THE UNIVERSITY OF MELBOURNE

COMP10001 - Sem 2 2024 - Week 4

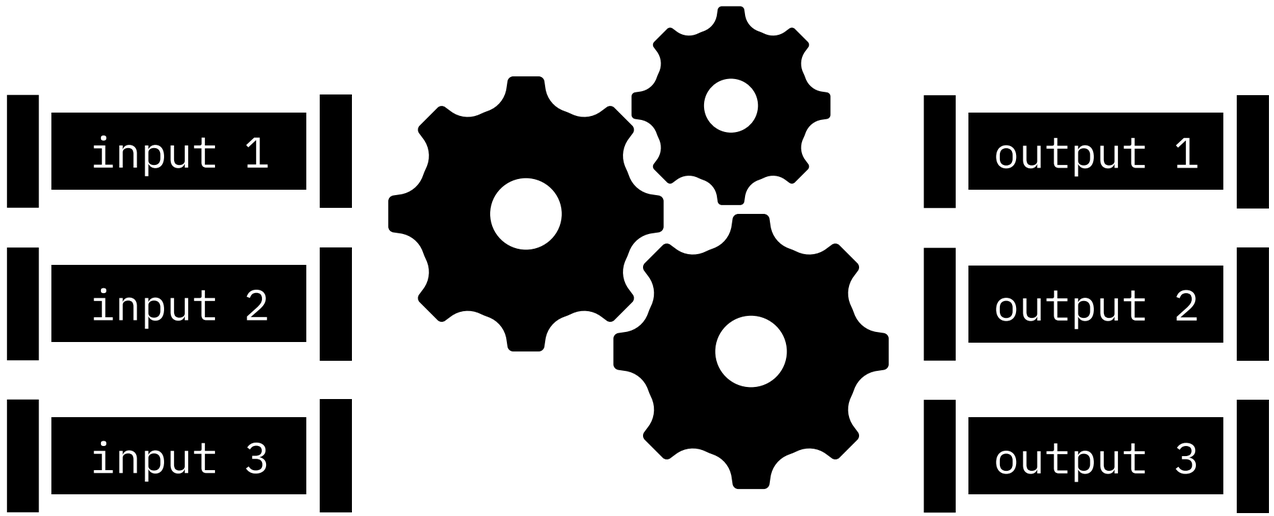
Foundations of Computing

Daksh Agrawal

Slides

With Annotations

Functions



```
def function_name(input1, input2):
    processing = input1 + input2
    return processing, output2

function_name(2,3)
```

```python
def ave(a, b):
    result = (a + b) / 2

def ave_p(a, b):
    result = (a + b) / 2
    print("p", result)

def ave_r(a, b):
    result = (a + b) / 2
    return result

def ave_pr(a, b):
    result = (a + b) / 2
    print("pr", result)
    return result

res = ave(1, 2)
res_p = ave_p(1, 2)
res_r = ave_r(1, 2)
res_pr = ave_pr(1, 2)
```

res_pr = 1.5
res_r = 1.5
res = None
res_p = None

User
p 1.5
pr 1.5

```
# Fix the Code
def calc(n1, n2):
    answer = n1 + (n1 * n2)
    print(answer)

num = int(input("Enter the second number: "))
result = calc(2, num)
print("The result is:", result)
```

2 5 12 return answer 5 None

User
12
the result is None

Evaluate by Hand, given s = "Computing is FUN!"

`s.isupper()`	`s.upper()`	`s.endswith("FUN!")`	`s.count('i')`
False	"COMPUTING IS FUN!" string	True	2 int
`s.strip('!')`	`s.replace('i', '!')`	`s.split()`	`s.isdigit()`
"Computing is FUN" lstrip rstrip	"Comput!ng !s FUN!"	["Computing", "is", "FUN!"] s.split('i')	False

fruits[2] = "mangoes"

info[2] = "hello" X Error

Data Type

A sequence of values

list ["apples", "oranges", 56]

"mangoes"

> mutability

tuple (True, 45, "Melbourne")

[(23,"Alice"), (45, "Bob"), (98, "Charlie")]

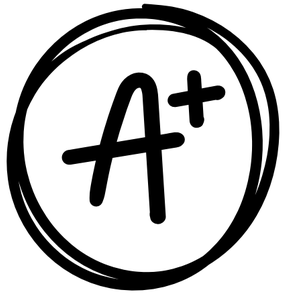
0 1 2

Evaluate, given lst = [2, ("green", "eggs", "ham"), False]

`lst[2]` False bool	`lst[1][-2]` "eggs" string	`lst[1][-2][:3]` "egg" string

`lst.append(5)`
`print(lst)`
add to end
[2, ("green", "eggs", "ham"), False, 5]
	`lst.pop(2)`
	`print(lst)`
	remove
	[2, ("green", "eggs", "ham")]
		`lst.reverse()`
		`print(lst)`
		[False, ("green", "eggs", "ham"), 2]

Past Exam Questions



Evaluate by Hand

```
22 % 4 * 4 + 1.5 - 22 // 3
```

2 * 4 + 1.5 - 22//3

8 + 1.5 - 22//3

8 + 1.5 - 7

9.5 - 7

2.5

```
"hello"[:3] + "p"
```

"hel" + "p"

"help"

tuple([1,2,3])

(1,2,3)

```
(1,2) + (3) + (4,5)
```

~~(1,2,3,4,5)~~ ✗ Error

(1,2) + 3 + (4,5)

(1,2) + (3,) + (4,5)

(1,2,3,4,5)

```
tuple('abc') + (4,)
```

('abc', 4) ✗

('a','b','c') + (4,)

('a','b','c',4)

One Liners!

Suppose that `str1` and `str2` are two strings, and that `k` is a positive integer. Give a single Python assignment statement that assigns `True` to `match` if `str1` and `str2` have the same first `k` characters, and assigns `False` to `match` if not.

```
match = str1[:k] == str2[:k]
```

One Liners!

Suppose that `lst` is a non-empty list of numbers. Give a single Python assignment statement that assigns the difference between the largest and smallest numbers in `lst` to the variable `diff`.

```
diff = max(lst) - min(lst)
diff = lst.sorted()[-1] - lst.sorted()[0]
```

One Liners!

Suppose that text is a Python string. Give a single Python assignment statement that assigns the number of words in text to wrds, where a “word” is any non-blank sequence of characters. (Hint: A method covered in previous exercises may be useful).

```
wrds = len(text.split())
```

```
"Happiness.is.key.to.success".split()
["Happiness", "is", "key", "to", "success"]
5
```

Paper Programming

Write a function which converts a temperature between degrees Celsius and Fahrenheit. It should take a float, the temperature to convert, and a string, either `'c'` or `'f'` indicating a conversion from degrees Celsius and Fahrenheit respectively. The formulae for conversion is: `Fahrenheit = Celsius * 1.8 + 32`

int

string "c" "f"

```
def conversion(temp,unit):
    if unit == "c":
        return temp * 1.8 + 32
    else:
        return (temp - 32) / 1.8
```

Write a function which takes a sentence as a single argument (in the form of a string), and evaluates whether it is valid based on whether the first letter is capitalised and the last character is a full stop. Return a Boolean value `True` or `False`.

```
def is_valid(sentence):
    if sentence[:1].isupper() and sentence[-1] == ".":
        return True
    else:
        return False
```

```
def is_valid(sentence):
    return sentence[0].isupper() and sentence.endswith(".")
```

Simple loop. Write two programs to print integer numbers from -5 to 5 (inclusive). The first program should use a for loop and the second program should use a while loop.

Write a function which takes a string containing an FM radio frequency and returns whether it is a valid frequency. A valid frequency is within the range 88.0-108.0 inclusive with 0.1 increments, meaning it must have only one decimal place.
`valid_fm('103.14')` should return `False`.

```
def valid_fm(freq):
    freq_float = float(freq)
    if 88.0 <= freq_float <= 108.0 and freq[-2]==".":
        return True
    else
        return False
```

103.14

["103", "14"]

Shake it off!

Create a function that removes all punctuation from a string, "shaking off" the unnecessary characters. For example, "Hello, world!" becomes "Hello world".