— First Steps —

The Zen of Python

```
import this
```

Interpreter Prompt

```
# python3 into terminal to start a REPL (Read-Eval-Print Loop) session
print("Hello World!")
quit() # quit REPL session. Alternative Ctrl-D,Z,C
```

Editor and Source File

TODO

- mkdir code
- cd code
- touch hello.py

```
# Save this statement to hello.py
print("Hello World from a source file")
# In terminal, type python3 hello.py
```

```
!mkdir code
!touch code/hello.py
!echo 'print("Hello World!")' >> code/hello.py
!python3 code/hello.py
```

Getting Help

```
help(print)
print('*********************************************')
help('keywords')
```

— BASICS PYTHON 1 —

print()

```
print("1729 is a boring number, isn't it", '?')
print('No! ', end = '')
print('It is the smallest number that can be expressed as sum of 2 cubes in 2 different ways')
print('1','2','3', sep = ' < ')
print('1','2','3', sep = ' > ', end = ' \U0001F914')
```

Comment

```
# "The 'Hello World' example is the traditional incantation to the programming gods
# and will ensure your quick mastery of the language,
# so please make sure you actually do this exercise, instead of just reading about it."
print('Hello World!') # said Simon Cozens in "Beginning Pearl"
```

Literal Constants

```
 # Number

# integer
1
1000000000000000000000000000000000000000
print(type(1000000000000000000000000000000000000000))

# float
2.1
0.999999
0.9999999999999999999999999999999999999
1.729e3
1.729e3
print(type(1.729e3))

# complex number
3+4j
print(type(3+4j))
```

```
 # String
'Hello World!'
"Hello World!"
"""Hello

World!"""
print(type("Hello World!"))

print("""Hello,

World!""")
```

```
 # Boolean
True
print(type(True))
False
```

Basic Operation with Numbers

```
 # Arithmetic Operation and Expression
print('17 + 29 =', 17+29)
print('17 - 29 =', 17-29)
print('17 + 29 =', 17*29)
print('17 / 29 =', 17/29)
print('17 // 29 =', 17//29)
print('17 % 29 =', 17%29)
print('17 ** 29 =', 17**29)

(1+0.01)**365

print(1**3+12**3)
print(9**3+10**3)
```

The mistery uncovered

\begin{equation}*} 1729 = 1^3 + 12^3\ 1729 = 9^3 + 10^3 \end{equation*}

```
 # Order of Evaluation
print(2+4*4/2**3)
print((2+4)*4/2**3)

# loss-of-precision
print({:.20f}.format(0.1))
0.1.as_integer_ratio()
bin(0.1.as_integer_ratio()[1])
print((0.1+0.1+0.1) == 0.3)
# overflow
1.7e308
1.8e308
# underflow
5e-324
1e-325
```

Basic Operation with Strings

```
 'coding'+'cat'
'cat'+str(3)
'cat'*3
'codingcat'[6:]
```

Escape Sequence

Escape sequence	Effect
\'	Single Quote
\\	Backslash
\n	New Line
\r	Carriage Return
\t	Tab
\b	Backspace
\f	Form Feed
\ooo	Character based on Octal value
\xhh	Character based on Hex value
\uxxxx	Unicode character with 16-bit hex value xxxx
\Uxxxxxxxx	Unicode character with 32-bit hex value xxxxxxxx
\N{name}	Character from Unicode database with given name

```
 print("\"")
print("\\")
print("1234\rabc")
print("1234\b\b")
print("\044")
print("\x32")
print("\u041b")
print("\U000001a9")
print("\N{face with tears of joy}")
```

Format String

```
import math
'{:.200f}'.format(math.pi)
f'This is week{2}'
```

DISPLAY multiplication_table.py

TODO rectangle.py

Given width and height of a triangle, print out its Perimeter and Area

Data Types

```
dir(str)
```

```
# Look up for str() attributes and functions
help(str)
help(str.__add__)
help(str.zfill)
```

```
# Explore what we can do with str()
"Hello".upper()
"123".zfill(10)
```

Casting

```
# str()
str(1729)
```

```
# int()
int('1729')
int('a')
```

```
# float()
float(1729)
float('1729')
```

```
# complex()
complex(1729)
complex(17.29)
complex('17+29j')
```

Variable

```
# Assignment
x = 1729
print("Assigned x =",x)

pi = π = Π = 3.14
print(f"Assigned pi = {pi}, π = {π}, Π = {Π}")
```

```
# Name convention
xy1 = 10
_xy1 = 20
xY1 = 10
xY_1 = 10
```

```
1xy = 10
class = 2024
```

```
# No keyword in naming variable
help('keywords')
```

TODO rectangle.py

Modify the rectangle.py script, now use variable to represent width and height lengths

Built-in function

1

Some common use functions:

Math

Function	Description
abs()	Returns absolute value of a number
divmod()	Returns quotient and remainder of integer division
max()	Returns the largest of the given arguments or items in an iterable
min()	Returns the smallest of the given arguments or items in an iterable
pow()	Raises a number to a power
round()	Rounds a floating-point value
sum()	Sums the items of an iterable

Type Conversion

Function	Description
ascii()	Returns a string containing a printable representation of an object
bin()	Converts an integer to a binary string
bool()	Converts an argument to a Boolean value
chr()	Returns string representation of character given by integer argument
complex()	Returns a complex number constructed from arguments
float()	Returns a floating-point object constructed from a number or string
hex()	Converts an integer to a hexadecimal string
int()	Returns an integer object constructed from a number or string
oct()	Converts an integer to an octal string
ord()	Returns integer representation of a character
repr()	Returns a string containing a printable representation of an object
str()	Returns a string version of an object
type()	Returns the type of an object or creates a new type object

Input/Output

Function	Description
format()	Converts a value to a formatted representation
input()	Reads input from the console
open()	Opens a file and returns a file object
print()	Prints to a text stream or the console

Variables, References, and Scope

Function	Description
dir()	Returns a list of names in current local scope or a list of object attributes
globals()	Returns a dictionary representing the current global symbol table
id()	Returns the identity of an object
locals()	Updates and returns a dictionary representing current local symbol table
vars()	Returns **dict** attribute for a module, class, or object

```
 i = input("input: ")
print(i)
```

TODO rectangle.py

Modify the script, now use user input for width and height lengths

TODO square.py

Given the Perimeter of a Square as user input, print its Area - Write docstring

TODO circle.py

Given the Circumference of a Circle as user input, print its Area - Write docstring

TODO regular_polygon.py

Write a script that receives the user inputs as initial Perimeter of a polygon and its number of vertices, then print out the polygon's Area - Write docstring - Check solution from **square.py** and **circle.py**

DISPLAY plot_polygon.py

Reference:

Format string

realpython.com: https://realpython.com/learning-paths/python3-introduction/:

- Interacting with Python
- Basic Python Data Types
- Variables in Python

A Byte of Python: https://python.swaroopch.com/

- Installation
- First Steps
- Basics
- Operators and Expression

List of Python built-in functions and their use: https://docs.python.org/3.6/library/functions.html#exec

Floating Point Arithmetic: Issues and Limitations in Python: https://docs.python.org/3.6/tutorial/floatingpoint.html