Code Nation

Python Next Steps

Recap from Develop: Coding

{codenation}®

Learning Objectives

- Set up your programming environment
- Familiarise yourself with key coding concepts
- Revisit and refresh Python knowledge
- Complete activities to utilise your Python skills

{codenation}®

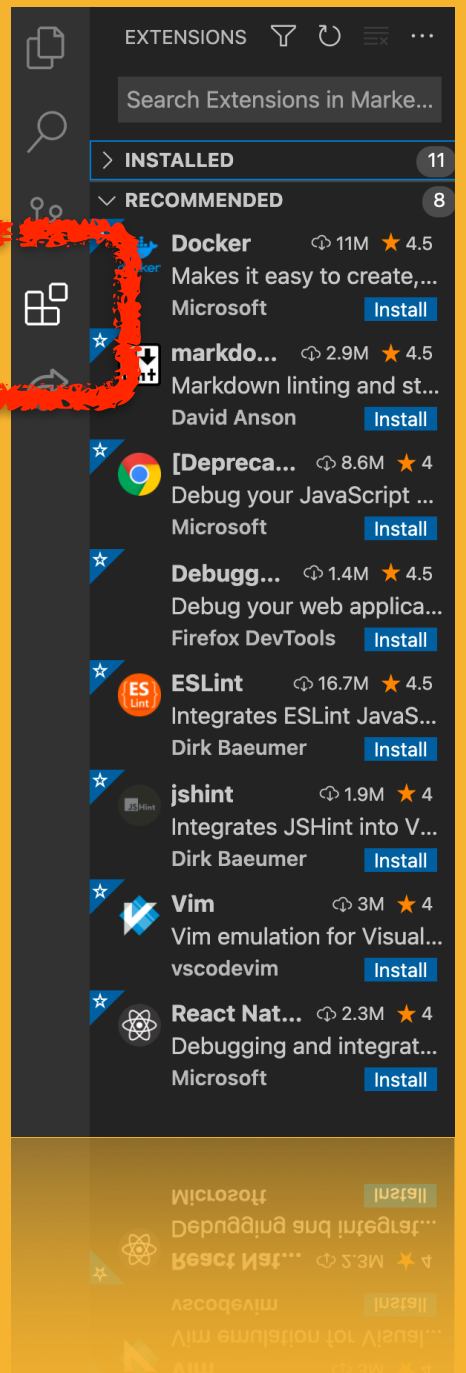
It's time to start revisiting what you've learnt and expanding those skills

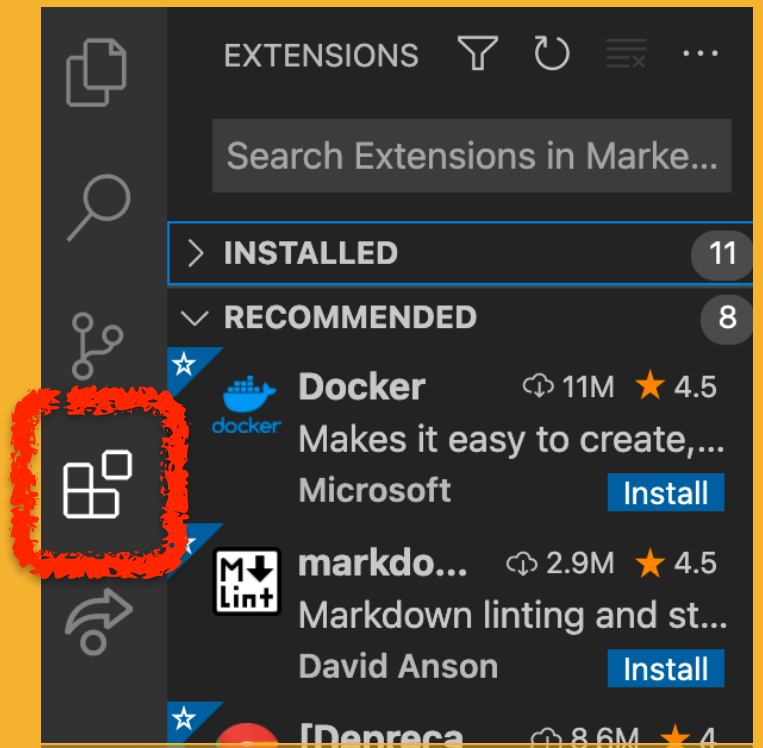
Make sure you're on the latest version of Python*

python™

*Current version 3.10.0

Get your coding environment set up like a professional



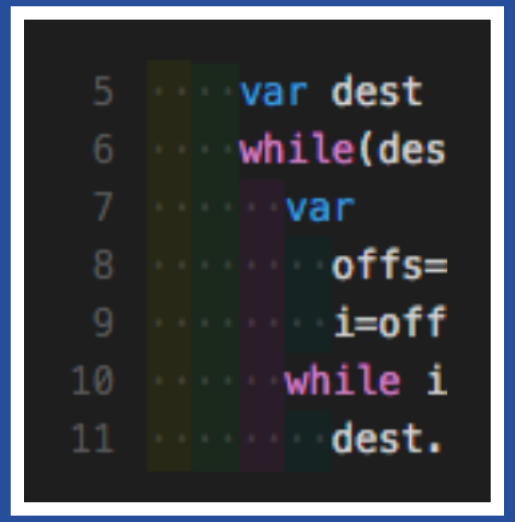


Navigate to the Extensions tab in the Activity Bar on the left hand side

Recommended extensions

Better Comments

Allows you to categorise your comments and make them easier to read



Indent Rainbow

Make your indentation more readable by colourising the different indents

Recommended extensions

Bracket Pair Colorizer 2

Matches and colours brackets to be more easily identifiable

...plus many more

These are just what your instructors will be using, feel free to customise your code editor how you like

Let's get started

Windows

Copy anything

}Use the keys `ctrl, c`

Paste

}Use the keys `ctrl, v`

Cut

}Use the keys `ctrl, x`

Undo

}Use the keys `ctrl, z`

Save your work

}Use the keys `ctrl, s`

Mac

Copy anything

}Use the keys `cmd, c`

Paste

}Use the keys `cmd, v`

Cut

}Use the keys `cmd, x`

Undo

}Use the keys `cmd, z`

Save your work

}Use the keys `cmd, s`

Test your program

Windows

Open the terminal

- Use the keys `ctrl, j`

Run the code

- Use the command `py` followed by your file name: `py filename.py`

Mac

Open the terminal

- Use the keys `cmd, j`

Run the code

- Use the command `python3` followed by your file name: `python3 filename.py`

#Comments

Windows

Add a comment

- Select your code
- Use the keys `ctrl, /`

Mac

Add a comment

- Select your code
- Use the keys `cmd, /`

Let's get warmed up

Give this a go.

```
greeting = "Hello world"

print(greeting)
```

What's happening here?

What do you remember?

```
greeting = "Hello world"

print(greeting)
```

What technical terms can you recall?

What do you remember?

```
greeting = "Hello world"
print(greeting)
```

variable

String data type

Data types, properties and methods

Boolean

String

Integer

None

Floating point

Boolean

String

Any text represented within quotes

Integer

None

Floating point

Boolean

True or False

String

Any text represented within quotes

Integer

None

Floating point

Boolean

True or False

String

Any text represented within quotes

Integer

Whole numbers

None

Floating point

Boolean

True or False

String

Any text represented within quotes

Integer

Whole numbers

None

Nothing - a null value (Falsy)

Floating point

Boolean

True or False

String

Any text represented within quotes

Integer

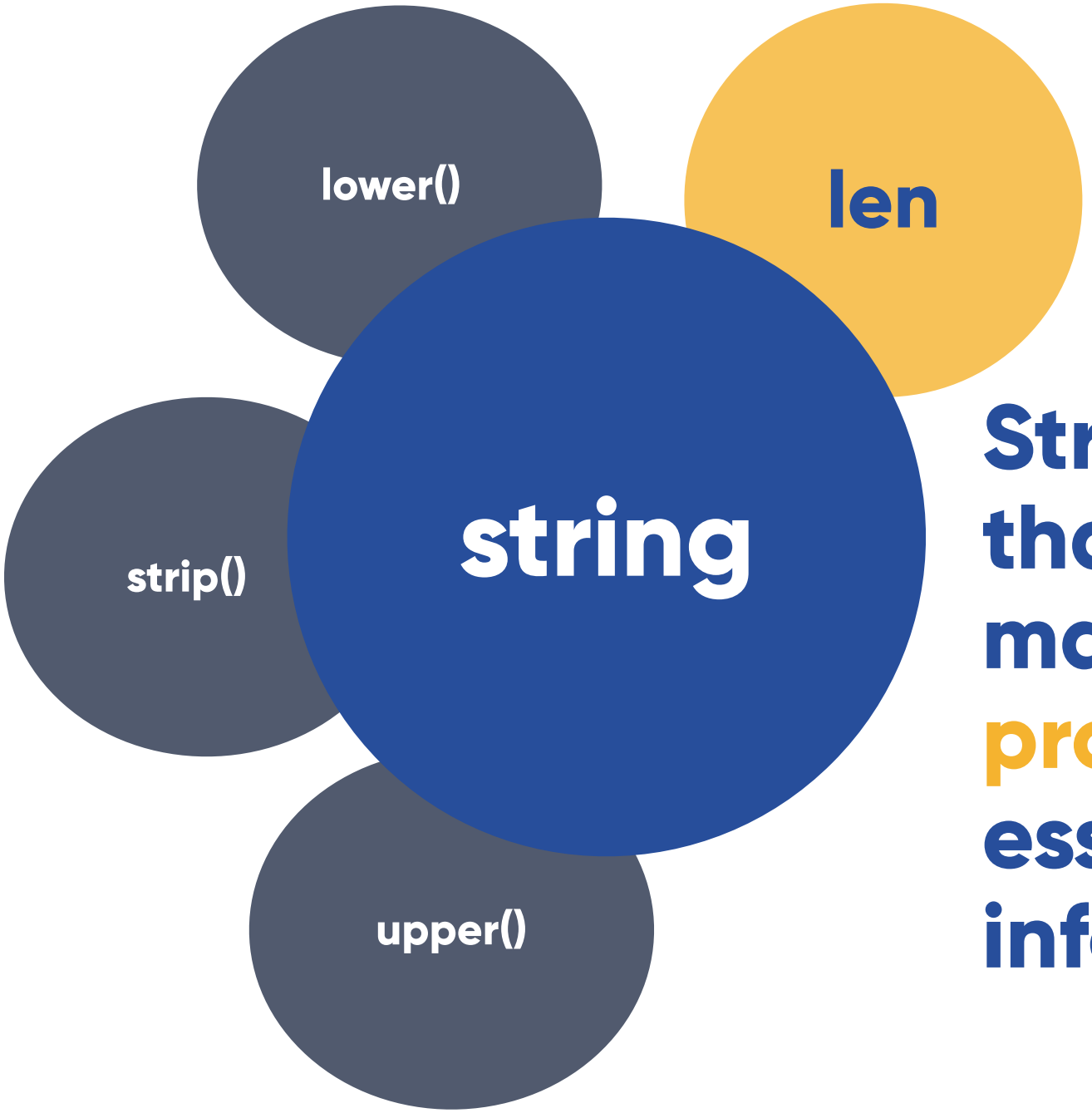
Whole numbers

None

Nothing - a null value (Falsy)

Floating point

Numbers with decimal points



Strings have **methods** that we can use to manipulate them, and **properties** which are essentially just information

```
print(len(greeting))
```

*finding the length property of this string

Finding a particular character in a string

```
print(greeting[1])
```

*finding the first character of this string, note that index begins at 0

```
print(greeting.upper())
```

*forcing the string to go into upper case

Other methods we looked at

- lower()
- capitalize()
- count()
- find()
- replace()
- strip()

Libraries

Libraries

```python
import random

print(random.random())
#Generates a random number between 0 and 1, including 0 only.

print(random.uniform(1, 10))
#Generates a random number between 1 and 10, inclusive.

print(random.randint(1,10))
#Generates a random integer between 1 and 10, inclusive.
```

Libraries

- `random` is a library in python which you must `import`
- `random, uniform, randint` are methods in the `random` library
  - `random.random()`
  - `random.uniform(1,10)`
  - `random.randint(1,10)`
- They may or may not need parameter(s)

To the next level...

```
import random

print(random.random())
print(random.uniform(1, 10))
print(random.randint(1,10))
```

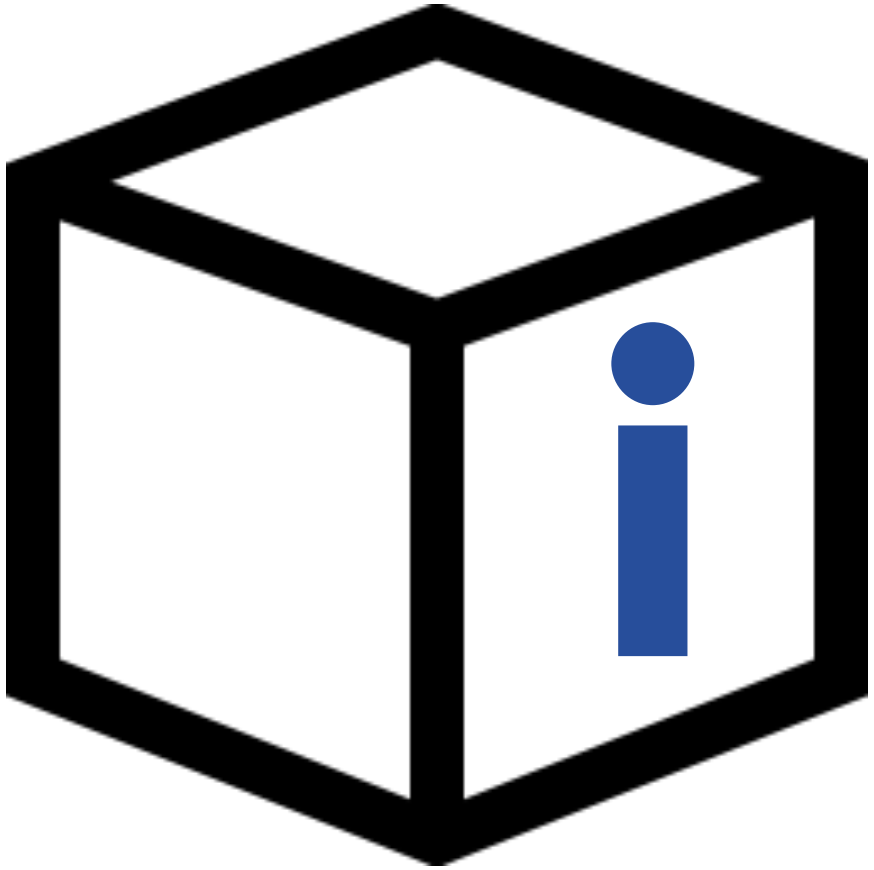
```
from random import random, randint, uniform

print(random())
print(uniform(1, 10))
print(randint(1,10))
```

Both are the same!

We'll come back to look at some more **libraries** later as we **level up** our Python knowledge 💪

Variables



We store items in boxes to retrieve later

Different items can be stored in the box at different times

In code, we give variables names so we can access things inside them. Exactly like saying “get me that thing from the **blue** box over there”

snake_case

```
my_name = "Ann"
my_age = 18
student = True
```

```
greeting = "hello"

print(greeting, "it’s nice to meet you!")
```

```
greeting = "hello"

print(greeting + "there, nice to meet you!")
```

```
greeting = "hello"

print("{} there, nice to meet
you!”.format(greeting))
```

```python
greeting = "hello"

print(f"{greeting} there, nice to meet you!")
```

=

Assignment Operator

Arithmetic Operators for calculations

+

−

*

**

/

%

Assignment Operators to store values

=

*=

+=

/=

-=

Input

```
response = input("How would you like to
respond? \n")
```

`input()` allows whatever a user types into the terminal to be saved to a variable - in this case `response`

```
response = input("How would you like to respond? \n")
```

```
response = input("How would you like to respond? \n")
```

`\n` in Python is the new line character

It's not mandatory when using input, but it makes typing in the terminal a bit easier.

```
response = input("How would you like to
respond? \n")

print(f"How did the user respond?:
\n'{response}'")
```

Print the user's response, or just save it for later

```
response = input("How would you like to respond? \n")

print(f"How did the user respond?: \n'{response}'")
```

```
response = input("How would you like to
respond? \n")

print(f"How did the user respond?:
\n'{response}'")

#Expected output:

How did the user respond?:
 'like this'
```

If else

In the condition we have

expression
To Be
Evaluated

logicalOperator
and/or

expression
To Be
Evaluated

```python
music = "classical"

if music == "classical":
    print("Oh no! It's classical music again.")
elif music == "no music":
    print("Ahh, peace and quiet!")
else:
    print("Nice and noisy.")
```

Comparison Operators

==	Equal
!=	Not equal

>=

>

<=

<

Greater than or equal to : >=

Greater than : >

Less than or equal to : <=

Less than : <

```python
place = "MCR"
weather = "Cloudy"

if place == "MCR" and weather == "Sunny":
    print("Check again")
elif place == "MCR" and weather == "Rain":
    print("Obvs")
else:
    print("Wait, it isn't raining?")
```

```python
age = 20
country = "UK"

if age > 17 and country == "UK":
    print("Yes I can serve you")
elif age > 17 and country != "UK":
    print("Where are you?")
else:
    print("You aren't old enough")
```

```
day = "Saturday"

if day == "Saturday" or day == "Sunday":
    print(“It's the weekend!")
else:
    print("When's the weekend?")
```

and

True and True –> True

True and False –> False

False and False –> False

or

True or True —> True

True or False —> True

False or False —> False

Modulus %

Modulus shows the remainder of a division

```
print(10%2)
#Expected output: 0. The remainder of 10 divided
by 2 is 0.

print(10%3)
#Expected output: 1. The remainder of 10 divided
by 3 is 1.
```

Modulus %

We can incorporate if/else syntax to check if one number is divisible by another

```
print(10%2 == 0)
#Expected output: True. There is a remainder of 0
so 10 is divisible by 2.

print(10%3 == 0)
#Expected output: False. There is a remainder of
1 so 10 is not divisible by 3.
```

Functions

```python
def light_switch():
    print("Who turned out the lights?")

light_switch()
```

Function

```python
def cash_withdrawal(amount, accnum):
    print(f"Withdrawing {amount} from account {accnum}")

cash_withdrawal(300, 50449921)
```

Function with parameters

Let's take this in...

```
w_amount = 100
account_num = 12345678

def cash_withdrawal(amount, accnum):
    print(f"Withdrawing {amount} from account {accnum}”)

cash_withdrawal(w_amount, account_num)
cash_withdrawal(300, 50449921)
cash_withdrawal(30, 50449921)
```

Function with parameters
and variable parameters

Lists

```python
coffee_order = [
    "Alex – Cortado",
    "Ben – Latte",
    "Charlie – Whatever's new"
]

print(coffee_order)
```

```
print(coffee_order[2])
```

```
coffee_order = [
    "Alex – Cortado",
    "Ben – Latte",
    "Charlie – Whatever's new"
]

coffee_order[1] = "Ann – Vanilla latte”

print(coffee_order)
```

```
coffee_order = [
    "Alex - Cortado",
    "Ben - Latte",
    "Charlie - Whatever's new"
]

print(len(coffee_order))
```

```
coffee_order = [
    "Alex – Cortado",
    "Ben – Latte",
    "Charlie – Whatever's new"
]

coffee_order.append("Donna – espresso”)

print(coffee_order)
```

```python
coffee_order = [
    "Alex – Cortado",
    "Ben – Latte",
    "Charlie – Whatever's new"
]

coffee_order.pop()

print(coffee_order)
```

```
.remove()
.reverse()
.sort()
.count()
.extend()
```

so many...

Check out the Python Documentation for more.

https://docs.python.org/3/

For loops

```python
favourite_drinks = ["coke", "fanta", "tonic"]

for i in favourite_drinks:
    print(i)
```

```python
for i in range(10):
    print(i)

for i in range(0, 10):
    print(i)

for i in range(0, 10, 1):
    print(i)
```

While loops

```
num = 0

while num < 10:
    num += 1
    print(num)
```

```python
import random

rand_num = random.randint(0,50)

my_num = 50

while rand_num != my_num:
    print(rand_num)
    rand_num = random.randint(0,50)

print(f"You've found {my_num}!")
```

Learning Objectives

- Set up your programming environment
- Familiarise yourself with key coding concepts
- Revisit and refresh Python knowledge
- Complete activities to utilise your Python skills

{codenation}®

Let's ease you back into it...

Activity(1)

- Create variable that holds the text “Welcome to Code Nation”. Find the length of the string and save this to a new variable.
- Create a function that checks if the string length is even; if it is, print the string, the length and an appropriate message in one sentence. Do the same but with a different message if it’s odd.
- Change the string length so you can test all possible outcomes

Activity(2)

- Create a list that represents the alphabet:

```
alphabet = ["a", "b", "c", "d"...
```

- Create a for loop to iterate through this and print each letter in order
- Now using input, allow the user to type a number and print the letter it represents in the alphabet.
- Remember how index works – and think about how to structure your code

Activity(3)

- Remember the noughts and crosses activity? Let's revisit that and start to improve with our improved knowledge.
- Create a structure of functions that allow the player to play against the computer – here is a suggestion:
  - Function to start the game, let player choose '0' or 'X'
  - Function to print the board & show the player how to choose spaces
  - Function for the player to choose their space
  - Function for the computer to take its turn
  - Function to check the logic of if there's a win, lose or draw after every turn is taken