

TITLE

1st Lec

: Comments / ignored text

print() : becomes output in terminal

Variable_name = ...

Casting → str() make into string / text

int() become whole number

type() identifies data type

list() makes it a list

Formatted string f" text {Variable} text"

so u dont get confused by "space" + Variable

or print(f" {variable name} = ?") so u get name + answer

2nd Lec

input("prompt...?") : user can type smth after seeing question prompt

Data types

∞

int

"

str

T/F

boolean

using capital letter

None

True

False

a == b : ask computer if a = b → says True / False

a != b : ask if it is NOT true

not(boolean) → not(False) = true

not(False) = true

not(True) = False

if condition true :

action

action

elif

alternat condition :

elif

action alt condition 2 :

else :

action

action

If the condition is met then the indented actions will run

If unmet it will do the other thing

if _ not in / in _ :

cond1 and cond2 - checks both conditions, BOTH must be true

cond1 or cond2 - checks if either is true to give back true

alphabet's numerical value = ascii table

"A" = 65

"B" = 66

"Maddy" < "Sim"

bc S is later than M

"Sim" < "Simon"

"string"[n] takes out n-1 character of the string

```
print(favourite_animal[0]) # first letter
print(favourite_animal[-1]) # last letter
print(favourite_animal[1:-1]) # all the middle letters
```

n. letter of string in "a b c d" : sees if that letter is any of the mentioned
"bl" in "bla" → Trueprint("blablabla\nblabla") = blablabla /n to new line (enter)
blabla

if u rly want / u have to "//"

len() length / find out how many characters in a string

TOPIC

DATE

Methods

string/variable_name. count('x') : count how many x in that string
more important \leftarrow method that serves it
string/variable_name. replace('x', 'y') : replace all "x" in string to "y"
startswith('') : check if string starts with —

many more string methods!

W3Schools for more functions & methods!

% modulus : $8 \% 2 = 0$ returns remainder after division

string_name. split() : split a string into substrings

$zid, domain = email.split('@')$
makes 2 variables between split

function : on all data types
method : specific data types

rename symbol to rename a variable name in VS code



loop method 1

while < condition is true > :

do this action

update condition

as long as still true, will keep repeating action till false

```
# Step 1: get input AND initialisation
password = input("Enter your password: ")
# Step 2: put ur loop in
while len(password) < 8:
    print("Your password must be at least 8 characters long.") # if u stop here ur user doesn't get a chance
    # Step 3: opportunity to change the condition
    password = input("Enter your password: ")
print("Excellent password!") # Step 4: loop is now broken bc false was returned, next step commence
```

print("___", end=" ") put nothing after ur string so new line isn't made

loop method 2

for

action

in

: {

for

letter

in string-name:

print(letter)

range(x, y) = x up to y-1

ex: range(1, 5) = 1, 2, 3, 4

zid_dict = { 'course code': { 'time': 'time', 'room': 'location' }

UNSW courses

```
{
  COMP1010: {
    "code": "COMP1010"
    "name": "art..."
    "tut": [ {time: - room: -}
             {time: - room: -} ]
    "enrollm": [ {zid: - tut: -} ]
  },
  FIN1613: {
    ...
  }
}
```

[zid : tut :

zid: tut :

]

TOPIC

3- Functions (↓ methods)

DATE

 Python

def action_name (parameter 1, parameter 2 ...) :
 action / instructions

make your own DIY function

```
def get_box_string(height, width):  
    top_bottom = "~" * width + "~" * height + "~" * width  
    middle = "|" * width + "|" * height + "|" * width  
    my_box = top_bottom + middle + height * top_bottom  
    return my_box  
  
box1 = get_box_string(1, 1)  
print(box1)  
  
print(get_box_string(2, 2))  
print(get_box_string(3, 3))  
print(get_box_string(3, 1))  
print(get_box_string(2, 3))
```

4 - Collections



list_name = []

makes a list, multi things under the same variable names

more methods in W3schools List methods

- list_name.append(" ") : adds additional value at the end
- list_name.clear() deletes everything in list
- list_name.sort(reverse = True) will reverse alphabetical sorting
- list_name.insert(i, " ")
- index[n] = number of item on the list starting from 0

```
print(cafes[1])
cafes[1] = "Grumpy Baker"
print(cafes)
```

dictionary_name.keys() : returns the keys / values
dictionary_name.values() from dictionary

set() : deletes duplicates in a set

all_list = list 1 + list 2 joins 2 lists

* Dictionaries

Terminology:

- Key (eg the word)
- Value (eg the meaning)
- Each key has exactly one entry (if you look up a word, you aren't going to get different results each time you look it up)
- Different keys can have the same value (if you look up different words, they might have the same meaning)

1 key only to 1 value
diff keys can have same value, ex: student & pupil = same definition / means the same thing

key = string to prompt value / keyword

put in key, receive associated value

ex: key value
↓ ↓
student zid list of courses & grades achieved for it

dictionary_name = { "key1": "value1",
"key2": "value2",
... }

get value of a key → dictionary_name["key1"]

adding things to a dictionary

```
task = {}
task["name"] = "Brush teeth"
task["duration"] = "2"
print(task)
```

dictionary_name["newkey"] = "new value"

changing value of existing key: dictionary_name["existingkey"] = "diff value"

erase a key → `dictionary-name.pop("key-to-remove")`
 returns value of a key → `dictionary-name.get("key")`

method 1

returning keys = value = from

```
list_of_keys = list(task.keys())
print(list_of_keys)
for key in list_of_keys:
    print(f"{key}")
    associated_value = task[key]
    print(f"{associated_value}")
    print()
```

task _{key}	value
"name"	"brush teeth"
"due"	"5 th Mar"
"duration"	2
"unit"	"mins"

list of keys
 0 "name"
 1 "due"
 2 "duration"
 3 "unit"

for loop
 keys

"name"

assoc_value

`task["name"]`

method 2

ask for items

```
#print(list(task.items()))
for key,value in task.items():
    #print(f"{key}")
    #print(f"{value}")
    print(f"{key} : {value}")
```

`dictionary-name.items()` : Split key & value pairs from each other to be a list

for entry in task.items(): → for key, value in task.items():
 tells python that every entry consist of key, value

TOPIC

DATE

example !

```

# You find out the e-mail addresses of 4 different students:
emails = ["12345678@student.unsw.edu.au",
          "12345678@student.unsw.edu.au",
          "12345678@student.unsw.edu.au",
          "12345678@student.unsw.edu.au"]

# You can build a dictionary containing the zids as keys and the e-mail addresses as values using their zid to find their email

# Create an empty dictionary
zid_to_email = {}

# Dictionary is empty, now go through all emails and add each one to dictionary
for email in emails:
    # print(email)
    zid = email[:8]
    # print(zid)
    zid_to_email[zid] = email

# print(zid_to_email)

for key, value in zid_to_email.items():
    print("{}: {}".format(key, value))
  
```

- ① Make empty dictionary tinggal isi
- ② each item in emails makes a new "zid=" takes first 8 characters puts in "key zid" w/ "value email" into dictionary until all emails r inputted
- ③ takes every pairing in zid-to-email & breaks it up

Complex Data Types

assert true / false eval : make sure
 assert 1 = 1 (nothing happens)
 assert 1 = 2 error & stops the program

datatype = None

House Hunting 🏠

① Data bank

houses (list)

dictionaries	
key	value
{ address : 42 WW, S bed : 3 bath : 2 features : [-, -, -] }	{ address : 1 HS, K bed : 2 bath : 1 features : [,] }
{ address : "123 FS, M" bed : 6 bath : " feature : [-, -] }	{ address : "109 kA, k" bedrooms : 10 bath : 7 features : [-, -] }

② function to check if features are there

def has_all_features (v1 ft_wanted, v2 ft_has)

result = True

for ft_i - want in ft_wanted :

if ft_i - want not in ft_has :

result = False

return result (T / F)

ft_i - want ex: garage, pool

if garage not in ft_has
= False

③ find_houses (list of houses, n. bed, n. bath, [features_wanted])

Viable_addresses = []

for house in list of houses

address = house ["address"] = value of ["key"] from dict.

if n.bed >= bed: ✓

if n.bath >= bath: ✓

if has_all_features (features_want, features) ✓

Viable_addresses.append (address)

dictionary [key1][key2] = value



import webbrowser : start getting URLs

2. Web browser vs Web server

Browser

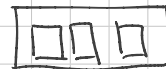
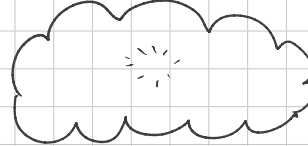
User's end: google, firefox

🔍 Cat pics



Server codes written here :)

google FR



generates all cat pic related data on server

① send request
"cat pics"

② response
sends back

Elements

open tag <html>
 <head>
 </head>
 <body>
 </body>
close tag </html>

<!-- comment -->

* Headings

<h1> - <h6> heading fonts

** ** bold

* Paragraphs

<p> paragraph 1 </p>

bla bla bla
bla bla

* Line breaks (separate line but not new paragraphs)

 blabla
blabla

* Images (from ur local server / internet)

online

<!-- from: credits > use copy image address

local

save as image in the VS code folder

* Links

Text to display / Links thumbnail
opens new tab

* Lists

a) ordered
numbered

 item 1
 item 2

b) unordered
(bullet points)

u can change the number/pt styles later

* Table

| table data | Mon | Tue | Wed | Thur |
|------------|-----|----------|-----|------|
| 9:00 | | COMPIO10 | | |
| 10:00 | | | | |
| 11:00 | | | | |
| 12:00 | | | | |

table row

```

<table>
  <tr>
    <td></td> - 1st data empty
    <td> Mon </td>
    <td> Tue </td>...
  </tr>
  <tr>
    <td> 9:00 </td>
    <td> </td>
    <td> COMPIO10 </td>
  </tr>

```

~ Forms (interactable)

<form>

every subform

</form>

* Input

<form>

```

<input type="text">
<input type="month"> etc 500 many types
<input type="Submit" value="Go back">

```

Submit!

goes to another page what u want the button to say

after inputting, everything in form gets packed in one & sent off to server.
put all in one form so it carries all the info.

radio buttons

```

<input type="radio" name="fav-color" value="Red">
<input type="radio" name="fav-color" value="Blue">
<input type="radio" name="yurr" value="Green">

```

same name group

~ Labels

note next to input

ex:

Username:

label

input type="username"

<label> Username: </label> <form></form>

? If I click label my cursor can jump to the input box

<label for="group-name"> ... </label> <input type="" id="group-name">

Dropdown (tons of options ex: which country?)

Radio button (1 out of some options)

Checkbox (several options u can choose)

building a web server on my own computer

! ? help with my Flask & pyhtml directory pls

flask is a library from other

```

1-template.py > ...
1 import pyhtml as p
2 from flask import Flask
3
4 app = Flask(__name__)
5
6
7 @app.get("/")
8 def home():
9     return "Hello"
10
11
12 if __name__ == "__main__":
13     app.run(port=5001)
14

```

import extra python stuff we need for server
library by other ppl help us build a webserver ez

Create Flask Web Server in variable "app"
flask will find the Python project name

which webpage?
pages in website (nomer blkg yg ganti tiap ganti page)

function that runs when webpage is navigated to. all this function displayed in HTML

when user goes to this page in ur web, run this function

String "Hello" is returned in HTML

if this file is main file of app, run it on this network - just copas aja.
runs web server created on Line 6

import pyhtml as p

p. Command activates pyhtml when "p" typed

```

remember, "
p.html (= <html>
p.body (p.ol (<ol>
p.li ( ),
p.li ( ),
</ol> )
</html>
p.h1, p.p
p.form ( p.input ( type=" " , value=" " )

```

Make it change page after pressing button

p.form (action = "/page-name") ()

or

p.form (p.input (type=" " value=" " formaction = "/page name"))

HTTP methods

1 @app.get ("/ ") : simple , go to URL

2 @app.post ("/ ") : sending a form's data elsewhere
u can only go here if u send us a form

print (request.form)

↳ all requests / form answers

adjust template to from flask import Flask, request, redirect

3. @app.route ('/page', methods = ['GET', 'POST'])

give the input form a variable name

p.input (type=" " , value=" " , name=" something")

↓ sends as dictionary { 'something' : ' answer' }

app.get answer_value = request.form [' something']

OR

request.form.get (' something' , " default value if something's value doesn't exist")

p.br line breaks

↳ blank text box is fine if input empty

POST → request.form.get (" textbox name" , " ") get me value from the key, password,
if not there just empty string

↳ going back to get pages from post pages

make a dropdown selection list

| | |
|----|---|
| \$ | ▼ |
| \$ | |
| \$ | |

python

```
p.select(name="price")(
    p.option(value="$")("$"),
    p.option(value="$")("$"),
    p.option(value="$")("$"),
)
```

make checkbox option in W8 lab checkbox_example.py

request.form.getlist

@ app.route('/', methods = ['GET', 'POST']) - get there by typing URL / submit form

TOPIC

7

- Look up other libraries

DATE

datetime & random

```
rickroll [ import webbrowser  
          webbrowser.open (" https:// youtube/ dQw4w9WgXcQ ")
```

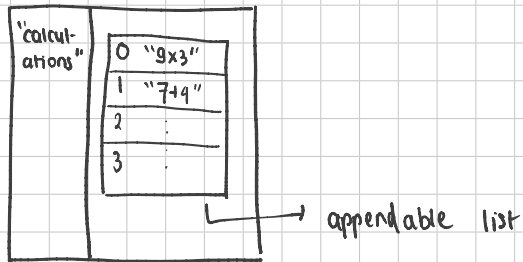
```
request_form.get ("textbox-name", "x")
```

W7 Thur Lec 1

if nothing in there, it will return "x"

from flask import Flask, Session

Session stores cookies as dictionary



u need a secret key encryption
app.config['SECRET_KEY'] = 'blablabla'

secretkey to encrypt data

ex: username: Sim
password: Sprite

cookie
logged in details
server checks

browser

if unencrypted
user can change cookie content
& hacks

if encrypted...

Sim → Tin

If "key-name" not in Session:
session["key-name"] = []

Session["key-name"].append(variable_name)

Session.modified = True updates session contents

```
# If no user is logged in, add a Login button, else add a Logout button
if session.get("username", "") == "":
    list_of_nav_buttons.append(
        p.input(type="submit", value="Login", formaction="/login")
    )
else:
    list_of_nav_buttons.append(
        p.p(f"Users: {session['username']}"),
        p.input(type="submit", value="Logout", formaction="/logout")
    )
```

if session.get("username", "") == ""
→ if entered username is not a key in the cookie,
give me empty

session.get("key") → reads only
session[key] = value → writes key + value
session.clear → delete everything from cookie

"key" in session → check if key exist

Session.permanent = True → cookie survives browser reset

session.get('key') VS session["key"]
None if key doesn't exist error if key doesn't exist

session.get("key", "sub value if key doesn't exist")

like a cookie but more universal
 from jsonhelper import *
 (template file) (all functions)

can combine diff users' data
 can't be cleaned by users
 doesn't have to be in webapp, can be in files

JSON integrates well w/ python

* figure how to data.json save in right path

python3 filename.py at integrated terminal

Read

vs

Write

accessing earlier info

adding text on file

1. open "r"
2. Read JSON data
3. Convert JSON → Python
4. Close file

1. Open "w"
2. Python → JSON convert
3. write JSON data
4. Close file

jsonhelpers.py

1. create-if-not-exist (filename, empty-data)

ex: [] or {}

2. write-json-file (filename, info)

3. read-json-file (filename)

filename → "name.json"

then tell it where to save (folder)

.py → right click → open in integrated terminal

→ python3 name.py → enter

TOPIC

Pypi & APIs

DATE

Pypi.org (outer library)

ex: pip install wikipedia
in pyproject.toml
add
"wikipedia >= 1.4.0"
copies the lib's code?

API supply dynamic data
rapidapi.com
transportation API ?? $\frac{1}{2}$

each selector has list of property value pairs in curly braces
 selector = html tag
 body { background-color: lightgrey; }

make paragraph red & center

```
p {
  text-align: center;
  color: red;
}
```

Connect CSS document to HTML

In html head, create link to ur CSS document

```
<link rel="stylesheet" href="static/my-style.css">
```

In pyhtml

```
response = html(
    head(
        link(rel="stylesheet", href="static/my-style.css") ) )
```

↓

```
p.head(
    p.title("Demo"),
    p.link(rel="stylesheet", href="static/style.css"))
```

its a stylesheet find "static" folder, use file.css

In CSS file:

```
ex: h1 {
  color: #nnnnnn;
  color: purple; }
```

in the webpage, R click > Inspect > click the text block to get values

```
h1 {
  display: block;
  font-size: 2em; 2em / px / %
  margin-block-start: 0.67em
  margins
  font-weight: bold;
  unicode-bidi: isolate;
  color: #111111
}
```



p.p ()

```
CSS:
p { width: xpx;
  border: xpx solid #111111;
  space between - padding: xpx;
  border & paragraph margin: xpx; }
```

space between - padding: xpx;
 border & paragraph margin: xpx; }

↳ using div instead of p use this style for that entire section

```
div { } & p.div ( p.p ( ) )
p.div ( stuff )
```

```
li { }
```

```
#div1 > p { color: black; }
```

= text color of paragraph inside div 1 is black (child properties)

TOPIC

DATE

Specificity - unique ID

in HTML everything w/ ID must have unique ID

ex:

p.p (id="p1") ("bla bla")
p.p (id="p2") ("bla bla")

in CSS
p { affects all paragraphs }
#p1 { only that element }
#p2 { }

#id_name > input { color: orange; }
the input of a text box holding that ID is orange

Using class

ID only for 1 element max

class is if 2 elements, same ID

p { }
p.class-name { }

CSS grid: designs page structure better than table row + data

class-name {

display: grid;

grid-template-columns: 1fr auto auto;

border: 2px solid; }

ex: p.div (class="class-name") (pdiv("1", _____, _____))

how many columns

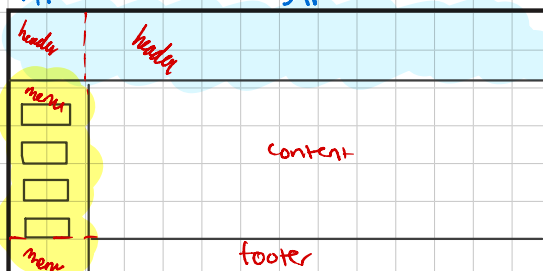
col1 col2 col3

fr = fraction

1fr 1fr
1: 1

will auto adjust when resized

What if I want this grid?
1fr 3fr



```

.container {
  display: grid;
  grid-template-columns: 1fr 3fr;
  grid-template-rows: 1fr 3fr;
  grid-template-columns: 1fr 3fr;
  background-color: #f0f0f0;
  padding: 10px;
  border: 2px solid #ccc;
}
.container div {
  background-color: #fff;
  padding: 10px;
}
.container div.header {
  grid-area: header;
  text-align: center;
}
.container div.menu {
  grid-area: menu;
}
.container div.content {
  grid-area: content;
}
.container div.footer {
  grid-area: footer;
  text-align: center;
}

```

```

def get_body_content():
    return <div class="container">
      <div class="header">
        <p>header stuff goes here</p>
      </div>
      <div class="menu">
        <p>navigation</p>
        <p>get here</p>
      </div>
      <div class="main">
        <p>main content goes here</p>
      </div>
    </div>

```

all is in .container class

body { color: }
input { color: }
.menu > input { } - all input in the main class has this
#ID { grid-column-start: 1;
grid-column-end: -2; } - 2nd column from the right

- * use proper inputs
 - date for date
 - restrict user input to make valid inputs
 - radio buttons for MCQ
 - dropdown for countries
 - + help error reduction
 - + let people use ur product

a. Learnability

b. Efficiency (less clicks)

c. Memorability (remember how to use?)

d. Errors (ez to recover, r u swe u wanna delete?)

e. Satisfaction (aesthetics)

Wordle?

```
session { "guesses" : [ 0      After
                1
                2
                3
                4          ],
          "answer" : "frame" }
          ↳ so the answer persists
          ]
```