

CSci 127: Introduction to Computer Science



hunter.cuny.edu/csci

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 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)

Variables



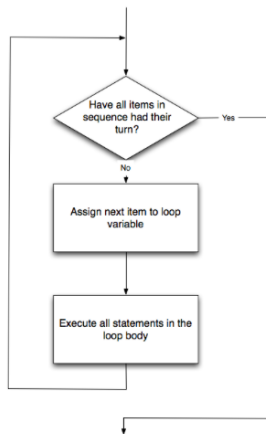
- A **variable** is a reserved memory location for storing a value.
- Different kinds, or **types**, of values need different amounts of space:
 - ▶ **int**: integer or whole numbers
 - ▶ **float**: floating point or real numbers
 - ▶ **string**: sequence of characters
 - ▶ **list**: a sequence of items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

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 (like `for`).
(List of reserved words in *Think CS*, §2.5.)

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()`.

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(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(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(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:

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

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]

Converting from Character to Code:

(There is an ASCII table on the back of today's lecture slip.)

ASCII TABLE

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	00		1	01		2	02		3	03	
4	04		5	05		6	06		7	07	
8	08		9	09		10	0A		11	0B	
12	0C		13	0D		14	0E		15	0F	
16	10		17	11		18	12		19	13	
20	14		21	15		22	16		23	17	
24	18		25	19		26	1A		27	1B	
28	1C		29	1D		30	1E		31	1F	
32	20	Space	33	21	!	34	22	"	35	23	#
36	24	\$	37	25	%	38	26	&	39	27	'
40	28	(41	29)	42	2A	*	43	2B	+
44	2C	,	45	2D	-	46	2E	.	47	2F	/
48	30	0	49	31	1	50	32	2	51	33	3
52	34	4	53	35	5	54	36	6	55	37	7
56	38	8	57	39	9	58	3A	:	59	3B	;
60	3C	<	61	3D	=	62	3E	>	63	3F	?
64	40	@	65	41	A	66	42	B	67	43	C
68	44	D	69	45	E	70	46	F	71	47	
72	48		73	49		74	4A		75	4B	
76	4C		77	4D		78	4E		79	4F	
80	50		81	51		82	52		83	53	
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	a	98	62	b	99	63	c
100	64	d	101	65	e	102	66	f	103	67	
104	68		105	69		106	6A		107	6B	
108	6C		109	6D		110	6E		111	6F	
112	70		113	71		114	72		115	73	
116	74		117	75		118	76		119	77	
120	78		121	79		122	7A		123	7B	
124	7C		125	7D		126	7E		127	7F	
128	80		129	81		130	82		131	83	
132	84		133	85		134	86		135	87	
136	88		137	89		138	8A		139	8B	
140	8C		141	8D		142	8E		143	8F	
144	90		145	91		146	92		147	93	
148	94		149	95		150	96		151	97	
152	98		153	99		154	9A		155	9B	
156	9C		157	9D		158	9E		159	9F	
160	A0		161	A1		162	A2		163	A3	
164	A4		165	A5		166	A6		167	A7	
168	A8		169	A9		170	AA		171	AB	
172	AC		173	AD		174	AE		175	AF	
176	B0		177	B1		178	B2		179	B3	
180	B4		181	B5		182	B6		183	B7	
184	B8		185	B9		186	BA		187	BB	
188	BC		189	BD		190	BE		191	BF	
192	C0		193	C1		194	C2		195	C3	
196	C4		197	C5		198	C6		199	C7	
200	C8		201	C9		202	CA		203	CB	
204	CC		205	CD		206	CE		207	CF	
208	D0		209	D1		210	D2		211	D3	
212	D4		213	D5		214	D6		215	D7	
216	D8		217	D9		218	DA		219	DB	
220	DC		221	DD		222	DE		223	DF	
224	E0		225	E1		226	E2		227	E3	
228	E4		229	E5		230	E6		231	E7	
232	E8		233	E9		234	EA		235	EB	
236	EC		237	ED		238	EE		239	EF	
240	F0		241	F1		242	F2		243	F3	
244	F4		245	F5		246	F6		247	F7	
248	F8		249	F9		250	FA		251	FB	
252	FC		253	FD		254	FE		255	FF	

- `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

```
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)

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`.
- `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

- On lecture slip, write down a topic you wish we had spent more time (and why).
- In Python, we introduced:
 - ▶ For-loops
 - ▶ `range()`
 - ▶ Variables: ints and strings
 - ▶ Some arithmetic
 - ▶ String concatenation
 - ▶ Functions: `ord()` and `chr()`

Name: _____ Email: _____ CSci 127 Sample Final #11

1. (4) What will the following Python code print:

```
months = ["Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"]
year = "2018"
msg = "Msg: " + year + " " + months[0]
last = months[-1]
print(msg.append())
print(msg[0])
print(months[-1].lower())
print(months[2:4])
month = 3
print(months[month-1])
year = 3
print(months[month+year-1][0])
```

Output:

Lecture Slip

Name:

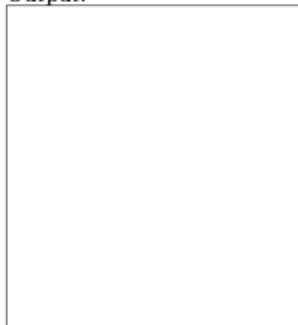
EmpID:

CSci 127 Sample Final, F17

1. (a) What will the following Python code print:

```
months = ["Jan", "Feb", "Mar", "Apr", "May", \
"Jun", "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"]
half = months[6]
print(half.upper())
print(half[0])
print(months[-1].lower())
print(months[2:4])
start = 9
print(months[start-1])
term = 3
print(months[(start+term-1)%12])
```

Output:



Frequently Asked Questions

From lecture slips & recitation sections.

- When is the midterm?
There is no midterm. Instead there's 11 in-class quizzes.
- When is the final?
Wednesday, July 10 1pm - 3pm
- Can I submit late homework?
No.
- I missed class. Do you need documentation?
No. Missing lecture slip & quiz grades are replaced by your final exam score. If you will miss ≥ 2 weeks ($> 20\%$), see us about taking this in 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, 4 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...”