

WELCOME TO PYTHON!

VARIABLES AND DATA TYPES

FUNCTIONS

COLLECTIONS

CONDITIONALS

LOOPS

DEBUGGING AND ERROR HANDLING

The slides are meant as visual support for the lecture.
They are neither a documentation nor a script.

Please do not print the slides.

Comments and feedback at n.meseth@hs-osnabrueck.de

WELCOME TO PYTHON!

[BACK](#)

```
name = input("What's your name? ")
```

```
print(f"Hello {name}")
```

functions or commands

built-in functions

```
print()  
input()  
...
```

functions from built-in modules

```
math.sqrt()  
time.sleep()  
sys.exit()  
...
```

external modules

```
requests.get()  
...
```

comments

```
# Ask the user for their name
name = input("What's your name? ")

# Greet the user
print(f"Hello {name}")
```



```
# Ask the user for their name
name = input("What's your name? ")

print(f"Hello {name}") # Greet the user
```

```
'''
```

```
a multi-line comment  
for longer descriptions  
'''
```

```
print("hello, world")
```

arguments / parameter

```
# Ask the user for their name
name = input("What's your name? ")

# Greet the user
print(f"Hello {name}")
```

bugs

syntax errors

runtime errors

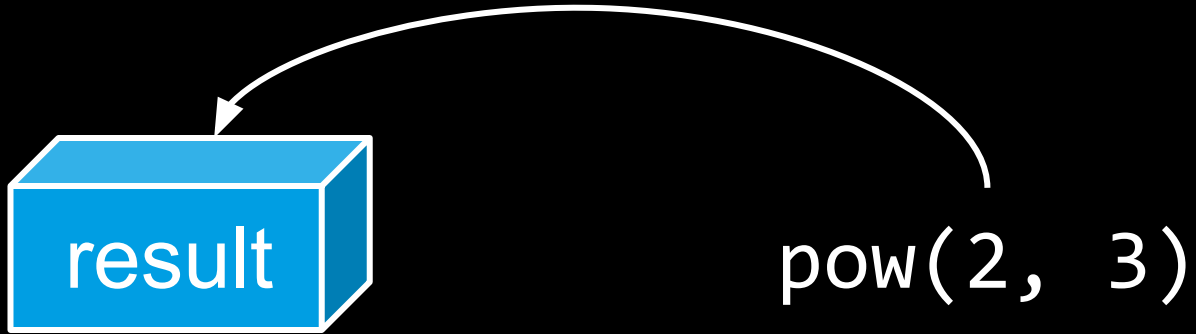
function's return values

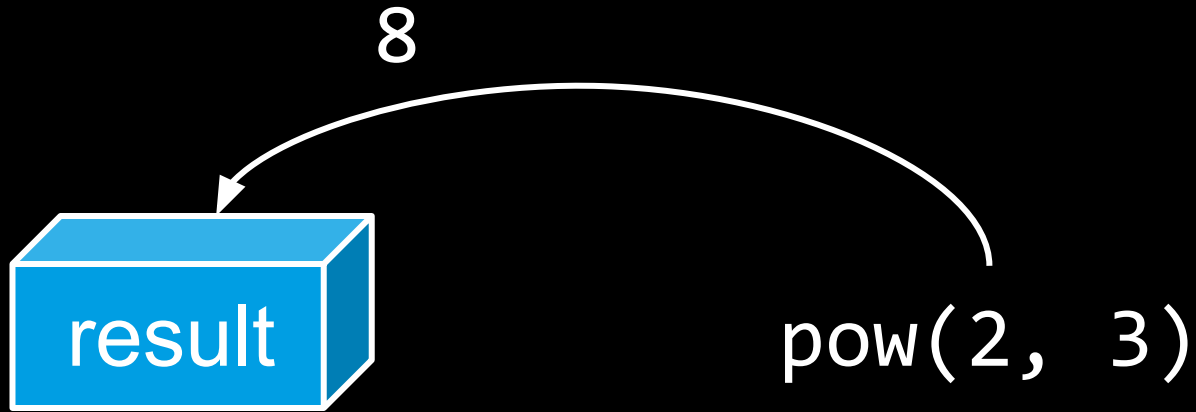

```
result = pow(2, 3)
```

VARIABLES AND DATA TYPES

[BACK](#)







```
exp = 4  
result = pow(2, exp)
```

 = 4

result = pow(2, )

```
exp = 4  
result = pow(2, exp)  
print(result)
```


constants

PI = 3.14159

UID = "ZeW"

naming variables

use_underscores_for_spaces
start_with_small_letter
only_0123456789_and_letters
english_and_speaking_names

operators

math

$$5 + 5$$

$$9 - 8$$

$$2 / 1$$

$$6 * 7$$

$$5 // 2$$

$$10 \% 3$$

$$2^{**}3$$

logic

`2 == 1`

`2*2 > 1+3`

`2*2 >= 1+3`

`"A" < "B"`

`"A" < "B" and 2 == 1`

`"A" < "B" or 2 == 1`

strings

== != > < >= <=

+

*

in / not in

[1] / [1:4]

strip()

capitalize()

title()

data types

integer

integer
float } numeric

integer
float
boolean

}

numeric

integer
float
boolean
string

} numeric

format strings

```
print(f"Hello {name}")
```

comments

step 1: determine exponent

step 2: calculate power

step 3: print result

problem solving → problem decomposition

step 1: determine exponent

step 2: calculate power

step 3: print result

```
# step 1: determine exponent
```

```
exp = 4
```

```
# step 2: calculate power
```

```
# step 3: print result
```

```
# step 1: determine exponent
```

```
exp = 4
```

```
# step 2: calculate power
```

```
result = pow(2, exp)
```

```
# step 3: print result
```



```
# step 1: determine exponent
```

```
exp = 4
```

```
# step 2: calculate power
```

```
result = pow(2, exp)
```

```
# step 3: print result
```

```
print(result)
```

FUNCTIONS

[BACK](#)

create functions

```
def greet():  
    print("hello")
```

parameters

```
def greet(name):  
    print(f"hello {name}")
```

parameter default values

```
def greet(name="world"):
    print(f"hello {name}")
```


returning results

returning results

```
def make_greeting(name):  
    greeting = f"hello {name}"  
    return greeting
```

calling functions

```
greeting = make_greeting("Mika")
```

variable to store return value



```
greeting = make_greeting("Mika")
```

COLLECTIONS

[BACK](#)

lists


```
fruits = ["apple", "banana", "cherry"]
```

```
fruits = ["apple", "banana", "cherry"]  
fruits[0]      # apple
```

```
fruits = ["apple", "banana", "cherry"]  
fruits[0]    # apple  
fruits[1]    # banana
```

```
fruits = ["apple", "banana", "cherry"]  
fruits[0]    # apple  
fruits[1]    # banana  
fruits[2]    # cherry
```

```
fruits = ["apple", "banana", "cherry"]  
fruits[0]    # apple  
fruits[1]    # banana  
fruits[2]    # cherry  
fruits[1:2]  # ["banana", "cherry"]
```

list operations

```
fruits.append("grape")
```

```
fruits.append("grape")
```

```
fruits.insert(1, "strawberry")
```



```
fruits.append("grape")
```

```
fruits.insert(1, "strawberry")
```

```
fruits.pop()
```

```
fruits.append("grape")
```

```
fruits.insert(1, "strawberry")
```

```
fruits.pop()
```

```
len(fruits)
```

```
fruits.append("grape")  
fruits.insert(1, "strawberry")  
fruits.pop()  
len(fruits)  
  
for fruit in fruits:  
    print(fruit) # print every fruit
```

CONDITIONALS

[BACK](#)

```
if <condition>:
```

```
...
```

```
if <condition>:
```

```
    ...
```

```
else:
```

```
    ...
```

```
if <condition>:  
    ...  
elif <condition>:  
    ...
```

LOOPS

[BACK](#)

while loop

for loop

DEBUGGING AND ERROR HANDLING