FOUNDATIONS CONTENT

What's a computer program?

\n new line – a print statement adds a new line

sep=' ' separator

You can multiply and add strings together (* and +). Note + joins without space.

Unix shell: It interprets commands typed by the user, then executes those commands.

$ indicates prompt – the shell is ready for you to type!

ls lists – shows what files are in the current directory (folder)

cat <filename> prints out the data in a file. aka concatenate

mkdir <name> makes a new directory (folder)

cd changes the working directory to the specified directory

cp copy

e.g. $ cp myfile Newdir copying the contents of myfile to the new folder Newdir.

echo <words> repeats the words passed in as arguments

touch <filename> creates a file

rm <filename> deletes a file

Bash: A variety of shell programs. It reads a line of input, separates it by spaces, and treats the first word as the command to search for the program to run. However, there are a few commands that are executed by the shell itself, not by a separate program. For example, the cd command. Bash also allows for if statements and while loops, making it a full-fledged programming language.

Streams: The shell connects the keyboard and screen to the program. Keyboard inputs are referred to as "Standard input" or stdin, and the screen outputs are referred to as "Standard output" or stdout. The shell allows you to redirect these streams, changing the input or output (or both) to be a file instead. To change the output, use the greater-than symbol (>) after the command.

e.g. To save the text "Hello world!" to a file named "newfile", use the following command: $ echo hello world! > newfile

Special File System Names

- `..` : The parent directory. `cd ..` moves you back to the parent directory (parent folder).
- `.` : The current directory.
- `-` : The previous directory. `cd -` allows you to "undo" the last `cd` command.
- `~` : The home directory. `cd ~` or just `cd` takes you to your user folder.
- `/` : The root directory, where the file system begins.

Variable and Data Types

Variable names must start with a letter or underscore. No spaces or special characters!

e.g year = 2025 #This is an assignment statement.

PEP-8 Naming Conventions:
Avoid using variable names that already exist as other objects in Python because they are already attached to existing Python variables or functions.
Variable names should have all alphabetical characters in lowercase, and spaces (if any) should be replaced with underscore characters.

e.g. colour or _colour rather than Co10Ur

Types of data

Python is a dynamically typed language, which means that the same variable name can be used to refer to different types of data when the program is running – but this is not the same for other languages.

Integer: whole numbers
Float: decimal numbers #(Data is binary, so float is stored as scientific notation to allow us to store a wide range of values in a limited storage space. However, the accuracy suffers as computer memories are finite; limited number of bits)
String: sequence of characters
#You can perform operations like converting to lowercase. “Hello”.lower()
Boolean: one of 2 values – True or False
Compound types: e.g. lists – ls = [1, ‘abc’, 3.14]
User defined: custom types. Aka class

#You can use the type() function to check the data type of an object.

Python objects have meta-data associated with them, being is variable name, the value it stores, its type of data, and its operations.

Type hints (optional, but good practice): str, int, float, list, dict (dictionary)

e.g.
VarName: *type*

VarName: *type* = value
VarName: list[*type*] #To hint for lists.
VarName: list[*type*] = [values, ...]

def Fun(param1: *type,* param2: *type*) -> *type*: #To hint for functions.

mypy is a program that will check the type hint information in your python program. It analyses your program for type errors.

The escape character \

It tells your program to interpret the next sequence of characters in a unique way if there is one defined.

```
1 print('Santa\'s biscuits')
2 print("Santa's biscuits")
```

```
Santa's biscuits
Santa's biscuits
```

#Want to add a tab? Do \t

Operators and Expressions

```
weight = 76
height = 1.89
bodyMassIndex = weight/(height**2)        #** is exponentiation operator
```

weight/(height**2) is an expression. The / and ** are operators.

- Symbol: Operators
- `+` : add
- `-` : subtract
- `/` : divide
- `*` : multiply
- `**` : exponentiation (raise to a power)
- `%` : modulus (the remainder after a division)
- `//` : integer divide (integer value lower than the result)

Boolean Operators

Boolean data type variables are either True or False.

Comparison operators (>, ≥, ≤, ==, !=) will always evaluate to a Boolean.

There are 3 Boolean operators: not, and, and or.
not inverts a Boolean.

and evaluates to True if both operands are True, else it evaluates False.
or evaluates to True if either operand is True.

If there are no brackets, Python performs the operations in the order of NAO.

Strings and TypeErrors

Using + between two strings concatenates them. Using *<integer> repeats the string.

Type errors occur when an operation is performed on data types that are not compatible with that operation.

You cannot:

- Multiply strings by a float or another string
- Add a string and an integer
- Multiply strings with floats or stings

len(): returns the number of items/length of an object, typically used with sequences and collections.

upper(), lower(), count(<*letter*>), find(<"word">), replace("<word 1>", "<word 2>)

string[::-1] to reverse a string

Comments:

Use # at the start for single-line comments.
Use '' or """ for multi-line comments. (encases the comment)
Triple quotes can also be used to initialise strings:

```
description = '''
This is a
multiline string
'''
```

The input() function:

A function that will read one line from the standard input.

e.g.
>>>myline = input()
Hello
>>> print(f"Input was <{myline}>")
Input was <Hello>

Prompt:

You can pass a string to `input()` to display a prompt to the user.

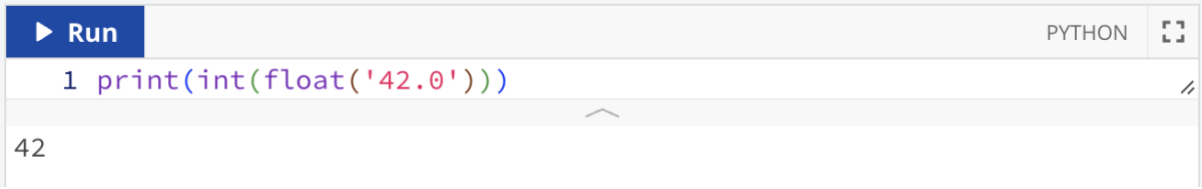
```
>>> name = input("Who are you? ")
Who are you? Fred
>>> print("Hello", name)
Hello Fred
```

input() will only return the string that was entered, not a number, so you need to convert the string that was read.

```
>>> size = input("How big in cm? ")
How big in cm? 123
>>> print("The size is ", int(size))
The size is 123
```

Type Casting; Using int() & float() # must use a numeric value/numeric strings

A string containing decimal points is considered a non-numeric string. You can still convert `42.0` to an integer but you would have to first convert it to a float.



This allows you to convert data from one type to another.

int() doesn't round, it simply truncates the decimal part of a float. It gets rid of 0

Output formatting

```
1 name = 'Iris'
2 age = 23
3
4 message = 'Hi {}, you are {} years old.'.format(name, age)
5 print(message)
```

```
Hi Iris, you are 23 years old.
```

You can use "...some string...".format(), or f-strings.

e.g. text1 = f"Hi {name}, you are {age}."

text2 = f""""Hello, {people}!

How is {}?"""" Putting the triple quote marks under/over will put a \n character.

e.g.

```
word = input("What is your favorite word? ")
print("The length of that is", len(word))
```

This prints: The length of that is <number>.