32	V	66	42	f
19	13	R	35	23	U	51	33	W	67	43	g
20	14	S	36	24	V	52	34	X	68	44	h
21	15	T	37	25	W	53	35	Y	69	45	i
22	16	U	38	26	X	54	36	Z	70	46	j
23	17	V	39	27	Y	55	37	[71	47	k
24	18	W	40	28	Z	56	38	\	72	48	l
25	19	X	41	29	[57	39]	73	49	m
26	1A	Z	42	2A	\	58	3A	^	74	4A	n
27	1B	[43	2B]	59	3B	_	75	4B	o
28	1C	\	44	2C	^	60	3C	`	76	4C	p
29	1D]	45	2D	_	61	3D	a	77	4D	q
30	1E	^	46	2E	`	62	3E	b	78	4E	r
31	1F	_	47	2F	a	63	3F	c	79	4F	s
32	20	R	48	30	T	64	40	d	80	50	t
33	21	S	49	31	U	65	41	e	81	51	u
34	22	T	50	32	V	66	42	f	82	52	v
35	23	U	51	33	W	67	43	g	83	53	w
36	24	V	52	34	X	68	44	h	84	54	x
37	25	W	53	35	Y	69	45	i	85	55	y
38	26	X	54	36	Z	70	46	j	86	56	z
39	27	Y	55	37	[71	47	k	87	57	{
40	28	Z	56	38	\	72	48	l	88	58	}
41	29	[57	39]	73	49	m	89	59	~
42	2A	\	58	3A	^	90	5A		91	5B	
43	2B]	59	3B	_	92	5C		92	5C	
44	2C	^	60	3C	`	93	5D		93	5D	
45	2D	_	61	3D	a	94	5E		94	5E	
46	2E	`	62	3E	b	95	5F		95	5F	
47	2F	a	63	3F	c	96	60		96	60	
48	30	T	64	40	d	97	61		97	61	
49	31	U	65	41	e	98	62		98	62	
50	32	V	66	42	f	99	63		99	63	
51	33	W	67	43	g	100	64		100	64	
52	34	X	68	44	h	101	65		101	65	
53	35	Y	69	45	i	102	66		102	66	
54	36	Z	70	46	j	103	67		103	67	
55	37	[71	47	k	104	68		104	68	
56	38	\	72	48	l	105	69		105	69	
57	39]	73	49	m	106	6A		106	6A	
58	3A	^	74	4A	n	107	6B		107	6B	
59	3B	_	75	4B	o	108	6C		108	6C	
60	3C	`	76	4C	p	109	6D		109	6D	
61	3D	a	77	4D	q	110	6E		110	6E	
62	3E	b	78	4E	r	111	6F		111	6F	
63	3F	c	79	4F	s	112	70		112	70	
64	40	d	80	50	t	113	71		113	71	
65	41	e	81	51	u	114	72		114	72	
66	42	f	82	52	v	115	73		115	73	
67	43	g	83	53	w	116	74		116	74	
68	44	h	84	54	x	117	75		117	75	
69	45	i	85	55	y	118	76		118	76	
70	46	j	86	56	z	119	77		119	77	
71	47	k	87	57	{	120	78		120	78	
72	48	l	88	58	}	121	79		121	79	
73	49	m	89	59	~	122	7A		122	7A	
74	4A	n	90	5A		123	7B		123	7B	
75	4B	o	91	5B		124	7C		124	7C	
76	4C	p	92	5C		125	7D		125	7D	
77	4D	q	93	5D		126	7E		126	7E	
78	4E	r	94	5E		127	7F		127	7F	
79	4F	s	95	5F		128	80		128	80	
80	50	t	96	60		129	81		129	81	
81	51	u	97	61		130	82		130	82	
82	52	v	98	62		131	83		131	83	
83	53	w	99	63		132	84		132	84	
84	54	x	100	64		133	85		133	85	
85	55	y	101	65		134	86		134	86	
86	56	z	102	66		135	87		135	87	
87	57	{	103	67		136	88		136	88	
88	58	}	104	68		137	89		137	89	
89	59	~	105	69		138	8A		138	8A	
90	5A		106	6A		139	8B		139	8B	
91	5B		107	6B		140	8C		140	8C	
92	5C		108	6C		141	8D		141	8D	
93	5D		109	6D		142	8E		142	8E	
94	5E		110	6E		143	8F		143	8F	
95	5F		111	6F		144	90		144	90	
96	60		112	70		145	91		145	91	
97	61		113	71		146	92		146	92	
98	62		114	72		147	93		147	93	
99	63		115	73		148	94		148	94	
100	64		116	74		149	95		149	95	
101	65		117	75		150	96		150	96	
102	66		118	76		151	97		151	97	
103	67		119	77		152	98		152	98	
104	68		120	78		153	99		153	99	
105	69		121	79		154	9A		154	9A	
106	6A		122	7A		155	9B		155	9B	
107	6B		123	7B		156	9C		156	9C	
108	6C		124	7C		157	9D		157	9D	
109	6D		125	7D		158	9E		158	9E	
110	6E		126	7E		159	9F		159	9F	
111	6F		127	7F		160	A0		160	A0	
112	70		128	80		161	A1		161	A1	
113	71		129	81		162	A2		162	A2	
114	72		130	82		163	A3		163	A3	
115	73		131	83		164	A4		164	A4	
116	74		132	84		165	A5		165	A5	
117	75		133	85		166	A6		166	A6	
118	76		134	86		167	A7		167	A7	
119	77		135	87		168	A8		168	A8	
120	78		136	88		169	A9		169	A9	
121	79		137	89		170	AA		170	AA	
122	7A		138	8A		171	AB		171	AB	
123	7B		139	8B		172	AC		172	AC	
124	7C		140	8C		173	AD		173	AD	
125	7D		141	8D		174	AE		174	AE	
126	7E		142	8E		175	AF		175	AF	
127	7F		143	8F		176	B0		176	B0	
128	80		144	90		177	B1		177	B1	
129	81		145	91		178	B2		178	B2	
130	82		146	92		179	B3		179	B3	
131	83		147	93		180	B4		180	B4	
132	84		148	94		181	B5		181	B5	
133	85		149	95		182	B6		182	B6	
134	86		150	96		183	B7		183	B7	
135	87		151	97		184	B8		184	B8	
136	88		152	98		185	B9		185	B9	
137	89		153	99		186	BA		186	BA	
138	8A		154	9A		187	BB		187	BB	
139	8B		155	9B		188	BC		188	BC	
140	8C		156	9C		189	BD		189	BD	
141	8D		157	9D		190	BE		190	BE	
142	8E		158	9E		191	BF		191	BF	
143	8F		159	9F		192	C0		192	C0	
144	90		160	A0		193	C1		193	C1	
145	91		161	A1		194	C2		194	C2	
146	92		162	A2		195	C3		195	C3	
147	93		163	A3		196	C4		196	C4	
148	94		164	A4		197	C5		197	C5	
149	95		165	A5		198	C6		198	C6	
150	96		166	A6		199	C7		199	C7	
151	97		167	A7		200	C8		200	C8	
152	98		168	A8		201	C9				

Converting from Character to Code:

(There is a link to the ASCII table on the course webpage, under 'Useful Links'.)

ASCII TABLE

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	00		16	10	P	32	20	[48	30	0
1	01		17	11	Q	33	21	\	49	31	1
2	02		18	12	R	34	22]	50	32	2
3	03		19	13	S	35	23	^	51	33	3
4	04		20	14	T	36	24	_	52	34	4
5	05		21	15	U	37	25	`	53	35	5
6	06		22	16	V	38	26	{	54	36	6
7	07		23	17	W	39	27		55	37	7
8	08		24	18	X	40	28	}	56	38	8
9	09		25	19	Y	41	29	~	57	39	9
10	0A		26	1A	Z	42	2A		58	3A	.
11	0B		27	1B	[43	2B		59	3B	,
12	0C		28	1C	\	44	2C		60	3C	<
13	0D		29	1D]	45	2D		61	3D	=
14	0E		30	1E	^	46	2E		62	3E	>
15	0F		31	1F	_	47	2F		63	3F	?
16	10	@	32	20	[48	30	0	64	40	@
17	11	A	33	21	\	49	31	1	65	41	A
18	12	B	34	22]	50	32	2	66	42	B
19	13	C	35	23	^	51	33	3	67	43	C
20	14	D	36	24	_	52	34	4	68	44	D
21	15	E	37	25	`	53	35	5	69	45	E
22	16	F	38	26	{	54	36	6	70	46	F
23	17	G	39	27		55	37	7	71	47	G
24	18	H	40	28	}	56	38	8	72	48	H
25	19	I	41	29	~	57	39	9	73	49	I
26	1A	J	42	2A		58	3A	.	74	4A	J
27	1B	K	43	2B		59	3B	,	75	4B	K
28	1C	L	44	2C		60	3C	<	76	4C	L
29	1D	M	45	2D		61	3D	=	77	4D	M
30	1E	N	46	2E		62	3E	>	78	4E	N
31	1F	O	47	2F		63	3F	?	79	4F	O
32	20	P	48	30	0	64	40	@	80	50	P
33	21	Q	49	31	1	65	41	A	81	51	Q
34	22	R	50	32	2	66	42	B	82	52	R
35	23	S	51	33	3	67	43	C	83	53	S
36	24	T	52	34	4	68	44	D	84	54	T
37	25	U	53	35	5	69	45	E	85	55	U
38	26	V	54	36	6	70	46	F	86	56	V
39	27	W	55	37	7	71	47	G	87	57	W
40	28	X	56	38	8	72	48	H	88	58	X
41	29	Y	57	39	9	73	49	I	89	59	Y
42	2A	Z	58	3A	.	74	4A	J	90	5A	Z
43	2B	[59	3B	,	75	4B	K			
44	2C	\	60	3C	<	76	4C	L			
45	2D]	61	3D	=	77	4D	M			
46	2E	^	62	3E	>	78	4E	N			
47	2F	_	63	3F	?	79	4F	O			
48	30	0	64	40	@	80	50	P			
49	31	1	65	41	A	81	51	Q			
50	32	2	66	42	B	82	52	R			
51	33	3	67	43	C	83	53	S			
52	34	4	68	44	D	84	54	T			
53	35	5	69	45	E	85	55	U			
54	36	6	70	46	F	86	56	V			
55	37	7	71	47	G	87	57	W			
56	38	8	72	48	H	88	58	X			
57	39	9	73	49	I	89	59	Y			
58	3A	.	74	4A	J	90	5A	Z			
59	3B	,	75	4B	K						
60	3C	<	76	4C	L						
61	3D	=	77	4D	M						
62	3E	>	78	4E	N						
63	3F	?	79	4F	O						
64	40	@	80	50	P						
65	41	A	81	51	Q						
66	42	B	82	52	R						
67	43	C	83	53	S						
68	44	D	84	54	T						
69	45	E	85	55	U						
70	46	F	86	56	V						
71	47	G	87	57	W						
72	48	H	88	58	X						
73	49	I	89	59	Y						
74	4A	J	90	5A	Z						
75	4B	K									
76	4C	L									
77	4D	M									
78	4E	N									
79	4F	O									
80	50	P									
81	51	Q									
82	52	R									
83	53	S									
84	54	T									
85	55	U									
86	56	V									
87	57	W									
88	58	X									
89	59	Y									
90	5A	Z									
91	5B	[
92	5C	\									
93	5D]									
94	5E	^									
95	5F	_									
96	60	`									
97	61	a									
98	62	b									
99	63	c									
100	64	d									
101	65	e									
102	66	f									
103	67	g									
104	68	h									
105	69	i									
106	6A	j									
107	6B	k									
108	6C	l									
109	6D	m									
110	6E	n									
111	6F	o									
112	70	p									
113	71	q									
114	72	r									
115	73	s									
116	74	t									
117	75	u									
118	76	v									
119	77	w									
120	78	x									
121	79	y									
122	7A	z									
123	7B	{									
124	7C										
125	7D	}									
126	7E	~									
127	7F										

- `ord(c)`: returns Unicode (ASCII) of the character.
- Example: `ord('a')` returns 97.
- `chr(x)`: returns the character whose Unicode is x.

Converting from Character to Code:

(There is a link to the ASCII table on the course webpage, under 'Useful Links'.)

ASCII TABLE

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	00		16	10	P	32	20	[48	30	0
1	01		17	11	Q	33	21	\	49	31	1
2	02		18	12	R	34	22]	50	32	2
3	03		19	13	S	35	23	^	51	33	3
4	04		20	14	T	36	24	_	52	34	4
5	05		21	15	U	37	25	`	53	35	5
6	06		22	16	V	38	26	{	54	36	6
7	07		23	17	W	39	27		55	37	7
8	08		24	18	X	40	28	~	56	38	8
9	09		25	19	Y	41	29		57	39	9
10	0A		26	1A	Z	42	2A		58	3A	.
11	0B		27	1B	[43	2B		59	3B	,
12	0C		28	1C	\	44	2C		60	3C	;
13	0D		29	1D]	45	2D		61	3D	'
14	0E		30	1E	^	46	2E		62	3E	"
15	0F		31	1F	_	47	2F		63	3F	!
16	10	@	32	20	[48	30	0	64	40	+
17	11	A	33	21	\	49	31	1	65	41	*
18	12	B	34	22]	50	32	2	66	42	-
19	13	C	35	23	^	51	33	3	67	43	=
20	14	D	36	24	_	52	34	4	68	44	&
21	15	E	37	25	`	53	35	5	69	45	%
22	16	F	38	26	{	54	36	6	70	46	@
23	17		39	27		55	37	7	71	47	#
24	18		40	28	~	56	38	8	72	48	\$
25	19		41	29		57	39	9	73	49	%
26	1A		42	2A		58	3A	.	74	4A	&
27	1B		43	2B		59	3B	,	75	4B	'
28	1C		44	2C		60	3C	;	76	4C	"
29	1D		45	2D		61	3D	'	77	4D	!
30	1E		46	2E		62	3E	"	78	4E	+
31	1F		47	2F		63	3F	!	79	4F	*
32	20		48	30	0	64	40	+	80	50	^
33	21	!	49	31	1	65	41	*	81	51	_
34	22	"	50	32	2	66	42	-	82	52	~
35	23	"	51	33	3	67	43	=	83	53	
36	24	\$	52	34	4	68	44	&	84	54	
37	25	%	53	35	5	69	45	%	85	55	
38	26	&	54	36	6	70	46	@	86	56	
39	27	'	55	37	7	71	47	#	87	57	
40	28	(56	38	8	72	48	\$	88	58	
41	29)	57	39	9	73	49	%	89	59	
42	2A	*	58	3A	.	74	4A	&	90	5A	
43	2B	+	59	3B	,	75	4B	'	91	5B	
44	2C	,	60	3C	;	76	4C	"	92	5C	
45	2D	-	61	3D	'	77	4D	!	93	5D	
46	2E	.	62	3E	"	78	4E	+	94	5E	
47	2F	/	63	3F	!	79	4F	*	95	5F	
48	30	0	64	40	+	80	50	^	96	60	
49	31	1	65	41	*	81	51	_	97	61	
50	32	2	66	42	-	82	52	~	98	62	
51	33	3	67	43	=	83	53		99	63	
52	34	4	68	44	&	84	54		100	64	
53	35	5	69	45	%	85	55				
54	36	6	70	46	@	86	56				
55	37	7	71	47	#	87	57				
56	38	8	72	48	\$	88	58				
57	39	9	73	49	%	89	59				
58	3A	.	74	4A	&	90	5A				
59	3B	,	75	4B	'	91	5B				
60	3C	;	76	4C	"	92	5C				
61	3D	'	77	4D	!	93	5D				
62	3E	"	78	4E	+	94	5E				
63	3F	!	79	4F	*	95	5F				
64	40	+	80	50	^	96	60				
65	41	*	81	51	_	97	61				
66	42	-	82	52	~	98	62				
67	43	=	83	53		99	63				
68	44	&	84	54							
69	45	%	85	55							
70	46	@	86	56							
71	47	#	87	57							
72	48	\$	88	58							
73	49	%	89	59							
74	4A	&	90	5A							
75	4B	'	91	5B							
76	4C	"	92	5C							
77	4D	!	93	5D							
78	4E	+	94	5E							
79	4F	*	95	5F							
80	50	^	96	60							
81	51	_	97	61							
82	52	~	98	62							
83	53		99	63							
84	54										
85	55										
86	56										
87	57										
88	58										
89	59										
90	5A										
91	5B										
92	5C										
93	5D										
94	5E										
95	5F										
96	60										
97	61										
98	62										
99	63										
100	64										

- `ord(c)`: returns Unicode (ASCII) of the character.
- Example: `ord('a')` returns 97.
- `chr(x)`: returns the character whose Unicode is x.
- Example: `chr(97)` returns 'a'.

Converting from Character to Code:

(There is a link to the ASCII table on the course webpage, under 'Useful Links'.)

ASCII TABLE

Decimal	Hex Char	Decimal	Hex Char	Decimal	Hex Char	Decimal	Hex Char
0		16	P	32	@	48	0
1	SOH	17	Q	33	A	49	1
2	STX	18	R	34	B	50	2
3	ETX	19	S	35	C	51	3
4	END	20	T	36	D	52	4
5	SO	21	U	37	E	53	5
6	ST	22	V	38	F	54	6
7	HT	23	W	39	G	55	7
8	LF	24	X	40	H	56	8
9	VT	25	Y	41	I	57	9
10	FF	26	Z	42	J	58	:
11		27	[43	K	59	;
12		28	\	44	L	60	<
13		29]	45	M	61	=
14		30	^	46	N	62	>
15		31	_	47	O	63	?

- `ord(c)`: returns Unicode (ASCII) of the character.
- Example: `ord('a')` returns 97.
- `chr(x)`: returns the character whose Unicode is x.
- Example: `chr(97)` returns 'a'.
- What is `chr(33)`?

In Pairs or Triples...

Some review and some novel challenges:

```
1 #Predict what will be printed:
2
3 for c in range(65,90):
4     print(chr(c))
5
6 message = "I love Python"
7 newMessage = ""
8 for c in message:
9     print(ord(c))    #Print the Unicode of each number
10    print(chr(ord(c)+1))    #Print the next character
11    newMessage = newMessage + chr(ord(c)+1) #add to the new message
12 print("The coded message is", newMessage)
13
14 word = "zebra"
15 codedWord = ""
16 for ch in word:
17     offset = ord(ch) - ord('a') + 1 #how many letters past 'a'
18     wrap = offset % 26 #if larger than 26, wrap back to 0
19     newChar = chr(ord('a') + wrap) #compute the new letter
20     print(wrap, chr(ord('a') + wrap))    #print the wrap & new lett
21     codedWord = codedWord + newChar #add the newChar to the coded w
22
23 print("The coded word (with wrap) is", codedWord)
```

Python Tutor

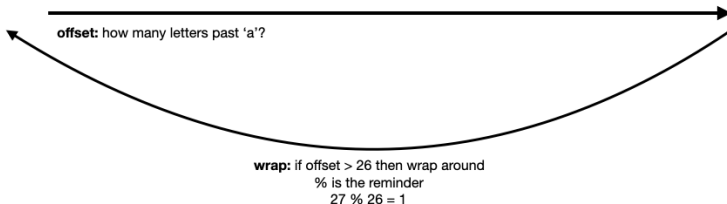
```
1 #Predict what will be printed:
2
3 for c in range(65,90):
4     print(chr(c))
5
6 message = "I love Python"
7 newMessage = ""
8 for c in message:
9     print(ord(c))    #Print the Unicode of each number
10    print(chr(ord(c)+1))    #Print the next character
11    newMessage = newMessage + chr(ord(c)+1) #Add to the new message
12 print("The coded message is", newMessage)
13
14 word = "zebra"
15 codedWord = ""
16 for ch in word:
17     offset = ord(ch) - ord('a') + 1 #How many letters past 'a'
18     wrap = offset % 26 #if larger than 26, wrap back to 0
19     newChar = chr(ord('a') + wrap) #compute the new letter
20     print(wrap, chr(ord('a') + wrap))    #Print the wrap & new lett
21     codedWord = codedWord + newChar #add the newChar to the coded w
22
23 print("The coded word (with wrap) is", codedWord)
```

(Demo with pythonTutor)

Wrap



<code>chr()</code>	a	b	c				...				x	y	z
<code>ord()</code>	97	98	99				...				120	121	122



User Input

Covered in detail in Lab 2:

```
➔ 1 mess = input('Please enter a message: ')\n   2 print("You entered", mess)
```

(Demo with pythonTutor)

Side Note: '+' for numbers and strings



- `x = 3 + 5` stores the number 8 in memory location `x`.

Side Note: '+' for numbers and strings



- $x = 3 + 5$ stores the number 8 in memory location x .
- $x = x + 1$ increases x by 1.

Side Note: '+' for numbers and strings



- `x = 3 + 5` stores the number 8 in memory location `x`.
- `x = x + 1` increases `x` by 1.
- `s = "hi" + "Mom"` stores "hiMom" in memory locations `s`.

Side Note: '+' for numbers and strings



- `x = 3 + 5` stores the number 8 in memory location `x`.
- `x = x + 1` increases `x` by 1.
- `s = "hi" + "Mom"` stores "hiMom" in memory locations `s`.
- `s = s + "A"` adds the letter "A" to the end of the strings `s`.

Today's Topics



- For-loops
- `range()`
- Variables
- Characters
- **Strings**
- Guests: Internships & Clubs

More on Strings: String Methods

```
s = "FridaysSaturdaysSundays"  
num = s.count("s")
```

- The first line creates a variable, called `s`, that stores the string: "FridaysSaturdaysSundays"

More on Strings: String Methods

```
s = "FridaysSaturdaysSundays"  
num = s.count("s")
```

- The first line creates a variable, called `s`, that stores the string: "FridaysSaturdaysSundays"
- There are many useful functions for strings (more in Lab 2).

More on Strings: String Methods

```
s = "FridaysSaturdaysSundays"  
num = s.count("s")
```

- The first line creates a variable, called `s`, that stores the string: "FridaysSaturdaysSundays"
- There are many useful functions for strings (more in Lab 2).
- `s.count(x)` will count the number of times the pattern, `x`, appears in `s`.

More on Strings: String Methods

```
s = "FridaysSaturdaysSundays"  
num = s.count("s")
```

- The first line creates a variable, called `s`, that stores the string: "FridaysSaturdaysSundays"
- There are many useful functions for strings (more in Lab 2).
- `s.count(x)` will count the number of times the pattern, `x`, appears in `s`.
 - ▶ `s.count("s")` counts the number of lower case `s` that occurs.

More on Strings: String Methods

```
s = "FridaysSaturdaysSundays"  
num = s.count("s")
```

- The first line creates a variable, called `s`, that stores the string: "FridaysSaturdaysSundays"
- There are many useful functions for strings (more in Lab 2).
- `s.count(x)` will count the number of times the pattern, `x`, appears in `s`.
 - ▶ `s.count("s")` counts the number of lower case `s` that occurs.
 - ▶ `num = s.count("s")` stores the result in the variable `num`, for later.

More on Strings: String Methods

```
s = "FridaysSaturdaysSundays"  
num = s.count("s")
```

- The first line creates a variable, called `s`, that stores the string: `"FridaysSaturdaysSundays"`
- There are many useful functions for strings (more in Lab 2).
- `s.count(x)` will count the number of times the pattern, `x`, appears in `s`.
 - ▶ `s.count("s")` counts the number of lower case `s` that occurs.
 - ▶ `num = s.count("s")` stores the result in the variable `num`, for later.
 - ▶ What would `print(s.count("sS"))` output?

More on Strings: String Methods

```
s = "FridaysSaturdaysSundays"  
num = s.count("s")
```

- The first line creates a variable, called `s`, that stores the string: "FridaysSaturdaysSundays"
- There are many useful functions for strings (more in Lab 2).
- `s.count(x)` will count the number of times the pattern, `x`, appears in `s`.
 - ▶ `s.count("s")` counts the number of lower case `s` that occurs.
 - ▶ `num = s.count("s")` stores the result in the variable `num`, for later.
 - ▶ What would `print(s.count("sS"))` output?
 - ▶ What about:

```
mess = "10 20 21 9 101 35"  
mults = mess.count("0 ")  
print(mults)
```

More on Strings: Indexing & Substrings

```
s = "FridaysSaturdaysSundays"  
days = s[7]  
days = s[7:15]  
days = s[:-1]
```

- Strings are made up of individual characters (letters, numbers, etc.)

More on Strings: Indexing & Substrings

```
s = "FridaysSaturdaysSundays"  
days = s[7]  
days = s[7:15]  
days = s[:-1]
```

- Strings are made up of individual characters (letters, numbers, etc.)
- Useful to be able to refer to pieces of a string, either an individual location or a “substring” of the string.

More on Strings: Indexing & Substrings

```
s = "FridaysSaturdaysSundays"  
days = s[7]  
days = s[7:15]  
days = s[:-1]
```

- Strings are made up of individual characters (letters, numbers, etc.)
- Useful to be able to refer to pieces of a string, either an individual location or a “substring” of the string.

0	1	2	3	4	5	6	7	8	...	16	17	18	19	20	21	22
F	r	i	d	a	y	s	S	a	...	S	u	n	d	a	y	s

More on Strings: Indexing & Substrings

```
s = "FridaysSaturdaysSundays"  
days = s[7]  
days = s[7:15]  
days = s[:-1]
```

- Strings are made up of individual characters (letters, numbers, etc.)
- Useful to be able to refer to pieces of a string, either an individual location or a “substring” of the string.

0	1	2	3	4	5	6	7	8	...	16	17	18	19	20	21	22
F	r	i	d	a	y	s	S	a	...	S	u	n	d	a	y	s
												...	-4	-3	-2	-1

More on Strings: Indexing & Substrings

```
s = "FridaysSaturdaysSundays"
```

- Strings are made up of individual characters (letters, numbers, etc.)
- Useful to be able to refer to pieces of a string, either an individual location or a “substring” of the string.

0	1	2	3	4	5	6	7	8	...	16	17	18	19	20	21	22
F	r	i	d	a	y	s	S	a	...	S	u	n	d	a	y	s
												...	-4	-3	-2	-1

- `s[0]` is

More on Strings: Indexing & Substrings

```
s = "FridaysSaturdaysSundays"
```

- Strings are made up of individual characters (letters, numbers, etc.)
- Useful to be able to refer to pieces of a string, either an individual location or a “substring” of the string.

0	1	2	3	4	5	6	7	8	...	16	17	18	19	20	21	22
F	r	i	d	a	y	s	S	a	...	S	u	n	d	a	y	s
												...	-4	-3	-2	-1

- `s[0]` is 'F'.

More on Strings: Indexing & Substrings

`s = "FridaysSaturdaysSundays"`

- Strings are made up of individual characters (letters, numbers, etc.)
- Useful to be able to refer to pieces of a string, either an individual location or a “substring” of the string.

0	1	2	3	4	5	6	7	8	...	16	17	18	19	20	21	22
F	r	i	d	a	y	s	S	a	...	S	u	n	d	a	y	s
												...	-4	-3	-2	-1

- `s[1]` is

More on Strings: Indexing & Substrings

`s = "FridaysSaturdaysSundays"`

- Strings are made up of individual characters (letters, numbers, etc.)
- Useful to be able to refer to pieces of a string, either an individual location or a “substring” of the string.

0	1	2	3	4	5	6	7	8	...	16	17	18	19	20	21	22
F	r	i	d	a	y	s	S	a	...	S	u	n	d	a	y	s
												...	-4	-3	-2	-1

- `s[1]` is 'r'.

More on Strings: Indexing & Substrings

`s = "FridaysSaturdaysSundays"`

- Strings are made up of individual characters (letters, numbers, etc.)
- Useful to be able to refer to pieces of a string, either an individual location or a “substring” of the string.

0	1	2	3	4	5	6	7	8	...	16	17	18	19	20	21	22
F	r	i	d	a	y	s	S	a	...	S	u	n	d	a	y	s
												...	-4	-3	-2	-1

- `s[-1]` is

More on Strings: Indexing & Substrings

`s = "FridaysSaturdaysSundays"`

- Strings are made up of individual characters (letters, numbers, etc.)
- Useful to be able to refer to pieces of a string, either an individual location or a “substring” of the string.

0	1	2	3	4	5	6	7	8	...	16	17	18	19	20	21	22
F	r	i	d	a	y	s	S	a	...	S	u	n	d	a	y	s
												...	-4	-3	-2	-1

- `s[-1]` is 's'.

More on Strings: Indexing & Substrings

`s = "FridaysSaturdaysSundays"`

- Strings are made up of individual characters (letters, numbers, etc.)
- Useful to be able to refer to pieces of a string, either an individual location or a “substring” of the string.

0	1	2	3	4	5	6	7	8	...	16	17	18	19	20	21	22
F	r	i	d	a	y	s	S	a	...	S	u	n	d	a	y	s
												...	-4	-3	-2	-1

- `s[3:6]` is

More on Strings: Indexing & Substrings

`s = "FridaysSaturdaysSundays"`

- Strings are made up of individual characters (letters, numbers, etc.)
- Useful to be able to refer to pieces of a string, either an individual location or a “substring” of the string.

0	1	2	3	4	5	6	7	8	...	16	17	18	19	20	21	22
F	r	i	d	a	y	s	S	a	...	S	u	n	d	a	y	s
												...	-4	-3	-2	-1

- `s[3:6]` is 'day'.

More on Strings: Indexing & Substrings

`s = "FridaysSaturdaysSundays"`

- Strings are made up of individual characters (letters, numbers, etc.)
- Useful to be able to refer to pieces of a string, either an individual location or a “substring” of the string.

0	1	2	3	4	5	6	7	8	...	16	17	18	19	20	21	22
F	r	i	d	a	y	s	S	a	...	S	u	n	d	a	y	s
												...	-4	-3	-2	-1

- `s[:3]` is

More on Strings: Indexing & Substrings

`s = "FridaysSaturdaysSundays"`

- Strings are made up of individual characters (letters, numbers, etc.)
- Useful to be able to refer to pieces of a string, either an individual location or a “substring” of the string.

0	1	2	3	4	5	6	7	8	...	16	17	18	19	20	21	22
F	r	i	d	a	y	s	S	a	...	S	u	n	d	a	y	s
												...	-4	-3	-2	-1

- `s[:3]` is 'Fri'.

More on Strings: Indexing & Substrings

```
s = "FridaysSaturdaysSundays"
```

- Strings are made up of individual characters (letters, numbers, etc.)
- Useful to be able to refer to pieces of a string, either an individual location or a “substring” of the string.

0	1	2	3	4	5	6	7	8	...	16	17	18	19	20	21	22
F	r	i	d	a	y	s	S	a	...	S	u	n	d	a	y	s
												...	-4	-3	-2	-1

- `s[:-1]` is

More on Strings: Indexing & Substrings

```
s = "FridaysSaturdaysSundays"
```

- Strings are made up of individual characters (letters, numbers, etc.)
- Useful to be able to refer to pieces of a string, either an individual location or a “substring” of the string.

0	1	2	3	4	5	6	7	8	...	16	17	18	19	20	21	22
F	r	i	d	a	y	s	S	a	...	S	u	n	d	a	y	s
												...	-4	-3	-2	-1

- `s[:-1]` is 'FridaysSaturdaysSunday'.
(no trailing 's' at the end)

Lecture Slip

LECTURE 2, CSci 127
SPRING 2022

Name:								
EmpID:								

- **Introducing Design Challenges:** these are “think up an Algorithm”-type exercises. We introduce a topic in lecture, and then we ask you to apply it to solve a problem. Here we are asking you to come up with a sequence of steps (short English sentences) that describe the **process** – i.e. your Algorithm. You should also identify the **input** and **output**.
1. Design a program that **counts** the number of plural nouns provided as a string containing only the nouns separated by spaces. Think about what the input is, what the output is, and how you can determine if a noun is plural.
Note: To simplify the problem, assume all plural nouns end in “s”.

Input:

Output:

Process:

Today's Topics



- For-loops
- `range()`
- Variables
- Characters
- Strings
- **Guests: Internships & Clubs**

Guest Speakers

- Hunter staff
 - ▶ Elise Harris, Internship Manager, Cooperman Business Center and Computer Science
- Club officers
 - ▶ Asad Rafique, Hunter Association of Computing Machinery (ACM)
 - ▶ David Arcos Mawyin, Esports and Game Design Collective (EGD)
 - ▶ Kelly Camacho, Women in Computer Science (WiCS)
 - ▶ Isabel Abonitalla, Google Developers Student Club (DSC)
- See Announcement on Blackboard for links to important resources.

Recap

- In Python, we introduced:

```
1 #Predict what will be printed:
2 for i in range(4):
3     print('The world turned upside down')
4 for j in [0,1,2,3,4,5]:
5     print(j)
6 for count in range(6):
7     print(count)
8 for color in ['red', 'green', 'blue']:
9     print(color)
10 for i in range(2):
11     for j in range(2):
12         print('Look around,')
13     print('How lucky we are to be alive!')
```

Recap

- In Python, we introduced:
 - For-loops

```
1 #Predict what will be printed:
2 for i in range(4):
3     print('The world turned upside down')
4 for j in [0,1,2,3,4,5]:
5     print(j)
6 for count in range(6):
7     print(count)
8 for color in ['red', 'green', 'blue']:
9     print(color)
10 for i in range(2):
11     for j in range(2):
12         print('Look around,')
13     print('How lucky we are to be alive!')
```

Recap

- In Python, we introduced:

- ▶ For-loops
- ▶ `range()`

```
1 #Predict what will be printed:
2 for i in range(4):
3     print('The world turned upside down')
4 for j in [0,1,2,3,4,5]:
5     print(j)
6 for count in range(6):
7     print(count)
8 for color in ['red', 'green', 'blue']:
9     print(color)
10 for i in range(2):
11     for j in range(2):
12         print('Look around,')
13     print('How lucky we are to be alive!')
```

Recap

- In Python, we introduced:

- ▶ For-loops
- ▶ `range()`
- ▶ Variables: ints and strings

```
1 #Predict what will be printed:
2 for i in range(4):
3     print('The world turned upside down')
4 for j in [0,1,2,3,4,5]:
5     print(j)
6 for count in range(6):
7     print(count)
8 for color in ['red', 'green', 'blue']:
9     print(color)
10 for i in range(2):
11     for j in range(2):
12         print('Look around,')
13     print('How lucky we are to be alive!')
```

Recap

- In Python, we introduced:

- ▶ For-loops
- ▶ `range()`
- ▶ Variables: ints and strings
- ▶ Some arithmetic

```
1 #Predict what will be printed:
2 for i in range(4):
3     print('The world turned upside down')
4 for j in [0,1,2,3,4,5]:
5     print(j)
6 for count in range(6):
7     print(count)
8 for color in ['red', 'green', 'blue']:
9     print(color)
10 for i in range(2):
11     for j in range(2):
12         print('Look around,')
13     print('How lucky we are to be alive!')
```

Recap

- In Python, we introduced:

- ▶ For-loops
- ▶ `range()`
- ▶ Variables: ints and strings
- ▶ Some arithmetic
- ▶ String concatenation

```
1 #Predict what will be printed:
2 for i in range(4):
3     print('The world turned upside down')
4 for j in [0,1,2,3,4,5]:
5     print(j)
6 for count in range(6):
7     print(count)
8 for color in ['red', 'green', 'blue']:
9     print(color)
10 for i in range(2):
11     for j in range(2):
12         print('Look around,')
13     print('How lucky we are to be alive!')
```

Recap

```
1 #Predict what will be printed:
2 for i in range(4):
3     print('The world turned upside down')
4 for j in [0,1,2,3,4,5]:
5     print(j)
6 for count in range(6):
7     print(count)
8 for color in ['red', 'green', 'blue']:
9     print(color)
10 for i in range(2):
11     for j in range(2):
12         print('Look around,')
13     print('How lucky we are to be alive!')
```

- In Python, we introduced:

- ▶ For-loops
- ▶ `range()`
- ▶ Variables: ints and strings
- ▶ Some arithmetic
- ▶ String concatenation
- ▶ Functions: `ord()` and `chr()`

Recap

```
1 #Predict what will be printed:
2 for i in range(4):
3     print('The world turned upside down')
4 for j in [0,1,2,3,4,5]:
5     print(j)
6 for count in range(6):
7     print(count)
8 for color in ['red', 'green', 'blue']:
9     print(color)
10 for i in range(2):
11     for j in range(2):
12         print('Look around,')
13     print('How lucky we are to be alive!')
```

- In Python, we introduced:

- ▶ For-loops
- ▶ `range()`
- ▶ Variables: ints and strings
- ▶ Some arithmetic
- ▶ String concatenation
- ▶ Functions: `ord()` and `chr()`
- ▶ String Manipulation

Recap

```
1 #Predict what will be printed:
2 for i in range(4):
3     print('The world turned upside down')
4 for j in [0,1,2,3,4,5]:
5     print(j)
6 for count in range(6):
7     print(count)
8 for color in ['red', 'green', 'blue']:
9     print(color)
10 for i in range(2):
11     for j in range(2):
12         print('Look around,')
13     print('How lucky we are to be alive!')
```

- In Python, we introduced:

- ▶ For-loops
- ▶ `range()`
- ▶ Variables: ints and strings
- ▶ Some arithmetic
- ▶ String concatenation
- ▶ Functions: `ord()` and `chr()`
- ▶ String Manipulation

Practice Quiz & Final Questions



- Since you must pass the final exam to pass the course, we end every lecture with final exam review.

Practice Quiz & Final Questions



- Since you must pass the final exam to pass the course, we end every lecture with final exam review.
- Pull out something to write on (not to be turned in).
- Lightning rounds:
 - ▶ write as much you can for 60 seconds;
 - ▶ followed by answer; and
 - ▶ repeat.
- Past exams are on the webpage (under [Final Exam Information](#)).
- We're starting with Spring 2018, Mock Exam.

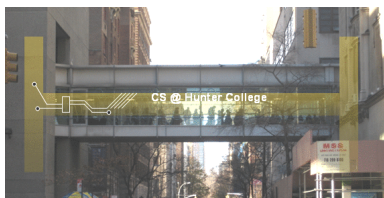
Weekly Reminders!



Before next lecture, don't forget to:

- Work on this week's Online Lab

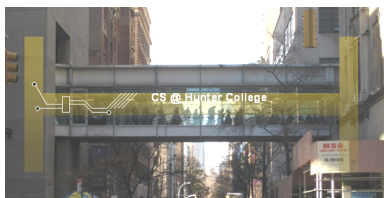
Weekly Reminders!



Before next lecture, don't forget to:

- Work on this week's Online Lab
- Schedule an appointment to take the Quiz in lab 1001E Hunter North

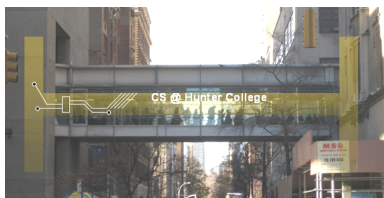
Weekly Reminders!



Before next lecture, don't forget to:

- Work on this week's Online Lab
- Schedule an appointment to take the Quiz in lab 1001E Hunter North
- If you haven't already, schedule an appointment to take the Code Review (**one every two weeks**) in lab 1001E Hunter North

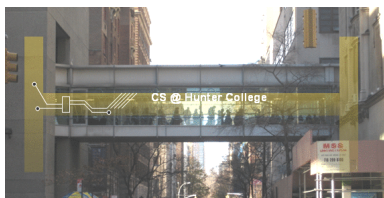
Weekly Reminders!



Before next lecture, don't forget to:

- Work on this week's Online Lab
- Schedule an appointment to take the Quiz in lab 1001E Hunter North
- If you haven't already, schedule an appointment to take the Code Review (**one every two weeks**) in lab 1001E Hunter North
- Submit this week's 5 programming assignments (**programs 6-10**)

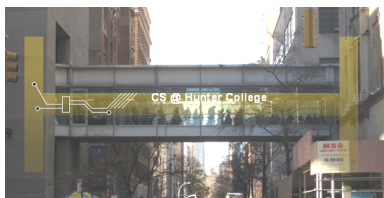
Weekly Reminders!



Before next lecture, don't forget to:

- Work on this week's Online Lab
- Schedule an appointment to take the Quiz in lab 1001E Hunter North
- If you haven't already, schedule an appointment to take the Code Review (**one every two weeks**) in lab 1001E Hunter North
- Submit this week's 5 programming assignments (**programs 6-10**)
- If you need help, schedule an appointment for Tutoring in lab 1001E 11am-5pm

Weekly Reminders!



Before next lecture, don't forget to:

- Work on this week's Online Lab
- Schedule an appointment to take the Quiz in lab 1001E Hunter North
- If you haven't already, schedule an appointment to take the Code Review (**one every two weeks**) in lab 1001E Hunter North
- Submit this week's 5 programming assignments (**programs 6-10**)
- If you need help, schedule an appointment for Tutoring in lab 1001E 11am-5pm
- Take the Lecture Preview on Blackboard on Monday (or no later than 10am on Tuesday)

Lecture Slips & Writing Boards



- Hand your lecture slip to a UTA.
- Return writing boards as you leave.