IT@JCU

CP1401 Module 5 Coding Checkpoint 1

JAMES COOK UNIVERSITY AUSTRALIA

Learning outcomes - you will be able to:

- Evaluate how you're going with the subject material we've taught so far
- Use appropriate comments in Python code
- Use basic string formatting to format values

IT@JCU

Comments are useful to make our code more readable

Don't use comments to make your code **less** readable!

JAMES COOK UNIVERSITY AUSTRALIA

What does this code do?

```
result = constant * user_float ** 2
```

- What about?

```
# Calculate area of circle (user_float is radius)
result = constant * user_float ** 2
```

- What about?

```
circle_area = math.pi * radius ** 2
```

Python has two kinds of comments

- Block / inline comments
  - Start with # ("hash", not "hash tag") then a space
  - One-line, short

```
# This is a comment
```

- Docstrings
  - Start and end with triple quotes """
  - Only used for modules, functions, classes, not in between normal code

```
"""
CP1401 - Coding Checkpoint 1
https://github.com/CP1401/Practicals
"""
```

Use block/inline comments appropriately

- Put # inline comments above the code they refer to.
- End-of-line comments are only acceptable if they are short (avoid horizontal scrolling)

```
# format date for pvoutput.org
date = date.replace('.', '-')
time = time[:-3]  # strip seconds
```

Only write helpful comments

- Comments should be used to add clarity in situations where it is not clear what is going on in the code
- Our goal is to write code that is totally clear
- Some programmers call this "self-documenting code"
- Brief comments should be used to explain complex pieces of code
- Write comments in the *imperative voice*
  E.g., `# Calculate chance of rain`
  not `# Calculates chance of rain`

Don't write "noise" comments

- Using too many comments to tell us what we already know just gets in the way of understanding the code
- Too many comments interrupts the flow when reading the code
  - Like normal movie sound AND a Director's commentary at the same time
  - Like too many footnotes
  - Like too many examples of things that noise comments are like
- If you can "refactor" your code (especially by using better names) to make it more clear, comments may not be necessary

Don't write bad/unnecessary comments

- "This function will...", "The following code…" or similar is always redundant.
- Who needs the following comments?

```
# initialise variables...
# import statements...
```

- Don't create *maintenance burdens* with your comments.

```
# Multiply value by 0.02
result = value * 0.02
```

Comments help you understand your own code
(think about having to read it again in a few years)

```
# format date for pvoutput.org
date = date.replace('.', '-')
time = time[:-3]  # strip seconds


# data looks like {'date': '01/03/2018 09:00:00',
'percent': '63%', 'volume': '146124 ML'}
print(data["percent"][:-1])
```

These come from:

https://github.com/lindsaymarkward/HelpLindsay/blob/master/solar_pvoutput.py

https://github.com/lindsaymarkward/HelpLindsay/blob/master/get_dam_level.py

IT@JCU

Python has useful ways of formatting string outputs

Just the basics for now

JAMES COOK UNIVERSITY AUSTRALIA

There are different ways of formatting strings

- Instead of:

```
print("You want ", product, ". It costs $", price, sep="")
```

- You can use the format method:

```
print("You want {}. It costs ${}".format(product, price))
```

- Since Python 3.6, you can use f-strings:

```
print(f"You want {product}. It costs ${price}")
```

Use format specifiers to customise output

- 2 Decimal places:

```
print(f"You want {product}. It costs ${price:.2f}")
```

The format is:

```
{value:specifier}
```

Here, `.2f` means: `.2` (two decimal places) `f` (float)

What is the output of?

```
total = 0
for i in range(1, 21, 2):
    total += i
    print(f"{i} {total}")
```

```
1 1
3 4
5 9
7 16
9 25
11 36
13 49
15 64
17 81
19 100
```

Line up outputs by specifying a width

```
total = 0
for i in range(1, 21, 2):
    total += i
    print(f"{i:2} {total:3}")
```

```
 1   1
 3   4
 5   9
 7  16
 9  25
11  36
13  49
15  64
17  81
19 100
```

Do this now

- Complete the questions in the checkpoint 1 practical

- We will work through the solutions in the lecture videos, highlighting important things and revising what we've learned
- The more you do *before* you see the solutions, the more helpful this exercise is.

Now do these next steps

- Revise ANYTHING that you had trouble with during this checkpoint
- Re-read (re-watch) your lectures/notes
- Re-do your practicals including the practice and extension sections
- Keep practising exercises like we have here, until it "sticks"
- Work on your assignment, following what you've learned