COMP120: Creative Computing: Tinkering

3: Maintainability

Learning Outcomes

- **Identify** code which threatens maintainability
- **Explain** the principles of good code
- **Analyse** code to identify potential improvements

Maintainability

Remember:

- Maintainability is one of the most heavily weighted criteria on your programming assignments
- Remeber to read *Clean Code*
- Clean Code describes many principles behind writing maintainable code
- Also look through PEP-8
- These are the Python standards against which your work will be reviewed

So, what is maintainability?

Maintainability

Code maintainability is "**itself a measure of the ease to modify code; higher maintainability means less time to make a change**"

Maintainability

We ensure maintainability by ensuring:

- Code is readable:
  - Code can be understood
  - Code is reasonably structured in an easy-to-read way
  - Code follows a house style (e.g., PEP-8)

Let's explore some of the `PEP-8` conventions:

`www.python.org/dev/peps/pep-0008/`

Maintainability

We ensure maintainability by ensuring:

- Code is not redundant
    - Code is not repeated
    - Instead, code is refactored into functions for re-use and is embued with versatility, thereby handling a range of different input

Maintainability

We ensure maintainability by ensuring:

- Appropriate architectures are embedded into code design:
  - Coupling is minimised
  - Cohesion of responsibility is maximised
  - Entanglement is avoided (i.e., spaghetti code)

Maintainability

We ensure maintainability by ensuring:

- Code is sufficiently documented:
  - Comments provide useful clarifications
  - "Code tells you how, comments tell you why"
  - Doc-strings describe the capabilities of the codebase
  - Appropriate links to living documentation, like wikis

Let's explore some of the `PEP-257` conventions:

`www.python.org/dev/peps/pep-0257/`

Spaghetti Code

```
import time

letters = "we gonna divide some stuff"
n1="type first number:  "
n2="type second number to divide by:  "

print(letters)

a=float(input(n1))
b=float(input(n2))

# ##### DONT TOUCH ANYTHING BELOW LINE #####
# #####  IT WORKS AND I DONT KNOW WHY   #####
add_used = 0
```

Spaghetti Code

```
# define add
def add(a, b):
    global add_used
    add_used += 1
    return a + b

# dont know why this works but it does.
def divide(a, b):
    quotient = 0
    c = 0
    d = 0
    while add(d, b) <= a:
        c = add(c, 1)
        d = add(d, b)
    return c

print("the answer is: ",divide(a, b))

time.sleep(3)
```

FALMOUTH UNIVERSITY

Better Code

```
import time

print('divide two numbers')

# get the user to enter in some integers
x=int(input('enter first number: '))
y=int(input('enter number to divide by: '))

print('the answer is: ',int(x/y)),

time.sleep(3) #delay of a few seconds before closing
```

Clean Code

A key issue that first-year students tend to encounter, are identifiers for their variables. Please use sensible names! Uncle Bob (author of Clean Code) suggests:

- Meaningful names, which:
  - Are explicit
  - Reveal intentions
  - Avoid disinformation
  - Make meaningful distinctions
  - Are searchable
  - Avoid arbitrary encodings
  - Avoid mental mapping
  - Aren't "cute" or puns
  - Use domain and solution terms

FALMOUTH UNIVERSITY

Clean Code

```
count_of_college_graduates = 2500
```

is better than:

```
gn = 2500
```

More on Readability

Review more readability issues here:

`treyhunner.com/readability-counts/#/`

PASS Challenge

Review the following python setup code:

```python
import random
randomNumber = random.randrange(0,100)
print("Random number has been generated")
```

PASS Challenge

Review the following pythoin game code:

```
guessed = False
while guessed==False:
    userInput = int(input("Your guess pleas:  "))
    if userInput==randomNumber:
        guessed = True
        print("Well done!")
    elif userInput>100:
        print("Our guess range is between 0 and 100,  ↩
            please try a bit lower")
    elif userInput<0:
        print("Our guess range is between 0 and 100,  ↩
            please try a bit higher")
    elif userInput>randomNumber:
        print("Try one more time, a bit lower")
    elif userInput < randomNumber:
        print("Try one more time, a bit higher")
print("You win!")
```

PASS Challenge

- In pairs
- **Implement** the code excerpt
- **Refactor** the code to improve readability
- **Improve** overall maintainability of the code, breaking it down into functions
- **Note** the principles which make the revised version better

You can learn more about PyGame `random` at:

`docs.python.org/3.6/library/random.html`

(40 minutes)