



Curtin University

Building Blocks

ISYS2001, School of Marketing and Management

***ELECTRONIC WARNING NOTICE FOR COPYRIGHT
STATUTORY LICENCES***

WARNING

This material has been reproduced and communicated to you by or on behalf of **Curtin University** in accordance with section 113P of the *Copyright Act 1968 (the Act)*

The material in this communication may be subject to copyright under the Act. Any further reproduction or communication of this material by you may be the subject of copyright protection under the Act.

Do not remove this notice.



I acknowledge the traditional custodians of
the land on which I work and live, and
recognise their continuing connection to land,
water and community. I pay respect to elders
past, present and emerging.



Previously

- Data = Value + Type
- Algorithms
- Control Structures
 - Sequence
 - Selection
 - Repetition
- Pseudocode
- Six things a computer can do (fast)



Today

- Calling Functions
- Using Modules
- Creating Functions



```
print("Hello, world!")
```



*We call or
invoke a
function by
using its name
in code*



```
print("Hello, world!")
```

*We call or
invoke a
function by
using its name
in code*

*The name is
followed by a
parenthesis*



```
print("Hello, world!")
```


*We call or
invoke a
function by
using its name
in code*

*The name is
followed by a
parenthesis*

*We provide zero
or more inputs
We "pass
arguments to
the function"*

`print("Hello, world!")`

*We call or
invoke a
function by
using its name
in code*

*The name is
followed by a
parenthesis*

*We provide zero
or more inputs
We "pass
arguments to
the function"*

*Finally we end
with a
parenthesis*

`print("Hello, world!")`

*Some functions
'return' a value*

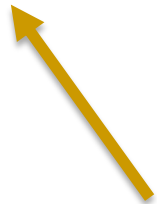


```
input("What is your name?")
```

*Some functions
'return' a value*



```
name = input("What is your name?")
```



*Save the return
value in a
variable*

Calling Functions

- Call/invoke using name
- Provide zero or more arguments (inputs)
- Save the return value to a variable (optional)



Modules

- Builtins
- 3rd Party



Built-In Functions

<code>input()</code>	Allowing user input
<code>int()</code>	Returns an integer number
<code>len()</code>	Returns the length of an object
<code>print()</code>	Prints to the standard output device
<code>type()</code>	Returns the type of an object

Many more

https://www.w3schools.com/python/python_ref_functions.asp

Modules

Import keyword



```
import random
```


Modules

Import keyword



*The module we
want to use*



```
import random
```

Modules

Import keyword

*The module we
want to use*

`import random`

`random.choice([1,2,3])`

*Call the choice()
function from the
random module*

Modules

Import keyword

*The module we
want to use*

`import random`

`random.choice([1,2,3])`

*Call the choice()
function from the
random module*

Module Examples

- pillow for **image manipulation**
- openCV for **computer vision**
- scikit-learn for **machine learning**
- pandas for **data analysis**
- keras for **deep learning**
- numpy for **numerical computations**
- matplotlib for **plotting**
- **And many, many others!**



First import the module

Import keyword



Module



```
import matplotlib.pyplot
```

Use the module

Import keyword



Module



```
import matplotlib.pyplot
```

```
matplotlib.pyplot.plot([1,3,5], [1,9,25])
```

*Module, followed by
dot, followed by
function*



Convenience

Import keyword



Module



```
import matplotlib.pyplot as plt
```

Convenience

Import keyword



Module



```
import matplotlib.pyplot as plt
```


Convenience

Import keyword



Module



*'as' keyword followed
by an alias*



```
import matplotlib.pyplot as plt
```

Convenience

Import keyword

Module

*'as' keyword followed
by an alias*

`import matplotlib.pyplot as plt`

`plt.plot([1,3,5], [1,9,25])`

*Alias, followed by dot,
followed by function*

Modules

- Builtins not enough
- Designed & tested by someone else
- Contain lots of functions
- Use *import* keyword
- Use module name as prefix to function name
- Can alias module name



Define functions

- Header
- Body



```
def calculate_price(price, discount):  
    amount_saved = price * (discount/100)  
    sale_price = price – amount_saved  
    gst = sale_price * 0.1  
    total_price = sale_price + gst  
    return total_price
```



*Start with
def keyword, and a
name*

```
def calculate_price(price, discount):  
    amount_saved = price * (discount/100)  
    sale_price = price – amount_saved  
    gst = sale_price * 0.1  
    total_price = sale_price + gst  
    return total_price
```

*Start with
def keyword, and a
name*

*We have zero or
more inputs
(parameters)*

```
def calculate_price(price, discount):  
    amount_saved = price * (discount/100)  
    sale_price = price - amount_saved  
    gst = sale_price * 0.1  
    total_price = sale_price + gst  
    return total_price
```

*Start with
def keyword, and a
name*

*We have zero or
more inputs
(parameters)*

*Colon start
block of code*

```
def calculate_price(price, discount):  
    amount_saved = price * (discount/100)  
    sale_price = price - amount_saved  
    gst = sale_price * 0.1  
    total_price = sale_price + gst  
    return total_price
```


*Start with
def keyword, and a
name*

*We have zero or
more inputs
(parameters)*

*Colon start
block of code*

```
def calculate_price(price, discount):  
    amount_saved = price * (discount/100)  
    sale_price = price - amount_saved  
    gst = sale_price * 0.1  
    total_price = sale_price + gst  
    return total_price
```

*Start with
def keyword, and a
name*

*We have zero or
more inputs
(parameters)*

*Colon start
block of code*

```
def calculate_price(price, discount):
```

```
    amount_saved = price * (discount/100)
```

```
    sale_price = price - amount_saved
```

```
    gst = sale_price * 0.1
```

```
    total_price = sale_price + gst
```

```
    return total_price
```

*Here is the
block of code*

*Start with
def keyword, and a
name*

*We have zero or
more inputs
(parameters)*

*Colon start
block of code*

```
def calculate_price(price, discount):
```

```
    amount_saved = price * (discount/100)
```

```
    sale_price = price - amount_saved
```

```
    gst = sale_price * 0.1
```

```
    total_price = sale_price + gst
```

```
    return total_price
```

*Block of code is
indented*

*Here is the
block of code*

*Start with
def keyword, and a
name*

*We have zero or
more inputs
(parameters)*

*Colon start
block of code*

```
def calculate_price(price, discount):
```

```
    amount_saved = price * (discount/100)
```

```
    sale_price = price - amount_saved
```

```
    gst = sale_price * 0.1
```

```
    total_price = sale_price + gst
```

```
    return total_price
```

*Block of code is
indented*

Return a value

*Here is the
block of code*

*Function
header*

```
def calculate_price(price, discount):
```

```
    amount_saved = price * (discount/100)
```

```
    sale_price = price – amount_saved
```

```
    gst = sale_price * 0.1
```

```
    total_price = sale_price + gst
```

```
    return total_price
```

*Function
header*

```
def calculate_price(price, discount):
```

*Function
body*

```
    amount_saved = price * (discount/100)
    sale_price = price - amount_saved
    gst = sale_price * 0.1
    total_price = sale_price + gst
    return total_price
```

Define functions

- Header

- Use def keyword

- List parameters (input)

- Colon define a code block

- Body

- Set of statement

- Code block indented

- Optionally returns a value

Today

- Calling Functions
- Using Modules
- Creating Functions



Next time

- Getting Help
- Handling Exceptions
- Validating Input
- File Input/Output

