

Functions

TEAM  
INFDEV

Introduction

What is  
abstraction?

Problem  
discussion

General idea

Technical  
details

Assignments

Conclusion

# Functions

TEAM INFDEV

Hogeschool Rotterdam  
Rotterdam, Netherlands

Functions

TEAM  
INFDEV

Introduction

What is  
abstraction?

Problem  
discussion

General idea

Technical  
details

Assignments

Conclusion

# Introduction

Functions

TEAM  
INFDEV

Introduction

What is  
abstraction?

Problem  
discussion

General idea

Technical  
details

Assignments

Conclusion

## Lecture topics

- Mechanism of abstraction
- The need for functions
- Functions in Python

Functions

TEAM  
INFDEV

Introduction

What is  
abstraction?

Problem  
discussion

General idea

Technical  
details

Assignments

Conclusion

# What is abstraction?

# What is abstraction?

Functions

TEAM  
INFDEV

Introduction

What is  
abstraction?

Problem  
discussion

General idea

Technical  
details

Assignments

Conclusion

## Introduction

- The big issue of the whole course is **abstraction** in programming
- Abstraction is a fundamental concept in programming to reduce repetition
- We sit atop a mountain of abstraction, which we make taller at every iteration

# What is abstraction?

Functions

TEAM  
INFDEV

Introduction

What is  
abstraction?

Problem  
discussion

General idea

Technical  
details

Assignments

Conclusion

## Grab the student next to you

- Describe what you just did so that someone else can perform the same action

# What is abstraction?

Functions

TEAM  
INFDEV

Introduction

What is  
abstraction?

Problem  
discussion

General idea

Technical  
details

Assignments

Conclusion

## Grab the student next to you

- Describe what you just did so that someone else can perform the same action
- Now add specific details about the movements of your arm and phalanges (pieces of fingers)

# What is abstraction?

Functions

TEAM  
INFDEV

Introduction

What is  
abstraction?

Problem  
discussion

General idea

Technical  
details

Assignments

Conclusion

## Grab the student next to you

- Describe what you just did so that someone else can perform the same action
- Now add specific details about the movements of your arm and phalanges (pieces of fingers)
- Now realize that there are even more subcomponents: individual muscles, tendons, etc.



# What is abstraction?

Functions

TEAM  
INFDEV

Introduction

What is  
abstraction?

Problem  
discussion

General idea

Technical  
details

Assignments

Conclusion

## Grab the student next to you

- Describe what you just did so that someone else can perform the same action
- Now add specific details about the movements of your arm and phalanges (pieces of fingers)
- Now realize that there are even more subcomponents: individual muscles, tendons, etc.
- But then we have also cells that make these up
- ...

# What is abstraction?

Functions

TEAM  
INFDEV

Introduction

What is  
abstraction?

Problem  
discussion

General idea

Technical  
details

Assignments

Conclusion

## Human love for abstraction

- Our brain cannot handle so many details
- To cope with this, we are structured in layers
- Our consciousness manipulates only the upper layers with simple instructions
- *Raise arm above head*

# What is abstraction?

Functions

TEAM  
INFDEV

Introduction

What is  
abstraction?

Problem  
discussion

General idea

Technical  
details

Assignments

Conclusion

## Human love for abstraction

- The same happens with regular language
- “*Go buy a liter of milk*” is quite a short description
- The underlying operation is very complex

# Complexity of simple instructions

Functions

TEAM  
INFDEV

Introduction

What is  
abstraction?

Problem  
discussion

General idea

Technical  
details

Assignments

Conclusion

```
1  Go buy a liter of milk =  
2    Turn game off  
3    Get up from the couch  
4    Curse the instruction giver  
5    Get dressed  
6    Put money in pocket  
7    Leave house  
8    Reach nearest shop  
9    Enter shop  
10   Find milk  
11   Take one liter bottle  
12   Pay milk  
13   Go home  
14   Give milk to instruction giver
```

# What is abstraction?

Functions

TEAM  
INFDEV

Introduction

What is  
abstraction?

Problem  
discussion

General idea

Technical  
details

Assignments

Conclusion

## Human love for abstraction

- And clearly something like “*reach nearest shop*” is not a trivial instruction by itself
- Think about all the things you give for granted
  - Crossing roads
  - Traffic lights
  - Pathfinding
  - Road work and obstructions
  - Use of transportation methods
  - ...

Functions

TEAM  
INFDEV

Introduction

What is  
abstraction?

Problem  
discussion

General idea

Technical  
details

Assignments

Conclusion

# Problem discussion

Functions

TEAM  
INFDEV

Introduction

What is  
abstraction?

Problem  
discussion

General idea

Technical  
details

Assignments

Conclusion

## Introduction

- Consider many operations on number
  - finding whether a number is prime
  - finding whether a number is odd or even
  - finding the Fibonacci value of a given number
  - ...

# counting down..

Functions

TEAM  
INFDEV

Introduction

What is  
abstraction?

Problem  
discussion

General idea

Technical  
details

Assignments

Conclusion

```
1 cnt = startAt
2 while not(cnt == 0):
3     cnt = cnt - 1
4 print("Lift_off!!!")
```

## Introduction



# counting down..

Functions

TEAM  
INFDEV

Introduction

What is  
abstraction?

Problem  
discussion

General idea

Technical  
details

Assignments

Conclusion

```
1 cnt = startAt
2 while not(cnt == 0):
3     cnt = cnt - 1
4 print("Lift_off!!!")
```

## Introduction

- What does cnt contain if startAt equals 10?
- What do we do with the value of startAt?
- Does it even matter?

Functions

TEAM  
INFDEV

Introduction

What is  
abstraction?

Problem  
discussion

General idea

Technical  
details

Assignments

Conclusion

## Introduction

- Suppose that we want another start, `newStartAt`
- How do we do it?

# Lenght of a list

Functions

TEAM  
INFDEV

Introduction

What is  
abstraction?

Problem  
discussion

General idea

Technical  
details

Assignments

Conclusion

```
1  
2 cnt = newStartAt  
3 while not(cnt == 0):  
4     cnt = cnt - 1  
5 print("Lift_off!!!")
```

## Introduction

Functions

TEAM  
INFDEV

Introduction

What is  
abstraction?

Problem  
discussion

General idea

Technical  
details

Assignments

Conclusion

```
1  
2 cnt = newStartAt  
3 while not(cnt == 0):  
4     cnt = cnt - 1  
5 print("Lift_off!!!")
```

## Introduction

- Looks suspiciously like the previous code block
- Why?

Functions

TEAM  
INFDEV

Introduction

What is  
abstraction?

Problem  
discussion

General idea

Technical  
details

Assignments

Conclusion

# General idea

Functions

TEAM  
INFDEV

Introduction

What is  
abstraction?

Problem  
discussion

General idea

Technical  
details

Assignments

Conclusion

## Adding our own layers

- The goal of this lecture is to add a new layer of abstraction to our programs
- We wish to reuse **implementations**
- This layer of abstraction is called **functions**

# Adding our own layers

Functions

TEAM  
INFDEV

Introduction

What is  
abstraction?

Problem  
discussion

General idea

Technical  
details

Assignments

Conclusion

```
1 +-----+
2 | ...      |
3 +-----+
4 | Functions      |
5 +-----+
6 | data structures      |
7 +-----+
8 | if, for, while, variables |
9 +-----+
10 | (Python) runtime      |
11 +-----+
12 ...
```

## Description

- A function is a collection of instructions and variables
- Some instructions and variables are fixed inside its **body**
- Other instructions and variables come from outside the function, and thus are not fixed; these are called **parameters** of the function
- We try to strike the right balance between flexibility and work done
- The function returns a final result that can be recovered by the code that uses the function



# Blueprint of a function (NOT ACTUAL PYTHON CODE!)

Functions

TEAM  
INFDEV

Introduction

What is  
abstraction?

Problem  
discussion

General idea

Technical  
details

Assignments

Conclusion

```
1 count_down starting from a number:  
2   cnt = number  
3   while not(cnt == 0):  
4       cnt = cnt - 1  
5   return "Lift_off!!!" as final result
```

## Description

# Blueprint of a function (NOT ACTUAL PYTHON CODE!)

Functions

TEAM  
INFDEV

Introduction

What is  
abstraction?

Problem  
discussion

General idea

Technical  
details

Assignments

Conclusion

```
1 count_down starting from a number:  
2   cnt = number  
3   while not(cnt == 0):  
4       cnt = cnt - 1  
5   return "Lift_off!!!" as final result
```

## Description

- count\_down is the **function name**
- number is the only **parameter**
- Lines 2 through 4 are **fixed**
- "Lift off!!!" is the **final result**

Functions

TEAM  
INFDEV

Introduction

What is  
abstraction?

Problem  
discussion

General idea

Technical  
details

Assignments

Conclusion

## Using the function

- Code that needs a count down function can now simply invoke function `count_down`
- The resulting code will simply be `result = count_down(5)`
- `result` will be assigned with the value returned by the function, i.e., "Lift off!!!"

Functions

TEAM  
INFDEV

Introduction

What is  
abstraction?

Problem  
discussion

General idea

Technical  
details

Assignments

Conclusion

# Technical details

## Introduction

- A function can be defined in Python quite easily
- The syntax is:
  - `def <<name>>(<<parameters>>):a`
    - `body`
    - `return <<result>>`
- Inside a function we can put whatever instructions we need
  - `if`
  - `for`
  - `...`

---

<sup>a</sup>Parameters might be none, thus we can write simply `()`

<sup>b</sup>Multiple parameters are separated by a comma, thus  
`(<<p1>>, <<p2>>, ..., <<pn>>)`

## Using the function

- After we declare a function, we can use it
- The syntax is quite simple
  - `<<name>>(<<parameters>>)` to just call the function and ignore the result
  - `<<v>> = <<name>>(<<parameters>>)` to call the function and assign the result to the `<<v>>` variable
- After calling the function, we enter the local environment of the function
- Variables, the PC, etc. are separate from those of the calling site

# Runtime example

Functions

TEAM  
INFDEV

Introduction

What is  
abstraction?

Problem  
discussion

General idea

Technical  
details

Assignments

Conclusion

PC
6

```
1 def count_down(number):  
2     cnt = number  
3     while not(cnt == 0):  
4         cnt = cnt - 1  
5     return "Lift_off!!!"  
6 print(count_down(2))
```

# Runtime example

Functions

TEAM  
INFDEV

Introduction

What is  
abstraction?

Problem  
discussion

General idea

Technical  
details

Assignments

Conclusion

PC
6

```
1 def count_down(number):  
2     cnt = number  
3     while not(cnt == 0):  
4         cnt = cnt - 1  
5     return "Lift_off!!!"  
6 print(count_down(2))
```

PC	count_down	PC	number
6	nil	2	2



# Runtime example

Functions

TEAM  
INFDEV

Introduction

What is  
abstraction?

Problem  
discussion

General idea

Technical  
details

Assignments

Conclusion

PC	count_down	PC	number
6	nil	2	2

```
1 def count_down(number):  
2     cnt = number  
3     while not(cnt == 0):  
4         cnt = cnt - 1  
5     return "Lift_off!!!"  
6 print(count_down(2))
```

# Runtime example

Functions

TEAM  
INFDEV

Introduction

What is  
abstraction?

Problem  
discussion

General idea

Technical  
details

Assignments

Conclusion

PC	count_down	PC	number
6	nil	2	2

```
1 def count_down(number):  
2     cnt = number  
3     while not(cnt == 0):  
4         cnt = cnt - 1  
5     return "Lift_off!!!"  
6 print(count_down(2))
```

PC	count_down	PC	number	cnt
6	nil	3	2	2

# Runtime example

Functions

TEAM  
INFDEV

Introduction

What is  
abstraction?

Problem  
discussion

General idea

Technical  
details

Assignments

Conclusion

PC	count_down	PC	number	cnt
6	nil	3	2	2

```
1 def count_down(number):  
2     cnt = number  
3     while not(cnt == 0):  
4         cnt = cnt - 1  
5     return "Lift_off!!!"  
6 print(count_down(2))
```

# Runtime example

Functions

TEAM  
INFDEV

Introduction

What is  
abstraction?

Problem  
discussion

General idea

Technical  
details

Assignments

Conclusion

PC	count_down	PC	number	cnt
6	nil	3	2	2

```
1 def count_down(number):  
2     cnt = number  
3     while not(cnt == 0):  
4         cnt = cnt - 1  
5     return "Lift_off!!!"  
6 print(count_down(2))
```

PC	count_down	PC	number	cnt
6	nil	4	2	2

# Runtime example

Functions

TEAM  
INFDEV

Introduction

What is  
abstraction?

Problem  
discussion

General idea

Technical  
details

Assignments

Conclusion

PC	count_down	PC	number	cnt
6	nil	4	2	2

```
1 def count_down(number):  
2     cnt = number  
3     while not(cnt == 0):  
4         cnt = cnt - 1  
5     return "Lift_off!!!"  
6 print(count_down(2))
```

# Runtime example

Functions

TEAM  
INFDEV

Introduction

What is  
abstraction?

Problem  
discussion

General idea

Technical  
details

Assignments

Conclusion

PC	count_down	PC	number	cnt
6	nil	4	2	2

```
1 def count_down(number):  
2     cnt = number  
3     while not(cnt == 0):  
4         cnt = cnt - 1  
5     return "Lift_off!!!"  
6 print(count_down(2))
```

PC	count_down	PC	number	cnt
6	nil	3	2	1

# Runtime example

Functions

TEAM  
INFDEV

Introduction

What is  
abstraction?

Problem  
discussion

General idea

Technical  
details

Assignments

Conclusion

*After a few steps...*

PC	count_down	PC	number	cnt
6	nil	5	2	0

```
1 def count_down(number):  
2     cnt = number  
3     while not(cnt == 0):  
4         cnt = cnt - 1  
5     return "Lift_off!!!"  
6 print(count_down(2))
```

# Runtime example

Functions

TEAM  
INFDEV

Introduction

What is  
abstraction?

Problem  
discussion

General idea

Technical  
details

Assignments

Conclusion

*After a few steps...*

PC	count_down	PC	number	cnt
6	nil	5	2	0

```
1 def count_down(number):  
2     cnt = number  
3     while not(cnt == 0):  
4         cnt = cnt - 1  
5     return "Lift_off!!!"  
6 print(count_down(2))
```

**Do we still need all the local variables of the function?**



*After a few steps...*

PC	count_down	PC	number	cnt
6	nil	5	2	0

```
1 def count_down(number):  
2     cnt = number  
3     while not(cnt == 0):  
4         cnt = cnt - 1  
5     return "Lift_off!!!"  
6 print(count_down(2))
```

**Do we still need all the local variables of the function?  
Where do we put the result?**

# Runtime example

Functions

TEAM  
INFDEV

Introduction

What is  
abstraction?

Problem  
discussion

General idea

Technical  
details

Assignments

Conclusion

*After a few steps...*

PC	count_down	PC	number	cnt
6	nil	5	2	0

```

1 def count_down(number):
2     cnt = number
3     while not(cnt == 0):
4         cnt = cnt - 1
5     return "Lift_off!!!"
6 print(count_down(2))

```

**Do we still need all the local variables of the function?  
Where do we put the result?**

PC	count_down
6	" Lift off!!!"

Functions

TEAM  
INFDEV

Introduction

What is  
abstraction?

Problem  
discussion

General idea

Technical  
details

Assignments

Conclusion

## Syntax and semantics

- We will now describe how Python functions work precisely
- This is a **fundamental** bit of knowledge that determines if you really do learn how to program or not
- This **absolutely requires** a lot of focus to get
- Please panic a bit on the inside

## Subtleties that make functions “fun” to use

- About variables
  - Variables and parameters inside a function have precise **scope** (visibility)
  - Primitive values given as parameters can be **changed only locally** to the function
  - References given as parameters can be **permanently changed** from within the function
  - Global variables defined outside the function may be **read but not changed** from within the function<sup>a</sup>
- About behaviour
  - A function may **call itself**, in a process known as **recursion**
  - A function may **get as parameters and return other functions**, in a process known as **higher order functions**

---

<sup>a</sup>Unless you use some tricks we strongly discourage

Functions

TEAM  
INFDEV

Introduction

What is  
abstraction?

Problem  
discussion

General idea

Technical  
details

Assignments

Conclusion

## Local and global variables (basics of scope)

- The parameters of a function are added to the list of accessible variables
- They are only visible from inside the function
- Global variables are also visible from inside the function

Functions

TEAM  
INFDEV

Introduction

What is  
abstraction?

Problem  
discussion

General idea

Technical  
details

Assignments

Conclusion

## Local and global variables (basics of scope)

- Every call to a function generates a new value of the stack memory  $S$
- This contains (private copy of) all local variables
- The original stack memory (the **global variables**) remains accessible, just read-only

Functions

TEAM  
INFDEV

Introduction

What is  
abstraction?

Problem  
discussion

General idea

Technical  
details

Assignments

Conclusion

## Local and global variables (basics of scope)

- Every call to a function also reserves some special locations in the stack
- The local PC of the function
- The local variables of the function
- The returned value when the function is done

# Locals and globals

Functions

TEAM  
INFDEV

Introduction

What is  
abstraction?

Problem  
discussion

General idea

Technical  
details

Assignments

Conclusion

```
1 x = 1
2
3 def f(z):
4     return x * z
5
6 print(f(10))
7 print(f(30))
8 x = 2
9 print(f(10))
```

## Local and global variables (basics of scope)

- `x` is a global variable, visible outside and inside the function
- `z` is a local variable, visible only inside the function



# Locals and globals

Functions

TEAM  
INFDEV

Introduction

What is  
abstraction?

Problem  
discussion

General idea

Technical  
details

Assignments

Conclusion

```
1 x = 1
2
3 def f(z):
4     return x * z
5
6 print(f(10))
7 print(f(30))
8 x = 2
9 print(f(10))
```

## Local and global variables (basics of scope)

- $x$  is a global variable, visible outside and inside the function
- $z$  is a local variable, visible only inside the function
- **What does this program print?**

# Locals and globals

Functions

TEAM  
INFDEV

Introduction

What is  
abstraction?

Problem  
discussion

General idea

Technical  
details

Assignments

Conclusion

```
1 x = 1
2
3 def f(z):
4     return x * z
5
6 print(f(10))
7 print(f(30))
8 x = 2
9 print(f(10))
```

## Local and global variables (basics of scope)

- $x$  is a global variable, visible outside and inside the function
- $z$  is a local variable, visible only inside the function
- **What does this program print?**
- 10, 30, 20

# Locals and globals

Functions

TEAM  
INFDEV

Introduction

What is  
abstraction?

Problem  
discussion

General idea

Technical  
details

Assignments

Conclusion

PC
1

```
1 x = 1
2
3 def f(z):
4     return x * z
5
6 print(f(10))
7 x = 2
8 print(f(10))
```

# Locals and globals

Functions

TEAM  
INFDEV

Introduction

What is  
abstraction?

Problem  
discussion

General idea

Technical  
details

Assignments

Conclusion

PC
1

```
1 x = 1
2
3 def f(z):
4     return x * z
5
6 print(f(10))
7 x = 2
8 print(f(10))
```

PC	x
6	1

# Locals and globals

Functions

TEAM  
INFDEV

Introduction

What is  
abstraction?

Problem  
discussion

General idea

Technical  
details

Assignments

Conclusion

PC	x
6	1

```
1 x = 1
2
3 def f(z):
4     return x * z
5
6 print(f(10))
7 x = 2
8 print(f(10))
```

# Locals and globals

Functions

TEAM  
INFDEV

Introduction

What is  
abstraction?

Problem  
discussion

General idea

Technical  
details

Assignments

Conclusion

PC	x
6	1

```
1 x = 1
2
3 def f(z):
4     return x * z
5
6 print(f(10))
7 x = 2
8 print(f(10))
```

PC	x	f	PC	z
6	1	nil	4	10

# Locals and globals

Functions

TEAM  
INFDEV

Introduction

What is  
abstraction?

Problem  
discussion

General idea

Technical  
details

Assignments

Conclusion

PC	x	f	PC	z
6	1	nil	4	10

```
1 x = 1
2
3 def f(z):
4     return x * z
5
6 print(f(10))
7 x = 2
8 print(f(10))
```

# Locals and globals

Functions

TEAM  
INFDEV

Introduction

What is  
abstraction?

Problem  
discussion

General idea

Technical  
details

Assignments

Conclusion

PC	x	f	PC	z
6	1	nil	4	10

```
1 x = 1
2
3 def f(z):
4     return x * z
5
6 print(f(10))
7 x = 2
8 print(f(10))
```

PC	x	f
7	1	10



# Locals and globals

Functions

TEAM  
INFDEV

Introduction

What is  
abstraction?

Problem  
discussion

General idea

Technical  
details

Assignments

Conclusion

PC	x	f
7	1	10

```
1 x = 1
2
3 def f(z):
4     return x * z
5
6 print(f(10))
7 x = 2
8 print(f(10))
```

# Locals and globals

Functions

TEAM  
INFDEV

Introduction

What is  
abstraction?

Problem  
discussion

General idea

Technical  
details

Assignments

Conclusion

PC	x	f
7	1	10

```
1 x = 1
2
3 def f(z):
4     return x * z
5
6 print(f(10))
7 x = 2
8 print(f(10))
```

PC	x
8	2

# Locals and globals

Functions

TEAM  
INFDEV

Introduction

What is  
abstraction?

Problem  
discussion

General idea

Technical  
details

Assignments

Conclusion

PC	x	f	PC	z
8	2	nil	4	10

```
1 x = 1
2
3 def f(z):
4     return x * z
5
6 print(f(10))
7 x = 2
8 print(f(10))
```

# Locals and globals

Functions

TEAM  
INFDEV

Introduction

What is  
abstraction?

Problem  
discussion

General idea

Technical  
details

Assignments

Conclusion

PC	x	f	PC	z
8	2	nil	4	10

```
1 x = 1
2
3 def f(z):
4     return x * z
5
6 print(f(10))
7 x = 2
8 print(f(10))
```

PC	x	f
8	2	20

# Locals and globals

Functions

TEAM  
INFDEV

Introduction

What is  
abstraction?

Problem  
discussion

General idea

Technical  
details

Assignments

Conclusion

```
1 x = 1
2
3 def f(z):
4     return x * z
5
6 print(f(10))
7 x = 2
8 print(f(10))
9 print(z)
```

## Local and global variables (basics of scope)

- `x` is a global variable, visible outside and inside the function
- `z` is a local variable, visible only inside the function

# Locals and globals

Functions

TEAM  
INFDEV

Introduction

What is  
abstraction?

Problem  
discussion

General idea

Technical  
details

Assignments

Conclusion

```
1 x = 1
2
3 def f(z):
4     return x * z
5
6 print(f(10))
7 x = 2
8 print(f(10))
9 print(z)
```

## Local and global variables (basics of scope)

- `x` is a global variable, visible outside and inside the function
- `z` is a local variable, visible only inside the function
- **What does this program do?**

# Locals and globals

Functions

TEAM  
INFDEV

Introduction

What is  
abstraction?

Problem  
discussion

General idea

Technical  
details

Assignments

Conclusion

```
1 x = 1
2
3 def f(z):
4     return x * z
5
6 print(f(10))
7 x = 2
8 print(f(10))
9 print(z)
```

## Local and global variables (basics of scope)

- `x` is a global variable, visible outside and inside the function
- `z` is a local variable, visible only inside the function
- **What does this program do?**
- Crash with `NameError: name 'z' is not defined`

# Locals and globals

Functions

TEAM  
INFDEV

Introduction

What is  
abstraction?

Problem  
discussion

General idea

Technical  
details

Assignments

Conclusion

```
1 def f(z):  
2     z = z + 1  
3     return z * 2  
4  
5 print(f(10))  
6 print(f(30))
```

## Local and global variables (basics of scope)

- `z` is a local variable, visible only inside the function



```
1 def f(z):  
2     z = z + 1  
3     return z * 2  
4  
5 print(f(10))  
6 print(f(30))
```

## Local and global variables (basics of scope)

- `z` is a local variable, visible only inside the function
- **What does this program print?**

Functions

TEAM  
INFDEV

Introduction

What is  
abstraction?

Problem  
discussion

General idea

Technical  
details

Assignments

Conclusion

```
1 def f(z):  
2     z = z + 1  
3     return z * 2  
4  
5 print(f(10))  
6 print(f(30))
```

## Local and global variables (basics of scope)

- $z$  is a local variable, visible only inside the function
- **What does this program print?**
- 22, 62

Functions

TEAM  
INFDEV

Introduction

What is  
abstraction?

Problem  
discussion

General idea

Technical  
details

Assignments

Conclusion

## Shadowing

- The parameters of a function have priority over globals
- They supersede global variables of the same name

```
1 x = 1
2
3 def f(x):
4     return x * 2
5
6 print(f(10))
7 print(f(20))
```

## Shadowing

- `x` is a global variable, potentially visible inside the function
- `x` is also a local variable of the function, which has priority over the global `x`

```
1 x = 1
2
3 def f(x):
4     return x * 2
5
6 print(f(10))
7 print(f(20))
```

## Shadowing

- $x$  is a global variable, potentially visible inside the function
- $x$  is also a local variable of the function, which has priority over the global  $x$
- **What does this program print?**

```
1 x = 1
2
3 def f(x):
4     return x * 2
5
6 print(f(10))
7 print(f(20))
```

## Shadowing

- $x$  is a global variable, potentially visible inside the function
- $x$  is also a local variable of the function, which has priority over the global  $x$
- **What does this program print?**
- 20, 40

Functions

TEAM  
INFDEV

Introduction

What is  
abstraction?

Problem  
discussion

General idea

Technical  
details

Assignments

Conclusion

PC	x
6	1

```
1 x = 1
2
3 def f(x):
4     return x * 2
5
6 print(f(10))
7 print(f(20))
```

Functions

TEAM  
INFDEV

Introduction

What is  
abstraction?

Problem  
discussion

General idea

Technical  
details

Assignments

Conclusion

PC	x
6	1

```
1 x = 1
2
3 def f(x):
4     return x * 2
5
6 print(f(10))
7 print(f(20))
```

PC	x	f	PC	x
6	1	nil	4	10



Functions

TEAM  
INFDEV

Introduction

What is  
abstraction?

Problem  
discussion

General idea

Technical  
details

Assignments

Conclusion

PC	x	f	PC	x
6	1	nil	4	10

```
1 x = 1
2
3 def f(x):
4     return x * 2
5
6 print(f(10))
7 print(f(20))
```

# Shadowing

Functions

TEAM  
INFDEV

Introduction

What is  
abstraction?

Problem  
discussion

General idea

Technical  
details

Assignments

Conclusion

PC	x	f	PC	x
6	1	nil	4	10

```
1 x = 1
2
3 def f(x):
4     return x * 2
5
6 print(f(10))
7 print(f(20))
```

PC	x	f
7	1	20

Functions

TEAM  
INFDEV

Introduction

What is  
abstraction?

Problem  
discussion

General idea

Technical  
details

Assignments

Conclusion

PC	x	f
7	1	20

```
1 x = 1
2
3 def f(x):
4     return x * 2
5
6 print(f(10))
7 print(f(20))
```

Functions

TEAM  
INFDEV

Introduction

What is  
abstraction?

Problem  
discussion

General idea

Technical  
details

Assignments

Conclusion

PC	x	f
7	1	20

```
1 x = 1
2
3 def f(x):
4     return x * 2
5
6 print(f(10))
7 print(f(20))
```

PC	x	f	PC	x
7	1	nil	4	20

Functions

TEAM  
INFDEV

Introduction

What is  
abstraction?

Problem  
discussion

General idea

Technical  
details

Assignments

Conclusion

PC	x	f	PC	x
7	1	nil	4	20

```
1 x = 1
2
3 def f(x):
4     return x * 2
5
6 print(f(10))
7 print(f(20))
```

# Shadowing

Functions

TEAM  
INFDEV

Introduction

What is  
abstraction?

Problem  
discussion

General idea

Technical  
details

Assignments

Conclusion

PC	x	f	PC	x
7	1	nil	4	20

```
1 x = 1
2
3 def f(x):
4     return x * 2
5
6 print(f(10))
7 print(f(20))
```

PC	x	f
8	1	40

Functions

TEAM  
INFDEV

Introduction

What is  
abstraction?

Problem  
discussion

General idea

Technical  
details

Assignments

Conclusion

# Assignments

Functions

TEAM  
INFDEV

Introduction

What is  
abstraction?

Problem  
discussion

General idea

Technical  
details

Assignments

Conclusion

## Build and test, on paper...

- A function `add` that increments a given number by a fixed value:
  - `add(1, 5) -> 6`
- A function `isEven` that returns `True` if a given number is even, `False` otherwise:
  - `isEven(6) -> True`
- A function `sum_between` that returns the sum of all integers between two given integer numbers:
  - `sum_between(2, 5) -> 14`



Functions

TEAM  
INFDEV

Introduction

What is  
abstraction?

Problem  
discussion

General idea

Technical  
details

Assignments

Conclusion

# Conclusion

Functions

TEAM  
INFDEV

Introduction

What is  
abstraction?

Problem  
discussion

General idea

Technical  
details

Assignments

Conclusion

## Lecture topics

- Often, user code needs to perform operations that are similar to each other
- Through the mechanism of function definition, we can recycle code
- Functions can encode algorithms in many way
  - Simple code abstractions to avoid repetition

Functions

TEAM  
INFDEV

Introduction

What is  
abstraction?

Problem  
discussion

General idea

Technical  
details

Assignments

Conclusion

The best of luck, and thanks for the  
attention!