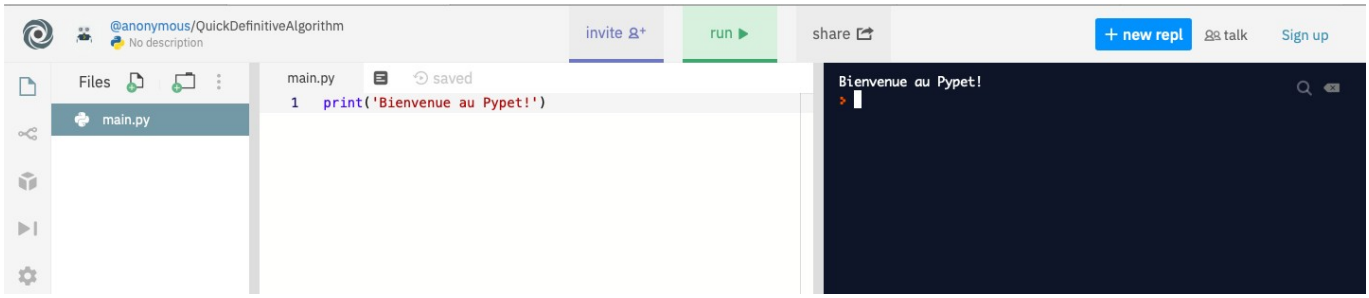
Pypet

Lancer Python pour la première fois

1. Créez un nouveau fichier repl.it et entrez le code ci-dessous:

```
print('Bienvenue au Pypet!')
```

2. Maintenant appuyez sur le bouton « Run » en haut de l'écran. La console va écrire "Bienvenue au Pypet!".



Vous venez d'écrire votre premier print ! Print est une fonction python qui écrit des choses dans la console. C'est vraiment pratique pour apprendre le python et débugger votre code.

Créer votre Pypet

Variables

Les variables sont une manière de stocker de l'information dans python. On va créer différentes variables pour notre Pypet comme `nom`, `poids`....

1. Créez une variable appelée `nom` égale à 'Fluffy' (ou 'Sonic' ou 'Coco' ou 'Medor'...)

```
nom = 'Fluffy'
```

Utiliser le signe égal (=) affecte à la variable une valeur comme `nom = 'Fluffy'`.
Les variables peuvent contenir différents types de données. Ici, `nom` est quelque chose appelé une chaîne de caractère (**string**) parce que `'Fluffy'` est entouré de guillemets. Un **string** est seulement constitué de caractères entourés de guillemets (par exemple `'Bob'`, `'New York'` or `'h4ck3r'`). *NOTE: On peut utiliser pour les chaînes de caractères les simples guillemets ou les doubles guillemets en python.* Un **string** peut aussi contenir des nombres tant qu'ils sont entre guillemets. Les entiers (integers) par contre n'ont pas besoin de guillemets. Regardons quelques autres types de données.

2. Créez trois autres variables comme `age`, `poids` et `faim`.

```
nom = 'Fluffy'
age = 5
poids = 3.5
faim = True
```

La variable age est un entier (**integer**) et donc il n'y a pas de virgule. La variable poids est un décimal (**float**) . Les décimaux sont des nombres qui ont des valeurs après la virgule (Attention, en python on note la virgule avec un point!!) . La variable faim est un booléen (**Boolean**). Les booléens contiennent soit `True` soit `False`.
NOTE: Il ne faut pas utiliser de guillemets pour ce type de données, sinon ils seront considérés comme des strings.

3. Choisissez la 'photo' de votre Pypet. Vous trouverez quelques exemples ci-dessous mais laissez libre court à votre imagination. *NOTE: Essayez de ne pas dépasser une ligne pour la 'photo' de votre Pypet, ce sera plus facile de suivre les instructions suivantes.*

Pychat

```
(=^o.o^=)__
```

Pysouris

```
<:3 )~~~~
```

Pypoisson

```
<`)))><
```

Py? Vous choisissez!

```
(^0M0^)
```

4. Maintenant ajoutez une autre variable qui sera une chaîne de caractère (***string***) contenant la photo de votre pet.

```
nom = 'Fluffy'
age = 5
poids = 3.5
faim = True
photo = '(=^o.o^=)__'
```

5. Ajoutez quelques print à votre code pour voir votre Pypet dans la console.

```
nom = 'Fluffy'
age = 5
poids = 3.5
faim = True
photo = '(=^o.o^=)__'

print('Hello ' + nom)
print(photo)
```

Remarquez l'espace après le Hello et avant l'apostrophe...

En écrivant `print('Hello ' + nom)` on a concaténé (c'est-à-dire lié ensemble) le string `'Hello'` avec la variable nom pour que la console écrive `Hello Fluffy`. N'oubliez pas de cliquer sur 'Run' pour lancer votre code.

Dictionnaires

On a besoin de dire à python que toutes ces variables représentent un chat (ou un chien, un poisson, une créature...). Une solution est d'utiliser un dictionnaire (**dictionary**). Les dictionnaires sont une façon de stocker des variables multiples qui contiennent des valeurs différentes.

1. Placez vos variables dans un dictionnaire. Essayez d'utiliser des valeurs différentes que celles ci-dessous pour personnaliser votre pypet.

```
chat = {
    'nom': 'Fluffy',
    'faim': True,
    'poids': 3.5,
    'age': 5,
    'photo': '(=^o.o^=)__',
}
```

On a créé ici un dictionnaire appelé `chat`. Chaque ligne contient un attribut de chat. Les attributs ont tous une clé (**key**) (ex 'nom', 'poids') et une valeur (**value**) (ex 'Fluffy', '9.5', True…). Dans un dictionnaire, les attributs utilisent les deux points : et une virgule, après chaque valeur ('nom' : 'Fluffy',).

2. Ajoutez un `print` pour voir votre nouveau dictionnaire pypet dans la console.

```
chat = {
    'nom': 'Fluffy',
    'faim': True,
    'poids': 3.5,
    'age': 5,
    'photo': '(=^o.o^=)__',
}

print(chat)
```

3. Affichez le nom de votre pypet ainsi que sa photo. Vous pouvez accéder aux variables dans un tableau en utilisant le format `dictionnaire['attribut']` comme `chat['nom']`.

```
chat = {
    'nom': 'Fluffy',
    'faim': True,
    'poids': 3.5,
    'age': 5,
    'photo': '(=^o.o^=)__',
}

print('Hello ' + chat['nom'])
print(chat['photo'])

print(chat)
```

Nourrir votre Pypet

Fonctions

Allons 'nourrir' notre pypet en utilisant une fonction python. Une fonction est un bloc de code organisé et réutilisable qui sert à réaliser une action. D'abord, nous devons définir notre fonction — `nourrir` — qui changera l'attribut 'faim' de notre pypet en `False` pour montrer qu'il n'a plus faim.

1. Créez cette fonction en rajoutant ces lignes sous votre code.

```
def nourrir(pet):
    pet['faim'] = False
```

Il y a plusieurs choses à remarquer ici.

En écrivant `def nourrir(pet):` on définit une fonction appelée `nourrir` qui accepte un paramètre `pet`. On voit aussi qu'on a indenté la ligne suivante `pet['faim'] = False`.

NOTE: En python, il faut indenter le contenu d'une fonction.

2. Ajoutez `nourrir(chat)` sous votre fonction pour utiliser la fonction nourrir sur votre pypet, ici le chat.

```
def nourrir(pet):
    pet['faim'] = False
nourrir(chat)
```

En appelant `nourrir(chat)` on passe la variable `chat` dans la fonction à la place de `pet`. `pet` agit comme un paramètre fictif pour n'importe quelle variable que l'on passera dans la fonction.

On devrait aussi augmenter un peu le poids du pypet puisqu'il a mangé.

3. Ajoutez `pet['poids'] = pet['poids'] + 1` à votre fonction `nourrir`.

```
def nourrir(pet):
    pet['faim'] = False
    pet['poids'] = pet['poidst'] + 1
```

On utilise cette notation pour augmenter la valeur des décimaux et des entiers.

4. Mettez la valeur initiale de `faim` de votre pypet à `True` et ajoutez un a `print chat` après `feed(cat)` pour voir si la variable `faim` de votre pypet change en `False` et si le `poids` augmente.

```
print( 'Bienvenue au Pypet!')

chat = {
    'nom': 'Fluffy',
    'faim': True,
    'poids: 3.5,
    'age': 5,
    'photo': '(=^o.o^=)__',
}

def nourrir(pet):
    pet['faim'] = False
    pet['poids'] = pet['poids'] + 1

print chat
nourrir(chat)
print(chat)
```

Quand le chat est affiché la deuxième fois, son poids aura augmenté. N'oubliez pas de cliquer sur 'Run' pour lancer votre code.

Condition If

Mais que se passe-t-il si votre pypet n'a pas faim ? On doit prendre en compte si notre variable `faim` a comme valeur `True` ou `False`. Pour savoir si notre pypet a faim, on va utiliser la condition **if**. En python, la condition **if** vérifie si une condition est réalisée ou non (comme si oui ou non faim = True).

Si le pypet a faim, le programme va mettre la variable `hungry` à `False` et augmenter son `poids`. Si le pypet n'a pas faim, il va écrire dans la console `Le pypet n'a pas faim !`.

1. Ajoutez une condition if dans votre fonction.

```
def nourrir(pet):
    if pet['faim'] == True:
        pet['faim'] = False
        pet['poids'] = pet['poids'] + 1
    else:
        print(''Le pypet n'a pas faim!'')
print(chat)
nourrir(chat)
print(chat)
```

Attention on utilise des guillemets double car il y a une apostrophe dans la phrase sinon il faudrait écrire 'Le pypet n\'a pas faim'

Remarquez qu'on utilise deux signes égals (==) pour vérifier une condition (par exemple `pet['faim'] == True`). Si la condition n'est pas vérifiée alors le code après `else:` s'éxécute. Rappelez-vous, un signe égal est utilisé pour assigner une valeur à une variable (`pet['faim'] = True` rend notre Pypet affamé), deux signes égals sont utilisés pour vérifier si une condition est vrai (`pet['faim'] == True` vérifie si notre Pypet a faim).

2. Ajoutez un autre `nourrir(chat)` sous votre fonction et essayez de nourrir le chat deux fois pour voir si la fonction marche !

```
def nourrir(pet):
    if pet['faim'] == True:
        pet['faim'] = False
        pet['poidst'] = pet['poids'] + 1
    else:
        print(''Le pypet n'a pas faim!'')

print(chat)
nourrir(chat)
print(chat)
nourrir(chat)
```

Des amis pour votre Pypet

Lists

1. Créons un autre Pypet en utilisant un dictionnaire. Ajoutez (ou personnalisez) le code ci-dessous sous votre dictionnaire de pypet précédent.

```
souris = {
    'nom': 'Mickey',
    'age': 6,
    'poids': 0.5,
    'faim': False,
    'photo': '<:3 )~~~~',
}
```

NOTE: Faites bien attention de placer ce nouveau pypet avant votre fonction.

2. Créez une liste contenant vos deux pypets en utilisant `pets = [chat, souris]`.

```
pets = [chat, souris]
```

Maintenant que nous avons plus d'un pypet, nous pouvons les stocker dans une liste python. Une liste est un autre type de donnée. Les listes stockent des variables avec un ordre.

Boucles For

Comment faire si on veut nourrir tous les pets de notre liste ? Si on veut utiliser une fonction pour chaque variable d'une liste, on peut utiliser quelque chose en python appelé une boucle. La boucle for en python peut répéter une même séquence sur des éléments, comme ceux d'une liste par exemple.

```
for pet in pets:
    nourrir(pet)
    print(pet)
```

Bonus

Les étapes précédentes terminées, faites preuve d'imagination ! Voici quelques idées :

- Faire une variable qui garde en mémoire les points de vie
- Créer une variable booléenne pour savoir si le pet est endormi
- Créer une fonction jouer()
- Créer une liste de phrases que votre pypet dira au hasard
- Demander quelque chose à l'utilisateur avec input()
- ...