UNIVERSITI SAINS MALAYSIA

SCHOOL OF COMPUTER SCIENCES
Semester 2, 2022/2023

CPC152: Foundations and Programming for Data Analytics
Tutorial 2

1. What type of value is 3.4? How can you find out?

   Solution: floating-point number
   print(type(3.4))

2. What type of value is 3.25 + 4?

   Solution:

   floating-point number

3. What type of value (integer, floating point number, or character string) would you use to represent each of the following? Try to come up with more than one good answer for each problem. For example, in # 1, when would counting days with a floating point variable make more sense than using an integer?

   1. Number of days since the start of the year.
   2. Time elapsed from the start of the year until now in days.
   3. Serial number of a piece of lab equipment.
   4. A lab specimen's age
   5. Current population of a city.
   6. Average population of a city over time.

   Solution:

   1. integer

   2. floating point number

   3. character , string , integer

   4. floating point number, string

   5. floating point number, integer

   6. floating point number

4. In Python 3, the `//` operator performs integer (whole-number) floor division, the `/` operator performs floating-point division, and the `%` (or *modulo*) operator calculates and returns the remainder from integer division:

Python
```
print('5 // 3:', 5 // 3)
print('5 / 3:', 5 / 3)
print('5 % 3:', 5 % 3)
```

Output
```
5 // 3: 1
5 / 3: 1.6666666666666667
5 % 3: 2
```

If `num_subjects` is the number of subjects taking part in a study, and `num_per_survey` is the number that can take part in a single survey, write an expression that calculates the number of surveys needed to reach everyone once.

Solution:

Python
```
num_subjects = 600
num_per_survey = 42
num_surveys =
```

Output

```
num_surveys = num_subjects // num_per_survey
if num_subjects % num_per_survey != 0:     # This checks if there is a remainder
    num_surveys += 1                       # This adds 1 if there is a remainder
```

5. Which of the following will return the floating point number `2.0`? Note: there may be more than one right answer.

Python
```
first = 1.0
second = "1"
third = "1.1"
```

1. `first + float(second)` T
2. `float(second) + float(third)` F
3. `first + int(third)` F : invalid literal for int() with base 10: '1.1'
4. `first + int(float(third))` T
5. `int(first) + int(float(third))` F
6. `2.0 * second` F : can't multiply sequence by non-int of type 'float'

Solution:

6. Python provides complex numbers, which are written as `1.0+2.0j`. If `val` is a complex number, its real and imaginary parts can be accessed using *dot notation* as `val.real` and `val.imag`.

Python

```
a_complex_number = 6 + 2j
print(a_complex_number.real)
print(a_complex_number.imag)
```

Output

```
6.0
2.0
```

1. What do you expect `1 + 2j + 3` to produce?
2. What do you expect `4j` to be? What about `4 j` or `4 + j`?

Solution:

1. 4.0
2.0

2. 4j will produce
0.0
4.0

both 4 j and 4 + j are invalid