



# Programming and Problem-Solving

## Loops, Lists, and Strings

Dennis Komm

Spring 2021 – March 4, 2021

# Output

## The Function `print()`

Output on screen with `print()`

- **Default:** Linebreak after output
- Linebreak can be replaced by `end`

```
print("The value of x is", end=" ")  
print(x)
```

- Multiple parameters also separated by comma

```
print("Two times the value of", x, "is", 2 * x)
```

- **Default:** Whitespace between strings
- Whitespace can be replaced by `sep`

# User Input

## User Input

### Variables

- Variables are “containers” for values
- So far values have been fixed in program

```
name = "Brunhold"
print("Hello", name)
```

- In the real world, values are mostly
  - entered by the user
  - read in from file / data base (later)

## Celsius to Fahrenheit Calculator

### User input with function `input()`

```
name = input("Enter your name: ")
print("Hello", name)
```

### Attention

Input is string (possibly made of digits) and **no number**

```
x = input("Enter a number: ")
output = 3 * x
print(output) # String concatenation instead of multiplication
```

## Celsius to Fahrenheit Calculator

To get a number, the input has to be converted using the function `int()`

```
x = input("Enter a temperature in degree Celsius: ")
celsius = int(x)
fahrenheit = 9 * celsius / 5 + 32
print("The temperature in degree Fahrenheit is", fahrenheit)
```

or shorter...

```
celsius = int(input("Enter a temperature in degree Celsius: "))
fahrenheit = 9 * celsius / 5 + 32
print("The temperature in degree Fahrenheit is", fahrenheit)
```

## Simple Loops

## for-Loops

Repeat a **statement block** a specific number of times

A **for-loop** consists of the following parts

- A **control variable**, which is defined in the **loop head**
- Additionally, a range is specified
- In the **loop body**, the control variable consecutively takes all values in the provided range

## for-Loops

```
for i in range(1, 4):  
    print("Test")  
    print(i)
```

Loop head  
Loop body (indented)  
Control variable  
Range

### Attention

Last value of *i* is 3 and not 4

## for-Loops

```
for i in range(1, 4):  
    print("Test")  
    print(i)
```

results in...

```
print("Test")  
print(1)
```

```
print("Test")  
print(2)
```

```
print("Test")  
print(3)
```

## Exercise – Square Numbers

### Write a program that

- gets an integer from the user
- stores the value in a variable *x*
- outputs the first *x* square numbers (starting with 1)



## Exercise – Square Numbers

```
x = int(input("Input: "))  
  
for i in range(1, x+1):  
    print(i * i)
```

## Nested Loops

## Nested Loops

### Program

```
for i in range(0, 10):  
    for j in range(0, 10):  
        print(j, end=" ")  
    print()
```

### Output

```
0 1 2 3 4 5 6 7 8 9  
0 1 2 3 4 5 6 7 8 9  
0 1 2 3 4 5 6 7 8 9  
0 1 2 3 4 5 6 7 8 9  
0 1 2 3 4 5 6 7 8 9  
0 1 2 3 4 5 6 7 8 9  
0 1 2 3 4 5 6 7 8 9  
0 1 2 3 4 5 6 7 8 9  
0 1 2 3 4 5 6 7 8 9  
0 1 2 3 4 5 6 7 8 9
```

## Exercise – Number Triangle

Write a program that generates the following output:

```
0 1 2 3 4 5 6 7 8 9  
1 2 3 4 5 6 7 8 9  
2 3 4 5 6 7 8 9  
3 4 5 6 7 8 9  
4 5 6 7 8 9  
5 6 7 8 9  
6 7 8 9  
7 8 9  
8 9  
9
```



## Exercise – Number Triangle

```
for i in range(0, 10):
    for j in range(i, 10):
        print(j, end=" ")
    print()
```

## Lists

## Lists

Simple access to related data

### Example

- `data = [5, 1, 4, 3]`
- Every element has an **index**, starting with 0
- Access to single element using brackets
- `data[0] = 5`
- Length of the list = Number of elements
- `len(data) = 4`

## Loops over Lists

Output all elements of list

```
data = [5, 1, 4, 3]
for item in data:
    print(item)
```

Takes values of all elements in list

or alternatively...

```
data = [5, 1, 4, 3]
for i in range(0, len(data)):
    print(data[i])
```

Takes values of all indices of elements in list

## Loops over Lists – The Function `reversed()`

Output all elements of list backwards

```
data = [6, 7, 5, 1]
for item in reversed(data):
    print(item, end=" ")
```

or alternatively...

```
for i in range(len(data)-1, -1, -1):
    print(data[i], end=" ")
```

**Result**

1 5 7 6

## Simple Initialization

Initialize list with same value in all cells

```
data = [0] * 1000
print(data)
```

```
data = [0, 0, 0, 0] * 250
print(data)
```

**Result**

[0, 0, 0, 0, 0, 0, 0, 0, ..., 0]

(1000 zeros)

## The Function `append()`

Add element to end of a list

**Example**

- List data
- `data.append(5)` inserts 5 at the end

```
data = [1, 4, 8]
data.append(9)
data.append(14)

print(data)
```

## Exercise – Initialize List

**Write a program that**

- gets an integer from the user
- stores the value in a variable `x`
- initializes a list with the first `x` even numbers (starting at 0)



## Exercise – Initialize List

```
x = int(input("Input: "))  
  
data = []  
  
for i in range(0, x):  
    data.append(2 * i)  
  
print(data)
```

## Merging lists

Merge lists using +

### Example

- List data
- `data + [1, 2]` adds elements 1 and two at the end

```
data = [4, 6, 7]  
data2 = data + [9]  
data2 = [1, 3] + data2 + [12, 14]  
  
print(data2)
```

## Strings

## Strings

Strings are “lists of characters” (there are differences)

- Characters correspond (mostly) to keys on keyboard
- Strings are written in quotation marks
- Access to single characters using brackets

- String `word = "HELLO WORLD"`
- `word[0]` is first character
- `word[1]` is second character
- ...
- `word[len(word)-1]` is last character ( `word[-1]`, respectively)

## Characters – The Unicode Table

0–18		19–37		38–56		57–75		76–94		95–113		114–127	
Dec.	Char.	Dec.	Char.	Dec.	Char.	Dec.	Char.	Dec.	Char.	Dec.	Char.	Dec.	Char.
0	NUL	19	DC3	38	&	57	9	76	L	95	_	114	r
1	SOH	20	DC4	39	'	58	:	77	M	96	'	115	s
2	STX	21	NAK	40	(	59	:	78	N	97	a	116	t
3	ETX	22	SYN	41	)	60	<	79	O	98	b	117	u
4	EOT	23	ETB	42	*	61	=	80	P	99	c	118	v
5	ENQ	24	CAN	43	+	62	>	81	Q	100	d	119	w
6	ACK	25	EM	44	,	63	?	82	R	101	e	120	x
7	BEL	26	SUB	45	-	64	@	83	S	102	f	121	y
8	BS	27	ESC	46	.	65	A	84	T	103	g	122	z
9	HT	28	FS	47	/	66	B	85	U	104	h	123	{
10	LF	29	GS	48	0	67	C	86	V	105	i	124	
11	VT	30	RS	49	1	68	D	87	W	106	j	125	}
12	FF	31	US	50	2	69	E	88	X	107	k	126	~
13	CR	32	SP	51	3	70	F	89	Y	108	l	127	DEL
14	SO	33	!	52	4	71	G	90	Z	109	m	...	
15	SI	34	"	53	5	72	H	91	[	110	n		
16	DLE	35	#	54	6	73	I	92	\	111	o		
17	DC1	36	\$	55	7	74	J	93	]	112	p		
18	DC2	37	%	56	8	75	K	94	^	113	q		

## Characters – The Unicode Table

Use functions `ord()` and `chr()`

- `ord(x)` returns position of character `x` in Unicode table
- `chr(y)` returns character at position `y` in Unicode table

```
x = input("Enter a character: ")
print("The character", x, "is at position", ord(x))
```

## Exercise – Print Characters

Write a program that

- outputs the first 26 uppercase letters
- uses a `for`-loop to this end

Recall

The letter `A` is located at position 65 in the Unicode table



## Exercise – Print Characters

```
for i in range(65,91):
    print(chr(i))
```



# Caesar Encryption

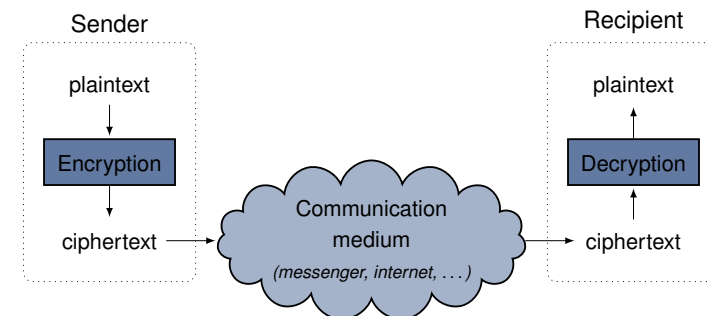


# Symmetric Encryption

## Situation

- Two parties  $A$  and  $B$  want to communicate through an insecure channel
  - Shared key  $k$
  - $A$  encrypts message with  $k$
  - $A$  sends encrypted message to  $B$
  - $B$  decrypts message with  $k$
- $A$  is called **sender**
- $B$  is called **recipient**
- **Symmetric:** Same key for encryption and decryption

# Symmetric Encryption



# Caesar Encryption

## Situation

- Parties *A* and *B* want to communicate over an insecure channel, this time using Caesar encryption
- Shared key *k* as number between 1 and 25
- A* encrypts message by adding *k* to each character
- A* sends encrypted message to *B*
- B* decrypts message by subtracting *k* from each character

# Caesar Encryption

Shift characters by fixed value *k* by adding *k*

## Example

A	B	C	D	E	F	G	H	I	J	K	L	M
W	X	Y	Z	A	B	C	D	E	F	G	H	I
N	O	P	Q	R	S	T	U	V	W	X	Y	Z
J	K	L	M	N	O	P	Q	R	S	T	U	V

- Plaintext:** HELLO WORLD
- Ciphertext:** DAHHK SKNHZ

# Caesar Encryption

- Entered letter is Unicode character between **A** and **Z**

A	B	...	W	X	Y	Z
65	66	...	87	88	89	90

- Subtract **65** so that the result is between **0** and **25**

A	B	...	W	X	Y	Z
0	1	...	22	23	24	25

- Now add key (for instance, **3**) and compute **modulo 26**

A	B	...	W	X	Y	Z
3	4	...	25	0	1	2

- Finally add **65** to the result

A	B	...	W	X	Y	Z
68	69	...	90	65	66	67

# Euclidean Division (Modulo Operation)

- Using “%,” we obtain the residue of the integer division
- Analogously, “//” the part before the decimal point

- $10 \% 3 = 1$ , because  $9 = 3 \cdot 3$
- $10 \% 4 = 2$ , because  $8 = 4 \cdot 2$
- $11 // 5 = 2$ , because  $10 = 5 \cdot 2$
- $23 // 4 = 5$ , because  $20 = 4 \cdot 5$
- $12 \% 3 = 0$ , because  $12 = 4 \cdot 3$

## Exercise – Caesar Encryption

### Write a program that

- runs through a given string
- decrypts each letter with a key  $k$
- tries out each key  $k$
- uses the following formula

$$e = (v - 65 - k) \% 26 + 65$$



Decrypt the ciphertext

TYQZCXLETVTDEVCPLETGPLCMPTE