Variable Types

Variables: Store a value
Strings: Enclosed in quotation marks
    Concatenation: `wizard = harry’+‘ ’+‘potter’`
Booleans: True or False
Integers: Whole numbers, ex. `1, 54 or -3`
Floats: Numbers with decimals, ex. `3.1415`

Mathematical operations

Addition: `+`
Subtraction: `-`
Multiplication: `*`
Division: `/`
Exponent: `**`

Printing and Input

`print(“Hello World”)` can be used to print the string: Hello World
`X = input(‘the answer to this question will be saved as variable x)` can be used to take in a value
Note: Input always assumes values are strings

Type Casting

`int()` turns a variable into an integer
`float()` turns a variable into a float
`str()` turns a variable into a string

Ex.
```
Age = input(“How old are you?”)
Age = int(Age)
```

Lists

Create a list: `ice_cream = [‘chocolate’, ‘vanilla’, ‘cookies and cream’]`
    Or: `numbers = [x for x in range(0,10)]`
Add to a list: `ice_cream.append(‘sorbet’)` or `ice_cream = ice_cream + [‘sorbet’]`
Get the first item in a list: `first = ice_cream[0]`
Get the last item in a list: `last = ice_cream[-1]`
Get the item located at the ith position of a list (indexing): `favourite = ice_cream[2]`
Remember in programming everything starts at zero, so the first item has an index of zero, and if a list has three items, the index of the last item is 2
Slicing a list: `first_half = ice_cream[:1]`
    `last_half =ice_cream[2:3]`
Copying the list: `copy = ice_cream[:]`
Never copy a list using `copy = original` as this means when one is changed, the other might be as well.

<u>Conditionals</u>

`If`
`Elif`
`Else`

Types of conditionnel

```
      equals: ==
      Not equal: !=
      Greater than or equal to: >=
      Greater than: >
      Less than or equal to: <=
      Less than: <

      Use a boolean
            Ex. cleared = True
                  if cleared:
                        #Do something …
                  else:
                        #exit
```

<u>Linking Conditions</u>

`Or` - if **either** of the conditions is **true**, proceed
`And` - proceed only if **both** of the conditions are **true**
`Not -` if the condition is **false,** proceed

<u>Loops</u>

For loop

```
      For i in range(start,finish,increment):
```

You can also go backwards through the list if the increment is negative - just remember to reverse the start and finish

While loop

```
      While condition:
            #perform an action repeatedly
            i = i+1
            #if something changes
            if i>100:
                  break
```

`break` can be used in both while loops and for loops to exit immediately