COMP120: Creative Computing: Tinkering

1: Python, Pairs, & PyGame

Learning Outcomes

- Explain the role and basic functions of the IDE
- Produce some basic Python programs
- **Apply** pair programming practices to solve simple problems

FALMOUTH UNIVERSITY

Professional Development

FALMOUTH UNIVERSITY

Continuing Professional Development

- Games industry is fast-moving
- Learning does not end at school and university
- A goal of this course is to facilitate your development as self-regulated learners
- Gradually, more independence across each year of study
- This is a science degree, which means you will become a producer of knowledge, not just a consumer of knowledge!

Continuing Professional Development

- It isn’t easy!
- Many of you will encounter programming anxiety
- Some will experience a sense of fear or a sense of hopelessness — it is more common than you think
- Some will need more support than others — this isn’t a bad thing
- Everyone who puts in the time and effort will eventually achieve mastery

FALMOUTH UNIVERSITY

The PyCharm IDE

Integrated Development Environment (IDE)

Integrated Development Environment (IDE)

- You *could* just write code in Notepad, but...

Integrated Development Environment (IDE)

- You *could* just write code in Notepad, but...
- An **Integrated Development Environment (IDE)** is an application providing several useful features for programmers, including:

Integrated Development Environment (IDE)

- You *could* just write code in Notepad, but...
- An **Integrated Development Environment (IDE)** is an application providing several useful features for programmers, including:
  - A "run" button

Integrated Development Environment (IDE)

- You *could* just write code in Notepad, but...
- An **Integrated Development Environment (IDE)** is an application providing several useful features for programmers, including:
  - A "run" button
  - Management of multi-file projects

Integrated Development Environment (IDE)

- You *could* just write code in Notepad, but...
- An **Integrated Development Environment (IDE)** is an application providing several useful features for programmers, including:
  - A “run” button
  - Management of multi-file projects
  - Syntax highlighting

Integrated Development Environment (IDE)

- You *could* just write code in Notepad, but...
- An **Integrated Development Environment (IDE)** is an application providing several useful features for programmers, including:
  - A “run” button
  - Management of multi-file projects
  - Syntax highlighting
  - Autocompletion

Integrated Development Environment (IDE)

- You *could* just write code in Notepad, but...
- An **Integrated Development Environment (IDE)** is an application providing several useful features for programmers, including:
  - A "run" button
  - Management of multi-file projects
  - Syntax highlighting
  - Autocompletion
  - Navigation

Integrated Development Environment (IDE)

- You *could* just write code in Notepad, but...
- An **Integrated Development Environment (IDE)** is an application providing several useful features for programmers, including:
  - A "run" button
  - Management of multi-file projects
  - Syntax highlighting
  - Autocompletion
  - Navigation
  - Language and API documentation

Integrated Development Environment (IDE)

- You *could* just write code in Notepad, but...
- An **Integrated Development Environment (IDE)** is an application providing several useful features for programmers, including:
  - A “run” button
  - Management of multi-file projects
  - Syntax highlighting
  - Autocompletion
  - Navigation
  - Language and API documentation
  - Debugging

Integrated Development Environment (IDE)

- You *could* just write code in Notepad, but...
- An **Integrated Development Environment (IDE)** is an application providing several useful features for programmers, including:
  - A “run” button
  - Management of multi-file projects
  - Syntax highlighting
  - Autocompletion
  - Navigation
  - Language and API documentation
  - Debugging
  - Profiling

Integrated Development Environment (IDE)

- You *could* just write code in Notepad, but...
- An **Integrated Development Environment (IDE)** is an application providing several useful features for programmers, including:
  - A "run" button
  - Management of multi-file projects
  - Syntax highlighting
  - Autocompletion
  - Navigation
  - Language and API documentation
  - Debugging
  - Profiling
  - Version control

FALMOUTH UNIVERSITY

Setting up your own PC

Setting up your own PC

- Python 2.7

Setting up your own PC

- Python 2.7
  - `https://www.python.org/`

Setting up your own PC

- Python 2.7
    - `https://www.python.org/`
    - Python 2.7 is included with Mac OSX and most Linux distributions, but needs to be installed separately on Windows

Setting up your own PC

- Python 2.7
  - `https://www.python.org/`
  - Python 2.7 is included with Mac OSX and most Linux distributions, but needs to be installed separately on Windows
  - Python 2.x and Python 3.x are (slightly) different programming languages; we are using 2.x (for now)

Setting up your own PC

- Python 2.7
  - `https://www.python.org/`
  - Python 2.7 is included with Mac OSX and most Linux distributions, but needs to be installed separately on Windows
  - Python 2.x and Python 3.x are (slightly) different programming languages; we are using 2.x (for now)
- PyCharm

Setting up your own PC

- Python 2.7
  - `https://www.python.org/`
  - Python 2.7 is included with Mac OSX and most Linux distributions, but needs to be installed separately on Windows
  - Python 2.x and Python 3.x are (slightly) different programming languages; we are using 2.x (for now)
- PyCharm
  - `https://www.jetbrains.com/student/`

Setting up your own PC

- Python 2.7
  - `https://www.python.org/`
  - Python 2.7 is included with Mac OSX and most Linux distributions, but needs to be installed separately on Windows
  - Python 2.x and Python 3.x are (slightly) different programming languages; we are using 2.x (for now)
- PyCharm
  - `https://www.jetbrains.com/student/`
  - Register with your `falmouth.ac.uk` email address to obtain PyCharm Professional Edition for free

Setting up your own PC

- Python 2.7
  - `https://www.python.org/`
  - Python 2.7 is included with Mac OSX and most Linux distributions, but needs to be installed separately on Windows
  - Python 2.x and Python 3.x are (slightly) different programming languages; we are using 2.x (for now)
- PyCharm
  - `https://www.jetbrains.com/student/`
  - Register with your `falmouth.ac.uk` email address to obtain PyCharm Professional Edition for free
  - Runs on Windows, Mac and Linux

Setting up your own PC

- Python 2.7
  - `https://www.python.org/`
  - Python 2.7 is included with Mac OSX and most Linux distributions, but needs to be installed separately on Windows
  - Python 2.x and Python 3.x are (slightly) different programming languages; we are using 2.x (for now)
- PyCharm
  - `https://www.jetbrains.com/student/`
  - Register with your `falmouth.ac.uk` email address to obtain PyCharm Professional Edition for free
  - Runs on Windows, Mac and Linux
  - Other Python IDEs are available

PyCharm in the Lab

FALMOUTH UNIVERSITY

PyCharm in the Lab

- You have to license your account to use PyCharm

PyCharm in the Lab

- You have to license your account to use PyCharm
- Run PyCharm and select **License server**

PyCharm in the Lab

- You have to license your account to use PyCharm
- Run PyCharm and select **License server**
- In the **License server address** enter the following **http://trlicefal.fal.ac.uk**

PyCharm in the Lab

- You have to license your account to use PyCharm
- Run PyCharm and select **License server**
- In the **License server address** enter the following **http://trlicefal.fal.ac.uk**
- This will be added to your user profile and you will not need to do this again

Getting started with PyCharm

Getting started with PyCharm

- Create a new project (from the start-up wizard or from the File menu)

Getting started with PyCharm

- Create a new project (from the start-up wizard or from the File menu)
- We want a “Pure Python” project

Getting started with PyCharm

- Create a new project (from the start-up wizard or from the File menu)
- We want a “Pure Python” project
- Right-click the project in the panel on the left, and choose “New $\rightarrow$ Python File”

Getting started with PyCharm

- Create a new project (from the start-up wizard or from the File menu)
- We want a “Pure Python” project
- Right-click the project in the panel on the left, and choose “New $\rightarrow$ Python File”
- Write some code!

Getting started with PyCharm

- Create a new project (from the start-up wizard or from the File menu)
- We want a “Pure Python” project
- Right-click the project in the panel on the left, and choose “New $\to$ Python File”
- Write some code!
- First run: click “Run $\to$ Run...” and choose the Python file

Getting started with PyCharm

- Create a new project (from the start-up wizard or from the File menu)
- We want a "Pure Python" project
- Right-click the project in the panel on the left, and choose "New → Python File"
- Write some code!
- First run: click "Run → Run..." and choose the Python file
- Subsequent runs: click the ▶ button

FALMOUTH UNIVERSITY

Basic Python programs

Your first Python program

```
print "Hello, world!"
```

Your second Python program

```
print "This is a very long line of code which had to  ↩
    be split to fit on the slide, but you should type  ↩
    it as a single line."
print "This is the second line of code."
```

Assigning to variables

```
a = 10
print a
```

Assigning to variables

```
a = 10
print a
```

Variable	Value
a	

Remember!

Remember!

- A program is a **sequence of instructions**

Remember!

- A program is a **sequence of instructions**
- The Python interpreter executes the **first line** of your program, then the **second line**, and so on

Remember!

- A program is a **sequence of instructions**
- The Python interpreter executes the **first line** of your program, then the **second line**, and so on
- When it reaches the end of the file, it **stops**

Reassigning variables (1)

```
a = 10
b = 20
b = a
print a
print b
```

FALMOUTH UNIVERSITY

Reassigning variables (1)

```
a = 10
b = 20
b = a
print a
print b
```

Variable	Value
a	
b	

Reassigning variables (2)

```
a = 10
b = 20
a = b
print a
print b
```

Reassigning variables (2)

```
a = 10
b = 20
a = b
print a
print b
```

Variable	Value
a	
b	

Reassigning variables (3)

```
big = 10
small = 20
big = small
print big
print small
```

Reassigning variables (3)

```
big = 10
small = 20
big = small
print big
print small
```

Variable	Value
big	
small	

Reassigning variables (4)

```
a = 10
b = 20
a = b
b = a
print a
print b
```

Reassigning variables (4)

```
a = 10
b = 20
a = b
b = a
print a
print b
```

Variable	Value
a	
b	

FALMOUTH UNIVERSITY

Reassigning variables (5)

```
a = 10
b = 20
c = 30

a = b
b = c

print a
print b
print c
```

Reassigning variables (5)

```
a = 10
b = 20
c = 30

a = b
b = c

print a
print b
print c
```

Variable	Value
a	
b	
c	

Reading input

Reading input

```
print "Enter your name:"
name = raw_input()

print "Enter your age:"
age = int(raw_input())

print "Hello", name
print "On your next birthday, you will be", age + 1, " ↩
    years old"
```

Reading input

```
print "Enter your name:"
name = raw_input()

print "Enter your age:"
age = int(raw_input())

print "Hello", name
print "On your next birthday, you will be", age + 1, " ↩
    years old"
```

- `raw_input()` reads a **string** as text from the command line

Reading input

```
print "Enter your name:"
name = raw_input()

print "Enter your age:"
age = int(raw_input())

print "Hello", name
print "On your next birthday, you will be", age + 1, " ↩
    years old"
```

- `raw_input()` reads a **string** as text from the command line
- `int(...)` converts a **string** into an **integer** (a number)

FALMOUTH UNIVERSITY

Conditionals (1)

```
a = int(raw_input())
b = 30

if a < 15:
    b = a

print a
print b
```

Conditionals (1)

```
a = int(raw_input())
b = 30

if a < 15:
    b = a

print a
print b
```

Variable	Value
a	
b	

Indentation

Indentation

- Unlike many other programming languages, **indentation has meaning** in Python!

Indentation

- Unlike many other programming languages, **indentation has meaning** in Python!
- Python uses indentation to denote the **block of code** inside a conditional, loop, function etc.

FALMOUTH UNIVERSITY

Indentation

- Unlike many other programming languages, **indentation has meaning** in Python!
- Python uses indentation to denote the **block of code** inside a conditional, loop, function etc.
- PEP-8 recommends **4 spaces** for indentation

Indentation

- Unlike many other programming languages, **indentation has meaning** in Python!
- Python uses indentation to denote the **block of code** inside a conditional, loop, function etc.
- PEP-8 recommends **4 spaces** for indentation
    - Some programmers use a tab character

Indentation

- Unlike many other programming languages, **indentation has meaning** in Python!
- Python uses indentation to denote the **block of code** inside a conditional, loop, function etc.
- PEP-8 recommends **4 spaces** for indentation
    - Some programmers use a tab character
    - **Never** mix tabs and spaces in the same file!

Indentation

- Unlike many other programming languages, **indentation has meaning** in Python!
- Python uses indentation to denote the **block of code** inside a conditional, loop, function etc.
- PEP-8 recommends **4 spaces** for indentation
  - Some programmers use a tab character
  - **Never** mix tabs and spaces in the same file!
  - PyCharm inserts 4 spaces by default when you press the tab key; other IDEs and text editors can be configured to do this

FALMOUTH UNIVERSITY

Conditionals (2)

```
a = int(raw_input())
b = 0

if a < 20:
    b = a + 1
elif a == 20:
    b = a * 2
else:
    a = 20
    b = 20

print a
print b
```

Conditionals (2)

```
a = int(raw_input())
b = 0

if a < 20:
    b = a + 1
elif a == 20:
    b = a * 2
else:
    a = 20
    b = 20

print a
print b
```

Variable	Value
a	
b	

Conditionals

Conditionals

An `if` statement can have:

Conditionals

An `if` statement can have:

- **Zero or more** `elif` clauses

Conditionals

An `if` statement can have:

- **Zero or more** `elif` clauses
- **An optional** `else` clause

Conditionals

An `if` statement can have:

- **Zero or more** `elif` clauses
- **An optional** `else` clause

In that order!

Mathematical operators

Mathematical operators

- + add

Mathematical operators

- + add
- − subtract

Mathematical operators

- + add
- − subtract
- * multiply

Mathematical operators

- + add
- − subtract
- * multiply
- / divide

Mathematical operators

- + add
- − subtract
- * multiply
- / divide
- ** power

Mathematical operators

- + add
- - subtract
- * multiply
- / divide
- ** power

Order of operations: **BIDMAS**

FALMOUTH UNIVERSITY

Mathematical operators

- + add
- − subtract
- * multiply
- / divide
- ** power

Order of operations: **BIDMAS**

- Brackets first

Mathematical operators

- + add
- − subtract
- * multiply
- / divide
- ** power

Order of operations: **BIDMAS**

- Brackets first
- Then indices (powers)

Mathematical operators

- + add
- − subtract
- * multiply
- / divide
- ** power

Order of operations: **BIDMAS**

- Brackets first
- Then indices (powers)
- Then division and multiplication (left to right)

Mathematical operators

- + add
- − subtract
- * multiply
- / divide
- ** power

Order of operations: **BIDMAS**

- Brackets first
- Then indices (powers)
- Then division and multiplication (left to right)
- Then addition and subtraction (left to right)

Comparison operators

Comparison operators

- $<$ less than

Comparison operators

- < less than
- <= less than or equal to

Comparison operators

- < less than
- <= less than or equal to
- > greater than

Comparison operators

- < less than
- <= less than or equal to
- > greater than
- >= greater than or equal to

Comparison operators

- < less than
- <= less than or equal to
- > greater than
- >= greater than or equal to
- == equal to

Comparison operators

- `<` less than
- `<=` less than or equal to
- `>` greater than
- `>=` greater than or equal to
- `==` equal to
- `!=` not equal to

Comparison operators

- `<` less than
- `<=` less than or equal to
- `>` greater than
- `>=` greater than or equal to
- `==` equal to
- `!=` not equal to

Note the difference between `=` and `==`

Comparison operators

- `<` less than
- `<=` less than or equal to
- `>` greater than
- `>=` greater than or equal to
- `==` equal to
- `!=` not equal to

Note the difference between `=` and `==`

- `a = b` means "make `a` be equal to `b`"

Comparison operators

- `<` less than
- `<=` less than or equal to
- `>` greater than
- `>=` greater than or equal to
- `==` equal to
- `!=` not equal to

Note the difference between `=` and `==`

- `a = b` means "make `a` be equal to `b`"
- `a == b` means "is `a` equal to `b`?"

FALMOUTH UNIVERSITY

For loops and ranges

```
for i in xrange(5):
    print i
```

FALMOUTH UNIVERSITY

For loops and ranges

```
for i in xrange(5):
    print i
```

- `xrange(n)` is the **sequence** $0, 1, 2, \ldots, n-1$

For loops and ranges

```
for i in xrange(5):
    print i
```

- `xrange(n)` is the **sequence** $0, 1, 2, \ldots, n - 1$
- So `xrange(5)` is the **sequence** $0, 1, 2, 3, 4$

For loops and ranges

```
for i in xrange(5):
    print i
```

- `xrange(n)` is the **sequence** $0, 1, 2, \ldots, n-1$
- So `xrange(5)` is the **sequence** $0, 1, 2, 3, 4$
- Note: `xrange(n)` **does not include** n

For loops and ranges

```
for i in xrange(5):
    print i
```

- `xrange(n)` is the **sequence** $0, 1, 2, \ldots, n - 1$
- So `xrange(5)` is the **sequence** $0, 1, 2, 3, 4$
- Note: `xrange(n)` **does not include** n
- The `for` loop iterates through the items in a sequence **in order**

For loops and ranges

```
for i in xrange(5):
    print i
```

- `xrange(n)` is the **sequence** $0, 1, 2, \ldots, n - 1$
- So `xrange(5)` is the **sequence** $0, 1, 2, 3, 4$
- Note: `xrange(n)` **does not include** n
- The `for` loop iterates through the items in a sequence **in order**
- Can also use `range` instead of `xrange`, but `range` is less efficient
  - Homework (advanced): what is the difference between `range` and `xrange`?

FALMOUTH UNIVERSITY

For loops (1)

```
a = 0
b = 0

for i in xrange(5):
    a = i
    b = b + i

print a
print b
```

For loops (1)

```
a = 0
b = 0

for i in xrange(5):
    a = i
    b = b + i

print a
print b
```

Variable	Value
a	
b	
i	

For loops (2)

```
a = 0
b = 0

for i in xrange(10):
    if i < 3 or i > 7:
        a += i
    else:
        b += i

print a
print b
```

For loops (2)

```
a = 0
b = 0

for i in xrange(10):
    if i < 3 or i > 7:
        a += i
    else:
        b += i

print a
print b
```

Variable	Value
a	
b	
i	

FALMOUTH UNIVERSITY

While loops

The `while` loop keeps executing while the condition is **true**

```
a = 1

while a < 100:
    a = a * 2

print a
```

While loops

The **`while`** loop keeps executing while the condition is **true**

```
a = 1

while a < 100:
    a = a * 2

print a
```

Variable	Value
a	

Looping forever

```
a = 1

while True:
    a = a * 2
    print a
```

Summary

Summary

We have seen some basic code constructions in Python

Summary

We have seen some basic code constructions in Python

- `print` and `raw_input` for command-line input and output

Summary

We have seen some basic code constructions in Python

- `print` and `raw_input` for command-line input and output
- Variable assignment using `=`

Summary

We have seen some basic code constructions in Python

- `print` and `raw_input` for command-line input and output
- Variable assignment using `=`
- `if` statements for choosing whether or not to execute a block of code

Summary

We have seen some basic code constructions in Python

- `print` and `raw_input` for command-line input and output
- Variable assignment using `=`
- `if` statements for choosing whether or not to execute a block of code
- `for` loops to execute a block of code a specified number of times

Summary

We have seen some basic code constructions in Python

- `print` and `raw_input` for command-line input and output
- Variable assignment using `=`
- `if` statements for choosing whether or not to execute a block of code
- `for` loops to execute a block of code a specified number of times
- `while` loops to execute a block of code until a condition is no longer true

Summary

We have seen some basic code constructions in Python

- `print` and `raw_input` for command-line input and output
- Variable assignment using `=`
- `if` statements for choosing whether or not to execute a block of code
- `for` loops to execute a block of code a specified number of times
- `while` loops to execute a block of code until a condition is no longer true

These are enough to write some simple programs, but you will see several more in coming weeks...

FALMOUTH
UNIVERSITY

Professional Practice

Pair Programming

Pair programming is an agile software development technique in which two programmers work together at one workstation.

One, the driver, writes code while the other, the observer or navigator, reviews each line of code as it is typed in.

The two programmers switch roles frequently.

Pair Programming

Watch the video at:

`https://www.youtube.com/watch?v=ET3Q6zNK3Io`

(5 minutes)

Pair Programming

Review the guidelines at:

`http://www.pairprogramming.co.uk/`

(10 minutes)

Pair Programming

Watch the video at:

`https://www.youtube.com/watch?v=ONnYCT_LJio`

(5 minutes)

FALMOUTH UNIVERSITY

Pair Programming Challenge

- In pairs
- **Implement** the code excerpt
- **Fix** the errors in the code excerpt
- **Modify** the code excerpt to incorporate functions and arguments
- **Post** your solution to the `#comp120` slack channel

You can learn more about functions and arguments at:

`https://docs.python.org/3/tutorial/controlflow.html#defining-functions`

(20 minutes)

Pair Programming Challenge

The function:

```
def madlib()
```

Should become:

```
def madlib(name, pet, verb, snack)
```

FALMOUTH UNIVERSITY

Pair Programming Challenge

```python
def madlib():
    name = 'Link'
    pet = 'Spyro'
    verb = 'ate'
    snack = 'doughnuts'
    line1 = 'once upon a time,' + name + ' walked'
    line2 = ' with ' + pet + ', a trained dragon.'
    line3 = 'Suddenly, ' + pet + ' announced,'
    line4 = 'I really want some ' + snack + '!'
    line5 = name + ' complained. Where am I going to  ↩
        get that?'
    line6 = 'Then ' + name + 'found a wizard's wand.'
    line 7 = 'With a wave of the wand, '
    line8 = pet + ' got ' + snack + '. '
    line9 = 'Perhaps surprisingly, ' + pet + ' ' +  ↩
        verb + ' ' + snack
    print line1 + line2 + line3 + line4
    print line5 + line6 + line7 + line8 + line9
```

Stretch Goal — PyGame

- In pairs
- **Incorporate** your code into the PyGame framework
- **Post** your solution to the `#comp120` slack channel

PyGame Live Coding

FALMOUTH
UNIVERSITY

Stretch Goal — PyGame

```python
import pygame

pygame.init()
screen = pygame.display.set_mode((640, 480))
font = pygame.font.SysFont(None, 14)
text = font.render(madlib(), True, (0, 128, 0))

done = False
while not done:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            done = True
        if event.type == pygame.KEYDOWN and event.key ↩
            == pygame.K_ESCAPE:
            done = True

    screen.fill((255, 255, 255))
    screen.blit(text,(0,0))
    pygame.display.flip()
```