# Introduction to Programming

Department of Computer Science and Information Systems

Lecturer: Steve Maybank

sjmaybank@dcs.bbk.ac.uk

Autumn 2019 and Spring 2020

## Week 5: Strings and Output

## Example 1 of a Function Call

- `first = input("Enter your first name: ")`
- Name of the function:
  - input
- Argument:
  - the string "Enter your first name: "
- Returned value:
  - a string entered at the keyboard
- The returned value becomes the value of `first`
- The detailed actions of input are hidden. The calling program has knowledge only of the argument and the returned value.

## Example 2 of a Function Call

- `mn = min(1, 5, 8, 6.2)`

- Name of the function: min
- Arguments: the numbers 1, 5, 8, 6.2
- Returned value: 1

- The returned value becomes the value of mn
- min is unusual in that it can have any number of arguments

## Revision: int and float

- `int("5")`

# returns the integer 5

- `int(5.999)`

# returns the integer 5

- `int(-5.999)`

# returns the integer -5

- `int("5.672")`

# error

- `float("5.67")`

# returns 5.67

- `float(5.67)`

# returns 5.67

- `float("3E2")`

# returns 300.0

- `float("3")`

# returns 3.0

- `float("5.2*6.7")`

# error

## Strings

- A string is a sequence of characters
- `greeting = "Hello"`
- "Hello" is a string with 5 characters

# "Hello" is the value of the variable `greeting`
# "Hello" is a string literal

## Alternative Notation

- 'Hello' is the same string as "Hello"

```
print('He said "Hello" today')
```

# The double quotes " are characters in the string

Result: He said "Hello" today

```
print("He said 'Hello' today")
```

# The single quotes ' are characters in the string

Result: He said 'Hello' today

```
print("He said \"Hello\" and 'Goodbye' today")
```

# The single quotes ' are characters in the string

Result: He said "Hello" and 'Goodbye' today

## Length of a String

- The length of string is the number of characters in the string.
- The function `len` takes a string as an argument and returns the length of the string

```
length1 = len("World")
```

# the variable length1 is assigned the value 5

```
length2 = len("")
```

# "" is the empty string

# the variable length2 is assigned the value 0

## Concatenation

- The + operator concatenates strings

```
firstName = "Harry"
lastName = "Morgan"
name = firstName+lastName
```

# name has the value "HarryMorgan"

How to have "Harry Morgan" as the value of name?

```
name = firstName+" "+lastName
```

- It is not possible to mix strings and numbers

```
test = "Harry"+5     # error
test = "Harry"+"5"
```

## Concatenation and Repetition

- The * operator can be used to repeat strings

```
dashes = "-" * 10      # dashes has the value "----------"
dashes = 10 * "-"      # also possible
```

What are the results?

```
separator1 = "_" * 5 + "&" * 10 + "#" * 5
```

Result: '_____&&&&&&&&&&#####'

```
separator2 = 3 * "@" + 2 * "w" * 5 + "9" * 4
```

Result: '@@@wwwwwwwwww9999'

```
separator3 = 10.0 * '&'
```

Result: error

## Convert Numbers to Strings

- Recall `int` and `float` convert strings to numbers
- The function `str` converts numbers to strings

```
string1 = str(45)
# string1 has the value "45"
string2 = str(3E2)
# string2 has the value "300.0"
string3 = str(6+5)
# string3 has the value "11"
string4 = str("45")
# string4 has the value "45"
```

```
test = "Harry"+5
# error   How to fix this?
test = "Harry"+"5"
test = "Harry"+str(5)
# Which way is preferred? Why?

string5 = str(pi)
#string5 has the value
"3.141592653589793"
```

## String Indexing

- The characters on a string are indexed left to right, starting from 0

| H | a | r | r | y |
|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 |

- Individual characters can be extracted from a string

```
name = "Harry"
first = name[0]
```

# first has the value "H"

```
last = name[4]
```

or

```
last = name[-1]
```

# last has the value "y"

```
other = name[5]
```

# error

## String Operations

| Statement | Result | Comment |
|---|---|---|
| string="Py"<br>string = string+"thon" | string is set to "Python" | When applied to strings, + denotes concatenation |
| print("Please" +<br>" enter your name: ") | Prints<br>Please enter your name: | Use concatenation to break up strings that don’t fit onto one line |
| team = str(49)+"ers" | team is set to "49ers" | Because 49 is an integer it must be converted to a string |
| greeting = "H & S"<br>n = len(greeting) | n is set to 5 | Each space counts as one character |
| string = "Sally"<br>ch = string[1] | ch is set to "a" | Note that the initial position has index 0 |
| last =<br>string[len(string)-1] | last is set to the string containing the last character in string | The last character has position len(string)-1 |

## Escape Sequences \", \n, \\

- `string = "He said \"Hello\""`

# Each \" is treated as a single character – the double quote.

`len(string)?`

# The value of string has 15 characters

- `print("*\n**\n***")`

# Each \n produces a new line. The result is

```
*
**
***
```

`len("*\n**\n***")?`

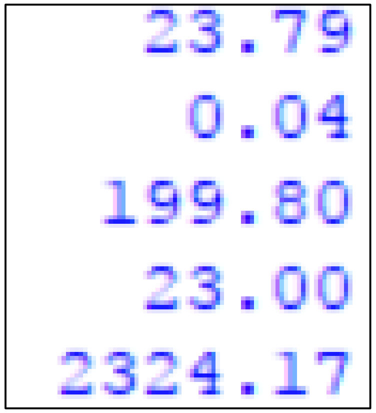
# 8

- `print("\\")`

# prints  \

## A Motivation Example

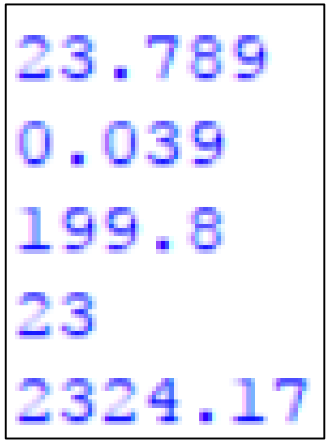
```
price1 = 23.789
price2 = 0.039
price3 = 199.8
price4 = 23
price5 = 2324.17
```



```
  23.79
   0.04
 199.80
  23.00
2324.17
```

desired output

```
print(price1)
print(price2)
print(price3)
print(price4)
print(price5)
```



```
23.789
0.039
199.8
23
2324.17
```

actual output

# String Format Operator %

`formatString % value` ➔ <u>this is a string!</u>

```
"%.2f"  % price
```

# create a string containing the value of price
# correct to two decimal places. f is for float.
# The rightmost % is the string format operator.

```
price = 10.809
string = "%.2f" % price
```

# value of string is "10.81"

## Vocabulary

- "%m.nf" is the format string
- %m.nf is the format specifier
- m is the field width
- % (outside the format string) is the string format operator
  - Apply the format string on a floating number

## Examples

```
value = 56.68
```

- `"%0.3f" % value`

# the result is the string "56.680"

- `"%8.3f" % value`

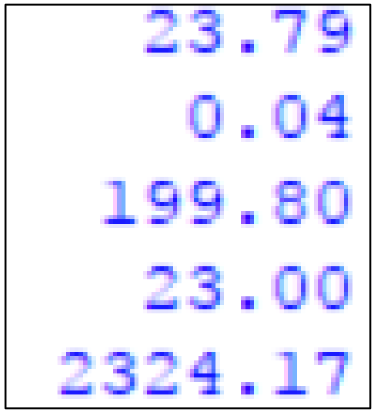
# the result is the string "  56.680"
# Note that "  56.680" has two spaces on the left, to ensure
# that the length of the string is 8, not 6

- `"%4.3f" % value`

# the result is the string "56.680"
# Note that "56.680" has length 6, not 4

## A Motivation Example

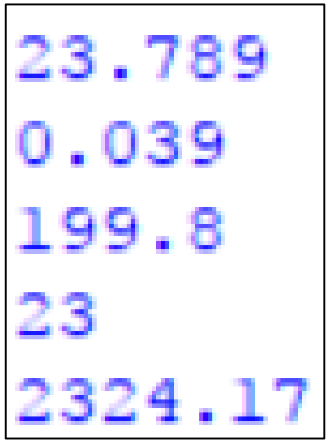
```
price1 = 23.789
price2 = 0.039
price3 = 199.8
price4 = 23
price5 = 2324.17
```



```
  23.79
   0.04
 199.80
  23.00
2324.17
```

desired output

```
print(price1)
print(price2)
print(price3)
print(price4)
print(price5)
```



```
23.789
0.039
199.8
23
2324.17
```

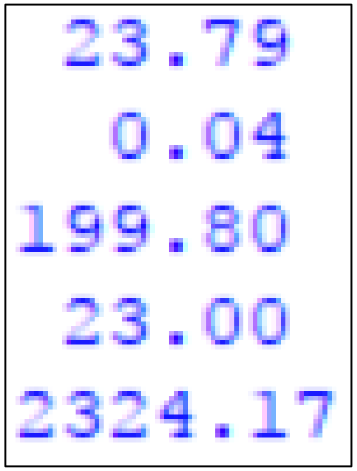
actual output

## A Motivation Example

```
price1 = 23.789
price2 = 0.039
price3 = 199.8
price4 = 23
price5 = 2324.17
```

```
formatPrice = "%6.2f"

print(formatPrice % price1)
print(formatPrice % price2)
print(formatPrice % price3)
print(formatPrice % price4)
print(formatPrice % price5)
```



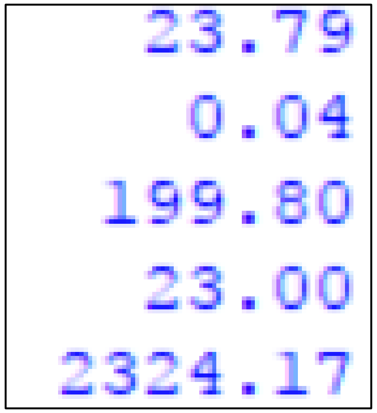
```
 23.79
  0.04
199.80
 23.00
2324.17
```

## A Motivation Example

```
price1 = 23.789
price2 = 0.039
price3 = 199.8
price4 = 23
price5 = 2324.17
```

```
formatPrice = "%8.2f"

print(formatPrice % price1)
print(formatPrice % price2)
print(formatPrice % price3)
print(formatPrice % price4)
print(formatPrice % price5)
```



```
   23.79
    0.04
  199.80
   23.00
 2324.17
```

## Integers and Strings

- For integers use %d or %nd, for example

```
price = 105

string1 = "%5d" % price
```

# result "  105"

```
string2 = "%2d" % price
```

# result "105"

- For strings use %s or %ns, for example

```
string3 = "%7s" % "Hello"
```

# result "  Hello"

## More Examples

| **Format String** | **Sample output** ~ is a space | **Comments** |
|---|---|---|
| "%d" | 24 | Use d with an integer |
| "%5d" | ~~~24 | Spaces are added so that the field width is 5 |
| "%05d" | 00024 | If you add 0 before the field width, zeroes are added instead of spaces. |
| "Quantity:%5d" | Quantity:~~~24 | Characters inside a format string but outside the format specifier appear in the output. |
| “%.f" | 1.21997 | Use f with a floating point number |
| "%.2f" | 1.22 | Prints two digits after the decimal point |
| "%7.2f" | ~~~1.22 | Spaces are added so that the field width is 7 |
| "%s" | Hello | Use s with a string |
| “%9s" | ~~~~Hello | Strings are right-justified by default |
| "%-9s" | Hello~~~~ | Use a negative field width to left-justify |
| "%d %.2f" | 24~1.22 | You can format multiple values at once |
| "%d%%" | 24% | To add a percentage sign to the output, use %% |

## Multiple Format Specifiers

- Syntax for the string format operator

```
    formatString % (value_1, value_2, .., value_n)
```

- The format string must contain n format specifiers, one for each value.
- If there is only one value then the brackets can be omitted,

```
    formatString % value
```

- Example

```
quantity = 100
total = 509.371
print("Quantity: %d Total: %10.2f" % (quantity, total))
```

# prints "Quantity: 100 Total:     509.37"

## Example

title = "Quantity"

print("%10s %8d" % (title, 24))

# “$\underbrace{\sim\sim\text{Quantity}}_{\text{10 widths}}\underbrace{\sim\sim\sim\sim\sim\sim\text{24}}_{\text{8 widths}}$“

print ("%-10s %8d" % (title, 24))

# "$\underbrace{\text{Quantity}\sim\sim}_{\text{10 widths}}\underbrace{\sim\sim\sim\sim\sim\sim\text{24}}_{\text{8 widths}}$“

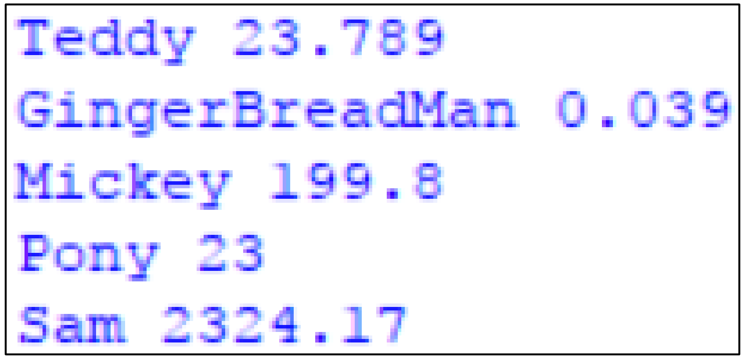
Note that ~ represents a space.

# A Motivation Example

```
price1 = 23.789
price2 = 0.039
price3 = 199.8
price4 = 23
price5 = 2324.17
```

```
item1 = "Teddy"
item2 = "GingerBreadMan"
item3 = "Mickey"
item4 = "Pony"
item5 = "Sam"
```

```
print(item1, price1)
print(item2, price2)
print(item3, price3)
print(item4, price4)
print(item5, price5)
```



```
Teddy 23.789
GingerBreadMan 0.039
Mickey 199.8
Pony 23
Sam 2324.17
```

## A Motivation Example

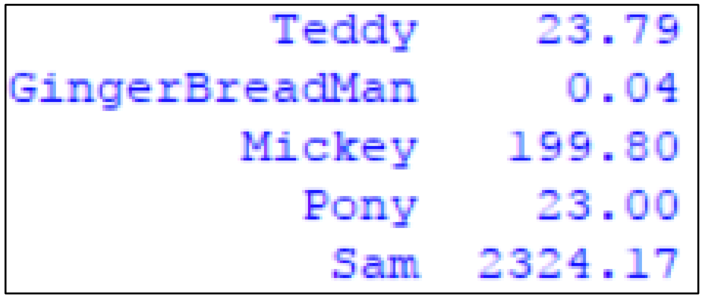
```
price1 = 23.789
price2 = 0.039
price3 = 199.8
price4 = 23
price5 = 2324.17

item1 = "Teddy"
item2 = "GingerBreadMan"
item3 = "Mickey"
item4 = "Pony"
item5 = "Sam"
```

```
formatItemPrice = "%14s %8.2f"

#or alternatively,
#formatItem = "%14s"
#formatPrice = "%8.2f"
#formatItemPrice = formatItem+" "+formatPrice

print(formatItemPrice % (item1, price1))
print(formatItemPrice % (item2, price2))
print(formatItemPrice % (item3, price3))
print(formatItemPrice % (item4, price4))
print(formatItemPrice % (item5, price5))
```



```
         Teddy    23.79
GingerBreadMan     0.04
        Mickey   199.80
          Pony    23.00
           Sam  2324.17
```

# A Motivation Example

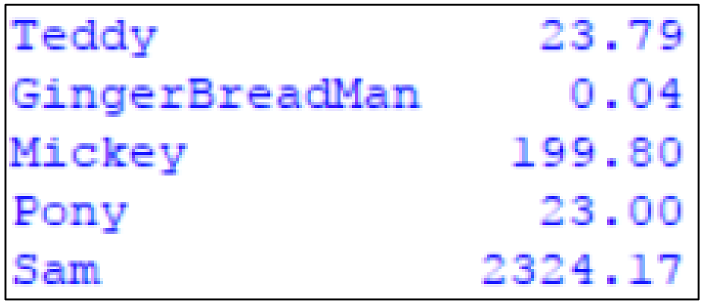
```
price1 = 23.789
price2 = 0.039
price3 = 199.8
price4 = 23
price5 = 2324.17

item1 = "Teddy"
item2 = "GingerBreadMan"
item3 = "Mickey"
item4 = "Pony"
item5 = "Sam"
```

```
formatItemPrice = "%-14s %8.2f"

#or alternatively,
#formatItem = "%-14s"
#formatPrice = "%8.2f"
#formatItemPrice = formatItem+" "+formatPrice

print(formatItemPrice % (item1, price1))
print(formatItemPrice % (item2, price2))
print(formatItemPrice % (item3, price3))
print(formatItemPrice % (item4, price4))
print(formatItemPrice % (item5, price5))
```



```
Teddy             23.79
GingerBreadMan     0.04
Mickey           199.80
Pony              23.00
Sam             2324.17
```

# A Motivation Example

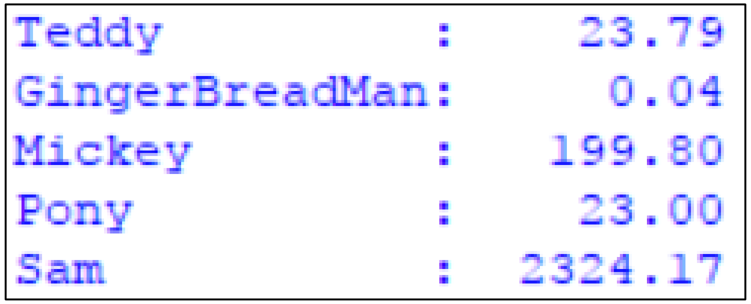
```
formatItemPrice = "%-14s: %8.2f"

#or alternatively,
#formatItem = "%-14s"
#formatPrice = "%8.2f"
#formatItemPrice = formatItem+“: "+formatPrice
```

```
price1 = 23.789
price2 = 0.039
price3 = 199.8
price4 = 23
price5 = 2324.17

item1 = "Teddy"
item2 = "GingerBreadMan"
item3 = "Mickey"
item4 = "Pony"
item5 = "Sam"
```

```
print(formatItemPrice % (item1, price1))
print(formatItemPrice % (item2, price2))
print(formatItemPrice % (item3, price3))
print(formatItemPrice % (item4, price4))
print(formatItemPrice % (item5, price5))
```



```
Teddy         :    23.79
GingerBreadMan:     0.04
Mickey        :   199.80
Pony          :    23.00
Sam           :  2324.17
```

# Format Specifier Summary

formatString % value

~ represents a space

e.g., print("%.f" % 35.678), print("%d" % -5), or print("%s" % "hello")

| formatString | | value | | |
|---|---|---|---|---|
| | | (float) 35.678 | (int) -5 | (string) "hello" |
| float (round) | "%.f" | "36" | "-5" | error |
| | "%.2f" | "35.68" | "-5.00" | |
| | "%6.1f" | "~~35.7" | "~~-5.0" | |
| | "%07.2f" | "0035.68" | "-005.00" | |
| int (trunc) | "%d" | "35" | "-5" | error |
| | "%5d" | "~~~35" | "~~~-5" | |
| | "%-5d" | "35~~~" | "-5~~~" | |
| string | "%s" | "35.678" | "-5" | "hello" |
| | "%7s" | ~35.678" | "~~~~~-5" | "~~hello" |
| | "%3s" | "35.678" | "~-5" | "hello" |

## Programming Exercise 1

- R2.15. Write pseudocode for a program that computes the first and last digit of a number.
- For example, if the input is 23456 the program should print 2 and 6.
- Use % and log(x, 10).

## Programming Exercise 1

- R2.15. Write pseudocode for a program that computes the first and last digit of a number. For example, if the input is 23456 the program should print 2 and 6. Use % and log(x, 10).

```
number = int(input("Please enter a number:"))
digits = int(log(number, 10))
#digits is the (number of digits of the number – 1)

firstDigit = int(number//10**digits)
lastDigit = number%10  # if number is an integer
```

## Programming Exercise 1

- R2.15. Write pseudocode for a program that computes the first and last digit of a number. For example, if the input is 23456 the program should print 2 and 6. Use % and log(x, 10).

```
numberStr = input(Please enter a number:)
firstDigit=int(numberStr[0])
lastDigit= int(numberStr[-1])
```

The program is valid for positive numbers only. It can be a floating point number.

## Programming Exercise 2

- R2.21.How do you get the first character of a string?
- The last character?
- The middle character (if the length is odd)?
- The middle two characters (if the length is even)?
- Harry: middle r
- Potter: middle tt

## Programming Exercise 2

- R2.21.How do you get the first character of a string? The last character? The middle character (if the length is odd)? The middle two characters (if the length is even)?

```
inputStr = input("Please input a string:)
firstChar = inputStr[0]
lastChar = inputStr[-1]

midChar = inputStr[(len(inputStr)-1)//2]# if length is odd
midChar1 = inputStr[len(inputStr)//2-1]  # if length is even

midChar2 = inputStr[len(inputStr)//2]
```