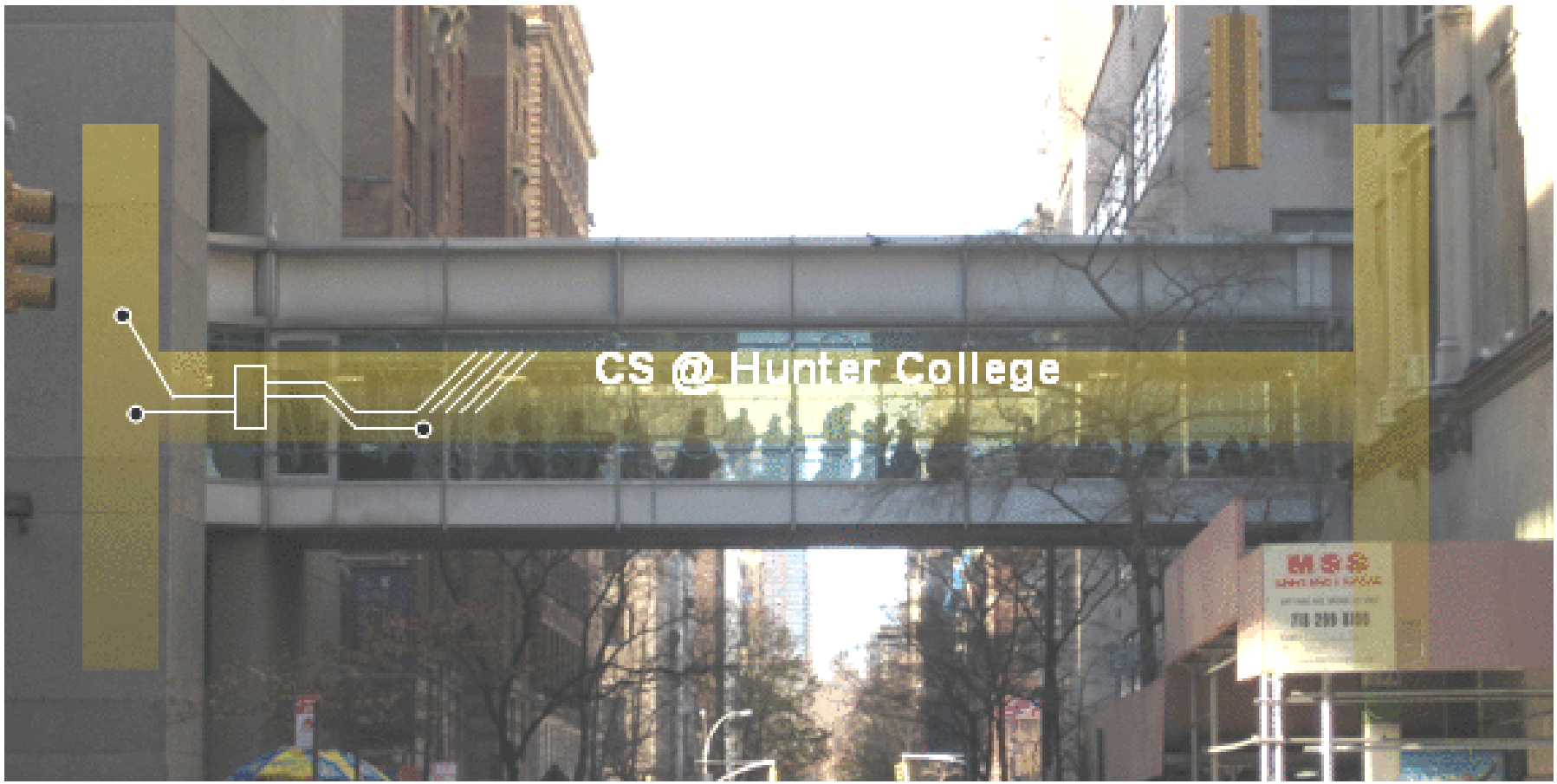
CSci 127: Introduction to Computer Science



hunter.cuny.edu/csci

Announcements

- Starting this week, undergraduate teaching assistants (UTAs) will be in 1001E North, assisting with recitation sessions and holding tutoring hours.

Announcements

- Starting this week, undergraduate teaching assistants (UTAs) will be in 1001E North, assisting with recitation sessions and holding tutoring hours.
- Today is the last day to submit the first program ('Hello, World!') to Gradescope.

Announcements

- Starting this week, undergraduate teaching assistants (UTAs) will be in 1001E North, assisting with recitation sessions and holding tutoring hours.
- Today is the last day to submit the first program ('Hello, World!') to Gradescope.
- Last day to complete in-class Quiz 1 is tomorrow (9/7). You must attend a recitation section to take your quiz.

Frequently Asked Questions

From lecture slips & recitation sections.

Frequently Asked Questions

From lecture slips & recitation sections.

- When is the midterm?

Frequently Asked Questions

From lecture slips & recitation sections.

- When is the midterm?
  There is no midterm. Instead there's 14 in-class quizzes.

Frequently Asked Questions

From lecture slips & recitation sections.

- When is the midterm?
  There is no midterm. Instead there's 14 in-class quizzes.
- When is the final?

Frequently Asked Questions

From lecture slips & recitation sections.

- When is the midterm?
  There is no midterm. Instead there's 14 in-class quizzes.
- When is the final?
  Wednesday, 20 December, 9-11am.

Frequently Asked Questions

From lecture slips & recitation sections.

- When is the midterm?
  There is no midterm. Instead there's 14 in-class quizzes.
- When is the final?
  Wednesday, 20 December, 9-11am.
- Can I submit late homework?

Frequently Asked Questions

From lecture slips & recitation sections.

- When is the midterm?
  There is no midterm. Instead there's 14 in-class quizzes.
- When is the final?
  Wednesday, 20 December, 9-11am.
- Can I submit late homework?
  No. Instead we drop the 5 lowest grades.

Frequently Asked Questions

From lecture slips & recitation sections.

- When is the midterm?
  There is no midterm. Instead there's 14 in-class quizzes.
- When is the final?
  Wednesday, 20 December, 9-11am.
- Can I submit late homework?
  No. Instead we drop the 5 lowest grades.
- I missed class. Do you need documentation?

Frequently Asked Questions

From lecture slips & recitation sections.

- When is the midterm?
  There is no midterm. Instead there's 14 in-class quizzes.
- When is the final?
  Wednesday, 20 December, 9-11am.
- Can I submit late homework?
  No. Instead we drop the 5 lowest grades.
- I missed class. Do you need documentation?
  No. We will automatically drop the 2 lowest quiz grades.
  If you will miss ≥ 3 weeks ($> 20\%$ of class), see us about taking a future term.

Frequently Asked Questions

From lecture slips & recitation sections.

- When is the midterm?
  There is no midterm. Instead there's 14 in-class quizzes.
- When is the final?
  Wednesday, 20 December, 9-11am.
- Can I submit late homework?
  No. Instead we drop the 5 lowest grades.
- I missed class. Do you need documentation?
  No. We will automatically drop the 2 lowest quiz grades.
  If you will miss ≥ 3 *weeks (*$> 20\%$ *of class), see us about taking a future term.*
- Why do I have to work in groups?

Frequently Asked Questions

From lecture slips & recitation sections.

- When is the midterm?
  There is no midterm. Instead there's 14 in-class quizzes.
- When is the final?
  Wednesday, 20 December, 9-11am.
- Can I submit late homework?
  No. Instead we drop the 5 lowest grades.
- I missed class. Do you need documentation?
  No. We will automatically drop the 2 lowest quiz grades.
  If you will miss ≥ 3 weeks ($> 20\%$ of class), see us about taking a future term.
- Why do I have to work in groups?
  It's great practice to explain technical work to others.

Frequently Asked Questions

From lecture slips & recitation sections.

- When is the midterm?
  There is no midterm. Instead there's 14 in-class quizzes.
- When is the final?
  Wednesday, 20 December, 9-11am.
- Can I submit late homework?
  No. Instead we drop the 5 lowest grades.
- I missed class. Do you need documentation?
  No. We will automatically drop the 2 lowest quiz grades.
  If you will miss ≥ 3 weeks ($> 20\%$ of class), see us about taking a future term.
- Why do I have to work in groups?
  It's great practice to explain technical work to others.
- Can I work ahead?

Frequently Asked Questions

From lecture slips & recitation sections.

- When is the midterm?
  There is no midterm. Instead there's 14 in-class quizzes.
- When is the final?
  Wednesday, 20 December, 9-11am.
- Can I submit late homework?
  No. Instead we drop the 5 lowest grades.
- I missed class. Do you need documentation?
  No. We will automatically drop the 2 lowest quiz grades.
  If you will miss ≥ 3 weeks ($> 20\%$ of class), see us about taking a future term.
- Why do I have to work in groups?
  It's great practice to explain technical work to others.
- Can I work ahead?
  Yes! All programs are available, on gradescope, 3 weeks before the deadline.

Frequently Asked Questions

From lecture slips & recitation sections.

- When is the midterm?
  There is no midterm. Instead there's 14 in-class quizzes.
- When is the final?
  Wednesday, 20 December, 9-11am.
- Can I submit late homework?
  No. Instead we drop the 5 lowest grades.
- I missed class. Do you need documentation?
  No. We will automatically drop the 2 lowest quiz grades.
  If you will miss ≥ 3 weeks ($> 20\%$ of class), see us about taking a future term.
- Why do I have to work in groups?
  It's great practice to explain technical work to others.
- Can I work ahead?
  Yes! All programs are available, on gradescope, 3 weeks before the deadline.
- You said "when you take second semester..." I just took this class for Pathways...

Frequently Asked Questions

From lecture slips & recitation sections.

- When is the midterm?
  There is no midterm. Instead there's 14 in-class quizzes.
- When is the final?
  Wednesday, 20 December, 9-11am.
- Can I submit late homework?
  No. Instead we drop the 5 lowest grades.
- I missed class. Do you need documentation?
  No. We will automatically drop the 2 lowest quiz grades.
  If you will miss ≥ 3 weeks ($> 20\%$ of class), see us about taking a future term.
- Why do I have to work in groups?
  It's great practice to explain technical work to others.
- Can I work ahead?
  Yes! All programs are available, on gradescope, 3 weeks before the deadline.
- You said "when you take second semester..." I just took this class for Pathways...
  This is Pathways, but we hope that you will be a CS major/minor.

Frequently Asked Questions

From lecture slips & recitation sections.

- When is the midterm?
  There is no midterm. Instead there's 14 in-class quizzes.
- When is the final?
  Wednesday, 20 December, 9-11am.
- Can I submit late homework?
  No. Instead we drop the 5 lowest grades.
- I missed class. Do you need documentation?
  No. We will automatically drop the 2 lowest quiz grades.
  If you will miss ≥ 3 weeks ($> 20\%$ of class), see us about taking a future term.
- Why do I have to work in groups?
  It's great practice to explain technical work to others.
- Can I work ahead?
  Yes! All programs are available, on gradescope, 3 weeks before the deadline.
- You said "when you take second semester..." I just took this class for Pathways...
  This is Pathways, but we hope that you will be a CS major/minor.
- *We also hope: "Get your education don't forget whence you came..."*

Today's Topics

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

In Pairs or Triples...

Some review and some novel challenges:

```
1  #Predict what will be printed:
2
3  for i in range(4):
4      print('The world turned upside down')
5
6  for j in [0,1,2,3,4,5]:
7      print(j)
8
9  for count in range(6):
10     print(count)
11
12 for color in ['red', 'green', 'blue']:
13     print(color)
14
15 print()
16 print()
17
18 for i in range(2):
19     for j in range(2):
20         print('Look around,')
21     print('How lucky we are to be alive!')
```

Python Tutor

```
#Predict what will be printed:

for i in range(4):
    print('The world turned upside down')

for j in [0,1,2,3,4,5]:
    print(j)

for count in range(6):
    print(count)

for color in ['red', 'green', 'blue']:
    print(color)

print()
print()

for i in range(2):
    for j in range(2):
        print('Look around,')
    print('How lucky we are to be alive!')
```

(Demo with pythonTutor)

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
  - **class variables**: for complex objects, like turtles.

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.

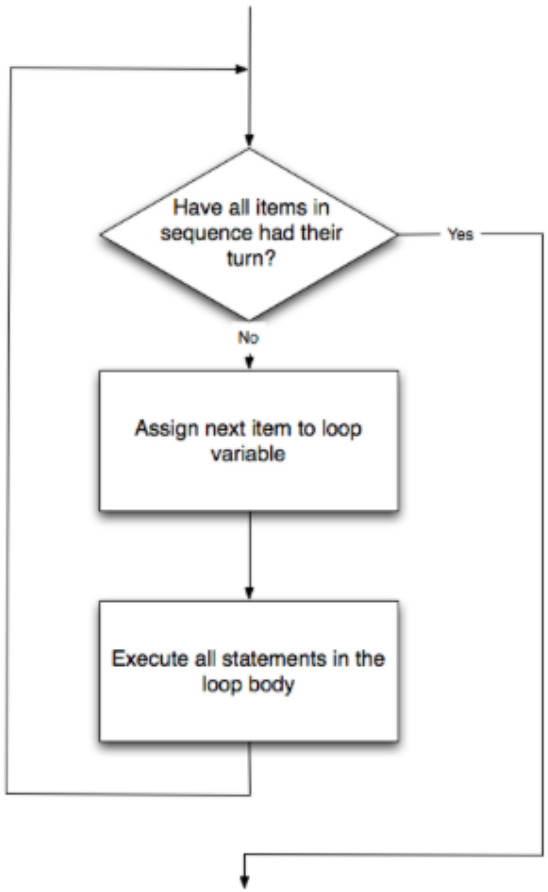
Variable Names

- 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.

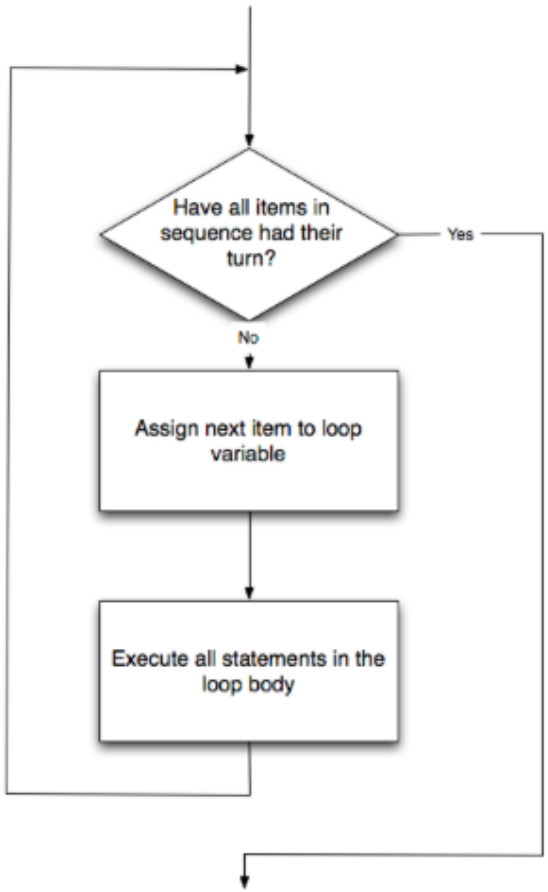
for-loop



How to Think Like CS, §4.5

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

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()`.

In Pairs or Triples...

Some review and some novel challenges:

```
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(x)
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(x)
12
13 for c in "ABCD":
14     print(c)
```

(Demo with pythonTutor)

`range()`

Simplest version:

`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 start+k*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 start+k*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 start+k*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)`

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).

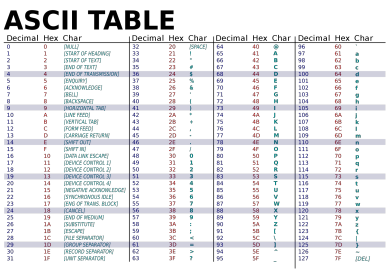
ASCII TABLE

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:

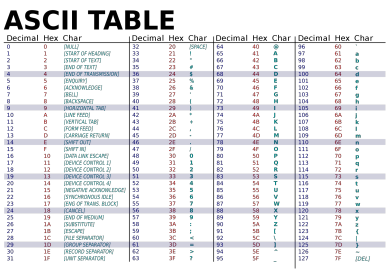
- `ord(c)`: returns Unicode (ASCII) of the character.



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	[END 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]

Converting from Character to Code:

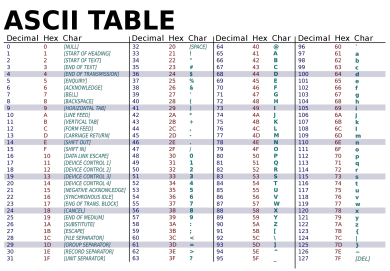


ASCII TABLE

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	[END 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]

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

Converting from Character to Code:

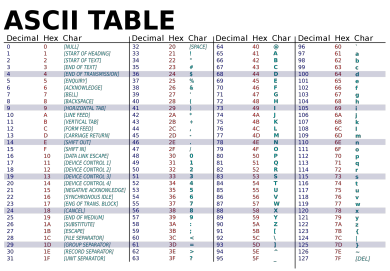


ASCII TABLE

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	[END 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]

- `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:

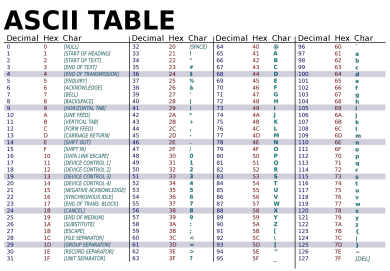


ASCII TABLE

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	[END 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]

- `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:



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	[END 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]

- `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'.

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

```
#Predict what will be printed:

for c in range(65,90):
    print(chr(c))

message = "I love Python"
newMessage = ""
for c in message:
    print(ord(c))   #Print the Unicode of each number
    print(chr(ord(c)+1))    #Print the next character
    newMessage = newMessage + chr(ord(c)+1) #add to the new message
print("The coded message is", newMessage)

word = "zebra"
codedWord = ""
for ch in word:
    offset = ord(ch) - ord('a') + 1 #how many letters past 'a'
    wrap = offset % 26  #if larger than 26, wrap back to 0
    newChar = chr(ord('a') + wrap)  #compute the new letter
    print(wrap, chr(ord('a') + wrap))    #print the wrap & new lett
    codedWord = codedWord + newChar #add the newChar to the coded w

print("The coded word (with wrap) is", codedWord)
```

(Demo with `pythonTutor`)

User Input

Covered in detail in Lab 2:

```
1  mess = input('Please enter a message: ')
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 x to the end of the strings `s`.

Recap

- For-loops

Recap

- For-loops
- `range()`

Recap

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

Recap

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

Recap

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

Recap

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